SQreamDB

Release 4.6

SQreamDB Documentation

Mar 18, 2024

CONTENTS:

1 Getting Started **3**
- 1.1 Preparing Your Machine to Install SQreamDB . . . 3
- 1.2 Installing SQreamDB . . . 3
- 1.3 Executing Statements in SQreamDB . . . 4
- 1.4 Performing Basic SQream Operations . . . 4
  - 1.4.1 Running the SQream SQL Client . . . 4
  - 1.4.2 Creating Your First Table . . . 4
  - 1.4.3 Listing Tables . . . 6
  - 1.4.4 Inserting Rows . . . 6
  - 1.4.5 Running Queries . . . 7
  - 1.4.6 Deleting Rows . . . 8
  - 1.4.7 Saving Query Results to a CSV or PSV File . . . 9
- 1.5 Hardware Guide . . . 10
  - 1.5.1 Cluster Architectures . . . 11
    - 1.5.1.1 Single-Node Cluster . . . 11
    - 1.5.1.2 Multi-Node Cluster . . . 12
    - 1.5.1.3 Metadata Server . . . 12
    - 1.5.1.4 SQreamDB Studio Server . . . 13
  - 1.5.2 Cluster Design Considerations . . . 13
    - 1.5.2.1 Balancing Cost and Performance . . . 13
    - 1.5.2.2 CPU Compute . . . 14
    - 1.5.2.3 GPU Compute and RAM . . . 14
    - 1.5.2.4 RAM . . . 14
    - 1.5.2.5 Operating System . . . 14
    - 1.5.2.6 Storage . . . 14
- 1.6 Staging and Development Hardware Guide . . . 14
  - 1.6.1 Development Desktop . . . 15
  - 1.6.2 Lab Server . . . 15

2 Installation Guides **17**
- 2.1 Installing and Launching SQreamDB . . . 17
  - 2.1.1 Pre-Installation Configuration . . . 17
    - 2.1.1.1 Recommended Basic Input/Output System Settings . . . 17
    - 2.1.1.2 Installing the Operating System . . . 19
      - 2.1.1.2.1 Before You Begin . . . 19
      - 2.1.1.2.2 Installation . . . 19
    - 2.1.1.3 Configuring the Operating System . . . 20
      - 2.1.1.3.1 Creating a `sqream` User . . . 20
      - 2.1.1.3.2 Setting Up A Locale . . . 21
      - 2.1.1.3.3 Installing Required Software . . . 21

2.1.1.3.3.1 Installing EPEL Repository . . . 21
2.1.1.3.3.2 Enabling Additional Red Hat Repositories . . . 21
2.1.1.3.3.3 Installing Required Packages . . . 22
2.1.1.3.3.4 Installing Recommended Tools . . . 22
2.1.1.3.3.5 Installing Python . . . 22
2.1.1.3.3.6 Installing NodeJS . . . 23
2.1.1.3.4 Configuring the Network Time Protocol . . . 25
2.1.1.3.5 Configuring the Network Time Protocol Server . . . 25
2.1.1.3.6 Configuring Chrony for RHEL8 Only . . . 26
2.1.1.3.7 Configuring the Server to Boot Without Linux GUI . . . 26
2.1.1.3.8 Configuring the Security Limits . . . 26
2.1.1.3.9 Configuring the Kernel Parameters . . . 26
2.1.1.3.10 Configuring the Firewall . . . 27
2.1.1.3.11 Disabling SELinux . . . 28
2.1.1.3.12 Configuring the `/etc/hosts` File . . . 28
2.1.1.4 Installing the NVIDIA CUDA Driver . . . 28
2.1.1.4.1 Installing the CUDA Driver . . . 29
2.1.1.4.1.1 Installing the CUDA Driver from the Repository . . . 30
2.1.1.4.1.2 Tuning Up NVIDIA Performance . . . 31
2.1.1.4.1.3 Tune Up NVIDIA Performance when Driver Installed from the Repository . . . 31
2.1.1.4.1.4 Tune Up NVIDIA Performance when Driver Installed from the Runfile 32
2.1.1.5 Enabling Core Dumps . . . 32
2.1.1.5.1 Checking the `abrtd` Status . . . 33
2.1.1.5.2 Setting the Limits . . . 33
2.1.1.5.3 Creating the Core Dump Directory . . . 33
2.1.1.5.4 Setting the Output Directory on the `/etc/sysctl.conf` File . . . 33
2.1.1.5.5 Verifying that the Core Dumps Work . . . 34
2.1.1.5.6 Verify Your SQreamDB Installation . . . 34
2.1.1.5.7 Troubleshooting Core Dumping . . . 35
2.1.2 Installing SQream Using Binary Packages . . . 35
2.1.3 Installing Monit . . . 38
2.1.3.1 Getting Started . . . 38
2.1.3.2 Overview . . . 38
2.1.3.2.1 Installing Monit on CentOS: . . . 39
2.1.3.2.2 Installing Monit on CentOS Offline: . . . 39
2.1.3.2.2.1 Building Monit from Source Code . . . 39
2.1.3.2.2.2 Building Monit from Pre-Built Binaries . . . 40
2.1.3.2.3 Installing Monit on Ubuntu: . . . 40
2.1.3.2.4 Installing Monit on Ubuntu Offline: . . . 40
2.1.3.3 Configuring Monit . . . 41
2.1.3.4 Starting Monit . . . 42
2.1.4 Launching SQream with Monit . . . 42
2.1.4.1 Launching SQream . . . 42
2.1.4.2 Monit Usage Examples . . . 45
2.1.4.2.1 Stopping Monit and SQream Separately . . . 45
2.1.4.2.2 Stopping SQream Using a Monit Command . . . 45
2.1.4.2.3 Monit Command Line Options . . . 45
2.1.4.3 Using Monit While Upgrading Your Version of SQream . . . 46
2.2 Installing SQream Studio . . . 47
2.2.1 Installing Prometheus Exporter . . . 47
2.2.1.1 Overview . . . 47
2.2.1.2 Adding a User and Group . . . 47
2.2.1.3 Cloning the Prometheus GIT Project . . . 48

2.2.1.4 Installing the Node Exporter and NVIDIA Exporter . . . 48
2.2.1.5 Installing the Process Exporter . . . 49
2.2.1.6 Opening the Firewall Ports . . . 50
2.2.2 Installing Prometheus Using Binary Packages . . . 50
2.2.2.1 Installing Prometheus . . . 50
2.2.2.2 Configuring Your Prometheus Settings . . . 52
2.2.2.3 Configuring Your Prometheus Service File . . . 53
2.2.2.4 Accessing the Prometheus User Interface . . . 53
2.2.3 Installing the Dashboard Data Collector . . . 54
2.2.3.1 Installing the Dashboard Data Collector . . . 54
2.2.4 Installing Studio on a Stand-Alone Server . . . 56
2.2.4.1 Installing NodeJS Version 12 on the Server . . . 57
2.2.4.2 Installing Studio . . . 58
2.2.4.3 Starting Studio Manually . . . 59
2.2.4.4 Starting Studio as a Service . . . 60
2.2.4.5 Accessing Studio . . . 61
2.2.4.6 Maintaining Studio with the Process Manager (PM2) . . . 61
2.2.4.7 Upgrading Studio . . . 62
2.2.5 Installing an NGINX Proxy Over a Secure Connection . . . 62
2.2.5.1 Overview . . . 63
2.2.5.2 Prerequisites . . . 63
2.2.5.3 Installing NGINX and Adjusting the Firewall . . . 63
2.2.5.4 Creating Your SSL Certificate . . . 65
2.2.5.5 Configuring NGINX to use SSL . . . 66
2.2.5.6 Redirecting Studio Access from HTTP to HTTPS . . . 68
2.2.5.7 Activating Your NGINX Configuration . . . 69
2.2.5.8 Verifying that NGINX is Running . . . 69

3 Upgrade Guides **71**
3.1 Version Upgrade . . . 71
3.2 Upgrade-Related Configuration Changes . . . 74

4 Operational Guides **75**
4.1 Access Control . . . 75
4.1.1 Overview . . . 75
4.1.2 Password Policy . . . 76
4.1.2.1 Password Strength Requirements . . . 76
4.1.2.2 Brute Force Prevention . . . 77
4.1.3 Managing Roles . . . 77
4.1.3.1 Creating New Roles (Users) . . . 77
4.1.3.2 Dropping a User . . . 78
4.1.3.3 Altering a User Name . . . 78
4.1.3.4 Changing a User Password . . . 78
4.1.3.5 Altering Public Role Permissions . . . 78
4.1.3.6 Altering Role Membership (Groups) . . . 79
4.1.4 Permissions . . . 79
4.1.4.1 Syntax . . . 81
4.1.4.1.1 GRANT . . . 81
4.1.4.1.2 REVOKE . . . 83
4.1.4.1.3 Altering Default Permissions . . . 84
4.1.4.2 Examples . . . 85
4.1.4.2.1 GRANT . . . 85
4.1.4.2.2 REVOKE . . . 86
4.1.5 Departmental Example . . . 86

4.1.5.1 Setting up the department permissions . . . 87
4.1.5.2 Creating new users in the departments . . . 88
4.2 Creating or Cloning Storage Clusters . . . 89
4.2.1 Creating a new storage cluster . . . 90
4.2.2 Tell SQream DB to use this storage cluster . . . 90
4.2.2.1 Permanently setting the storage cluster setting . . . 90
4.2.2.2 Start a temporary SQream DB worker with a storage cluster . . . 91
4.2.2.2.1 Using a configuration file (recommended) . . . 91
4.2.2.2.2 Using the command line parameters . . . 91
4.2.3 Copying an existing storage cluster . . . 91
4.3 Working with External Data . . . 91
4.3.1 Inserting Data Using Amazon S3 . . . 92
4.3.1.1 Configuring Amazon S3 . . . 92
4.3.1.2 Setting the S3 URI Format . . . 92
4.3.1.3 Authenticating Users . . . 92
4.3.1.4 Examples . . . 92
4.3.1.4.1 Planning for Data Staging . . . 93
4.3.1.4.2 Creating a Foreign Table . . . 93
4.3.1.4.3 Querying Foreign Tables . . . 94
4.3.1.4.4 Bulk Loading a File from a Public S3 Bucket . . . 94
4.3.1.4.5 Loading Files from an Authenticated S3 Bucket . . . 95
4.3.2 Using SQream in an HDFS Environment . . . 95
4.3.2.1 Configuring an HDFS Environment for the User **sqream** . . . 95
4.3.2.2 Authenticating Hadoop Servers that Require Kerberos . . . 96
4.4 Foreign Tables . . . 98
4.4.1 Supported Data Formats . . . 99
4.4.2 Supported Data Staging . . . 99
4.4.3 Using Foreign Tables . . . 99
4.4.3.1 Planning for Data Staging . . . 99
4.4.3.2 Creating a Foreign Table . . . 99
4.4.3.3 Querying Foreign Tables . . . 100
4.4.3.4 Modifying Data from Staging . . . 100
4.4.3.5 Converting a Foreign Table to a Standard Database Table . . . 101
4.4.4 Error Handling and Limitations . . . 102
4.5 Deleting Data . . . 102
4.5.1 The Deletion Process . . . 102
4.5.2 Usage Notes . . . 103
4.5.2.1 General Notes . . . 103
4.5.2.2 Clean-Up Operations Are I/O Intensive . . . 103
4.5.3 Examples . . . 103
4.5.3.1 Deleting Rows from a Table . . . 104
4.5.3.2 Deleting Values Based on Complex Predicates . . . 105
4.5.3.3 Identifying and Cleaning Up Tables . . . 105
4.5.3.3.1 Triggering a Clean-Up . . . 105
4.5.4 Best Practice . . . 106
4.6 Logging . . . 106
4.6.1 Locating the Log Files . . . 106
4.6.1.1 Log Structure and Contents . . . 107
4.6.1.2 Log-Naming . . . 109
4.6.2 Log Control and Maintenance . . . 109
4.6.2.1 Changing Log Verbosity . . . 109
4.6.2.2 Changing Log Rotation . . . 109
4.6.3 Collecting Logs from Your Cluster . . . 110
4.6.3.1 SQL Syntax . . . 110

4.6.3.2 Command Line Utility . . . 110
4.6.3.3 Parameters . . . 110
4.6.3.4 Example . . . 110
4.6.4 Troubleshooting with Logs . . . 111
4.6.4.1 Loading Logs with Foreign Tables . . . 111
4.6.4.2 Counting Message Types . . . 111
4.6.4.3 Finding Fatal Errors . . . 112
4.6.4.4 Countng Error Events Within a Certain Timeframe . . . 112
4.6.4.5 Tracing Errors to Find Offending Statements . . . 112
4.7 Query Split . . . 113
4.7.1 Syntax . . . 113
4.7.2 Example . . . 115
4.7.2.1 Creating a Sample Table and Query . . . 115
4.7.2.2 Splitting the Query . . . 117
4.7.3 Best Practices . . . 118
4.7.3.1 General . . . 118
4.7.3.2 Choosing a Column to Split by . . . 119
4.7.3.3 Aggregation Best Practices . . . 119
4.7.3.4 Date as Number best practices . . . 119
4.8 Monitoring Query Performance . . . 119
4.8.1 Setting Up the System for Monitoring . . . 120
4.8.1.1 Adjusting the Logging Frequency . . . 120
4.8.1.2 Reading Execution Plans with a Foreign Table . . . 120
4.8.2 Using the `SHOW_NODE_INFO` Command . . . 121
4.8.3 Understanding the Query Execution Plan Output . . . 122
4.8.3.1 Information Presented in the Execution Plan . . . 122
4.8.3.2 Commonly Seen Nodes . . . 122
4.8.4 Examples . . . 123
4.8.4.1 Spooling to Disk . . . 123
4.8.4.1.1 Identifying the Offending Nodes . . . 123
4.8.4.1.2 Common Solutions for Reducing Spool . . . 126
4.8.4.2 Queries with Large Result Sets . . . 126
4.8.4.2.1 Identifying the Offending Nodes . . . 126
4.8.4.2.2 Common Solutions for Reducing Gather Time . . . 128
4.8.4.3 Inefficient Filtering . . . 128
4.8.4.3.1 Identifying the Situation . . . 128
4.8.4.3.2 Common Solutions for Improving Filtering . . . 132
4.8.4.4 Joins with `text` Keys . . . 132
4.8.4.4.1 Identifying the Situation . . . 132
4.8.4.4.2 Improving Query Performance . . . 133
4.8.4.5 Sorting on big `TEXT` fields . . . 134
4.8.4.5.1 Identifying the Situation . . . 134
4.8.4.5.2 Improving Sort Performance on Text Keys . . . 137
4.8.4.6 High Selectivity Data . . . 137
4.8.4.6.1 Identifying the Situation . . . 137
4.8.4.6.2 Improving Performance with High Selectivity Hints . . . 137
4.8.4.7 Performance of unsorted data in joins . . . 138
4.8.4.7.1 Identifying the Situation . . . 138
4.8.4.7.2 Improving Join Performance when Data is Sparse . . . 139
4.8.4.8 Manual Join Reordering . . . 139
4.8.4.8.1 Identifying the situation . . . 139
4.8.4.8.2 Changing the Join Order . . . 139
4.8.5 Further Reading . . . 140
4.9 Security . . . 140

4.9.1 Overview . . . 141
4.9.2 Security best practices for SQream DB . . . 141
4.9.2.1 Secure OS access . . . 141
4.9.2.2 Change the default SUPERUSER . . . 141
4.9.2.3 Create distinct user roles . . . 141
4.9.2.4 Limit SUPERUSER access . . . 142
4.9.2.5 Password strength guidelines . . . 142
4.9.2.6 Use TLS/SSL when possible . . . 142
4.10 Saved Queries . . . 142
4.10.1 Syntax . . . 143
4.10.1.1 Parameter Support . . . 143
4.10.2 Permissions . . . 143
4.10.3 Parameterized Query . . . 144
4.11 Optimization and Best Practices . . . 144
4.11.1 Table design . . . 144
4.11.1.1 Using DATE and DATETIME Data Types . . . 144
4.11.1.2 Avoiding Data flattening and Denormalization . . . 144
4.11.1.3 Converting Foreign Tables to Native Tables . . . 144
4.11.1.4 Leveraging Column Data Information . . . 145
4.11.1.4.1 Appropriately Using NULL and NOT NULL . . . 145
4.11.2 Sorting . . . 145
4.11.3 Query Best Practices . . . 145
4.11.3.1 Reducing Datasets Before Joining Tables . . . 145
4.11.3.2 Using ANSI JOIN . . . 146
4.11.3.3 Using High-Selectivity hint . . . 146
4.11.3.4 Avoiding Aggregation Overflow . . . 147
4.11.3.5 Prefer COUNT(*) and COUNT to Non-nullable Columns . . . 147
4.11.3.6 Returning Only Required Columns . . . 147
4.11.3.7 Reducing Recurring Compilation Time . . . 147
4.11.3.8 Reducing JOIN Complexity . . . 147
4.11.4 Data Loading Considerations . . . 148
4.11.4.1 Using Natural Data Sorting . . . 148

5 Configuration Guides **149**
5.1 Configuring SQream . . . 149
5.1.1 Cluster and Session . . . 149
5.1.1.1 Setting the flags . . . 149
5.1.1.1.1 Syntax . . . 149
5.1.1.1.2 Configuration file . . . 150
5.1.1.2 Flag List . . . 150
5.1.2 Workers . . . 151
5.1.3 Modification Methods . . . 152
5.1.3.1 Worker Configuration File . . . 152
5.1.3.2 Cluster and Session Configuration File . . . 152
5.1.4 Parameter Values . . . 153
5.1.5 Showing All Flags in the Catalog Table . . . 154
5.2 LDAP . . . 154
5.2.1 Configuring SQream roles . . . 154
5.2.2 Configuring LDAP Authentication . . . 155
5.2.2.1 Configuration Methods . . . 155
5.2.2.2 Basic Method . . . 155
5.2.2.2.1 Flag Attributes . . . 155
5.2.2.2.2 Basic Method Configuration . . . 156
5.2.2.2.3 Example . . . 156

5.2.2.3 Advanced Method . . . 157
5.2.2.3.1 Flag Attributes . . . 157
5.2.2.3.2 Advanced Method Configuration . . . 157
5.2.2.3.3 Example . . . 158
5.2.2.4 Disabling LDAP Authentication . . . 159

6 Architecture **161**
6.1 Internals and Architecture . . . 161
6.1.1 Concurrency and Admission Control . . . 162
6.1.2 Statement Compiler . . . 162
6.1.3 Building Blocks (GPU Workers) . . . 162
6.1.4 Storage Layer . . . 163
6.1.4.1 Metadata Layer . . . 163
6.1.4.2 Bulk Data Layer Optimization . . . 163
6.1.5 Transactions . . . 164
6.2 Filesystem and Usage . . . 164
6.2.1 Directory organization . . . 164
6.2.1.1 `databases` . . . 165
6.2.1.2 `metadata` or `rocksdb` . . . 168
6.2.1.3 `temp` . . . 168
6.2.1.4 `logs` . . . 169
6.3 Sizing . . . 169
6.3.1 Concurrency and Scaling in SQreamDB . . . 169
6.3.1.1 Scaling When Data Sizes Grow . . . 170
6.3.1.2 Scaling When Queries Are Queuing . . . 170
6.3.1.3 What To Do When Queries Are Slow . . . 170
6.3.2 Spooling Configuration . . . 171
6.3.2.1 Example . . . 171
6.3.2.1.1 Setting Spool Memory . . . 171

7 Acceleration Studio **175**
7.1 Getting Started with SQream Acceleration Studio . . . 175
7.1.1 Setting Up and Starting Studio . . . 175
7.1.2 Logging In to Studio . . . 175
7.1.3 Navigating Studio's Main Features . . . 175
7.2 Executing Statements and Running Queries from the Editor . . . 176
7.2.1 Executing Statements from the Toolbar . . . 176
7.2.2 Performing Statement-Related Operations from the Database Tree . . . 177
7.2.2.1 Optimizing Database Tables Using the DDL Optimizer . . . 178
7.2.2.2 Executing Pre-Defined Queries from the System Queries Panel . . . 178
7.2.3 Writing Statements and Queries from the Statement Panel . . . 178
7.2.4 Viewing Statement and Query Results from the Results Panel . . . 179
7.2.4.1 Searching Query Results in the Results View . . . 179
7.2.4.1.1 Saving Results to the Clipboard . . . 180
7.2.4.1.2 Saving Results to a Local File . . . 180
7.2.4.1.3 Running Parallel Statements . . . 180
7.2.4.2 Execution Details View . . . 180
7.2.4.2.1 Viewing Query Statistics . . . 182
7.2.4.2.2 Using the Plain View . . . 183
7.2.4.3 Viewing Wrapped Strings in the SQL View . . . 183
7.3 Viewing Logs . . . 183
7.3.1 Filtering Table Data . . . 184
7.3.2 Viewing Query Logs . . . 184
7.3.3 Viewing Session Logs . . . 184

7.3.4 Viewing System Logs . . . 185
7.3.5 Viewing All Log Lines . . . 185
7.4 Creating, Assigning, and Managing Roles and Permissions . . . 186
7.4.1 Viewing Information About a Role . . . 186
7.4.2 Creating a New Role . . . 186
7.4.3 Editing a Role . . . 187
7.4.4 Deleting a Role . . . 187
7.5 Configuring Your Instance of SQreams . . . 187
7.5.1 Editing Your Parameters . . . 188
7.5.2 Exporting and Importing Configuration Files . . . 188

8 Connecting to SQreamDB 189
8.1 Client Platforms . . . 189
8.1.1 Data Integration Tools . . . 189
8.1.2 Business Intelligence (BI) Tools . . . 190
8.1.3 Data Analysis and Programming Languages . . . 190
8.1.3.1 Denodo Platform . . . 191
8.1.3.1.1 Before You Begin . . . 191
8.1.3.1.2 Setting Up a Connection to SQreamDB . . . 192
8.1.3.1.3 Limitation . . . 195
8.1.3.2 Informatica Cloud Services . . . 196
8.1.3.2.1 Overview . . . 196
8.1.3.2.1.1 Establishing a Connection between SQream and Informatica . . . 198
8.1.3.2.1.2 Establishing a Connection In Your Environment . . . 205
8.1.3.2.1.3 Establishing an ODBC DSN Connection In Your Environment . . . 207
8.1.3.2.1.4 Establishing a JDBC Connection In Your Environment . . . 209
8.1.3.2.1.5 Supported SQream Driver Versions . . . 212
8.1.3.3 MicroStrategy . . . 213
8.1.3.3.1 Overview . . . 213
8.1.3.3.1.1 What is MicroStrategy? . . . 214
8.1.3.3.1.2 Connecting a Data Source . . . 216
8.1.3.3.1.3 Supported SQream Drivers . . . 228
8.1.3.4 Pentaho Data Integration . . . 229
8.1.3.4.1 Overview . . . 229
8.1.3.4.1.1 Installing Pentaho . . . 230
8.1.3.4.1.2 Installing and Setting Up the JDBC Driver . . . 232
8.1.3.4.1.3 Creating a Transformation . . . 233
8.1.3.4.1.4 Defining Your Output . . . 239
8.1.3.4.1.5 Importing Data . . . 244
8.1.3.5 BI Desktop . . . 250
8.1.3.5.1 Prerequisites . . . 254
8.1.3.5.2 Installing Power BI Desktop . . . 255
8.1.3.5.3 Best Practices for Power BI . . . 260
8.1.3.6 R . . . 262
8.1.3.6.1 JDBC . . . 263
8.1.3.6.1.1 A full example . . . 269
8.1.3.6.2 ODBC . . . 273
8.1.3.6.2.1 A full example . . . 278
8.1.3.7 SAP BusinessObjects . . . 281
8.1.3.7.1 Overview . . . 282
8.1.3.7.2 Establishing a New Connection Using a Generic JDCB Connector . . . 284
8.1.3.8 SAS Viya . . . 289
8.1.3.8.1 Installing SAS Viya . . . 290
8.1.3.8.1.1 Downloading SAS Viya . . . 291

8.1.3.8.1.2 Installing the JDBC Driver . . . 291
8.1.3.8.2 Configuring SAS Viya . . . 293
8.1.3.8.3 Operating SAS Viya . . . 296
8.1.3.8.3.1 Using SAS Viya Visual Analytics . . . 297
8.1.3.8.4 Troubleshooting SAS Viya . . . 300
8.1.3.8.4.1 Inserting Only Required Data . . . 301
8.1.3.8.4.2 Creating a Separate Service for SAS Viya . . . 303
8.1.3.8.4.3 Locating the SQreamDB JDBC Driver . . . 303
8.1.3.8.4.4 Supporting TEXT . . . 306
8.1.3.9 SQL Workbench . . . 306
8.1.3.9.1 Installing SQL Workbench with the SQream Installer . . . 309
8.1.3.9.2 Installing SQL Workbench Manually . . . 314
8.1.3.9.2.1 Install Java Runtime . . . 314
8.1.3.9.2.2 Get the SQream DB JDBC Driver . . . 316
8.1.3.9.2.3 Install SQL Workbench . . . 317
8.1.3.9.2.4 Setting up the SQream DB JDBC Driver Profile . . . 318
8.1.3.9.3 Create a New Connection Profile for Your Cluster . . . 324
8.1.3.9.4 Suggested Optional Configuration . . . 326
8.1.3.10 Tableau . . . 328
8.1.3.10.1 Prerequisites . . . 330
8.1.3.10.2 Setting Up JDBC . . . 330
8.1.3.10.3 Installing the Tableau Connector . . . 332
8.1.3.10.4 Connecting to SQream . . . 333
8.1.3.11 Talend . . . 336
8.1.3.11.1 Overview . . . 336
8.1.3.11.1.1 Creating a New Metadata JDBC DB Connection . . . 337
8.1.3.11.1.2 Supported SQream Drivers . . . 345
8.1.3.11.1.3 Supported Data Sources . . . 346
8.1.3.11.1.4 Known Issues . . . 348
8.1.3.12 TIBCO Spotfire . . . 349
8.1.3.12.1 Overview . . . 349
8.1.3.12.1.1 Establishing a Connection between TIBCO Spotfire and SQream . . . 350
8.1.3.12.1.2 Creating a JDBC Connection . . . 352
8.1.3.12.1.3 Creating an ODBC Connection . . . 353
8.1.3.12.1.4 Creating the SQream Data Source Template . . . 359
8.1.3.12.1.5 Creating a Data Source . . . 372
8.1.3.12.1.6 Creating an Information Link . . . 376
8.1.3.12.1.7 Troubleshooting . . . 393
8.1.3.12.1.8 The JDBC Driver does not Support Boolean, Decimal, or Numeric Types . . . 394
8.1.3.12.1.9 Information Services do not Support Live Queries . . . 396
8.2 Client Drivers . . . 397
8.2.1 Client Driver Downloads . . . 398
8.2.1.1 JDBC . . . 398
8.2.1.1.1 Installing the JDBC Driver . . . 399
8.2.1.1.1.1 Prerequisites . . . 400
8.2.1.1.1.2 Getting the JAR file . . . 401
8.2.1.1.1.3 Setting Up the Class Path . . . 402
8.2.1.1.2 Connecting to SQream Using a JDBC Application . . . 404
8.2.1.1.2.1 Driver Class . . . 405
8.2.1.1.2.2 Connection String . . . 405
8.2.1.1.2.3 Connection Parameters . . . 406
8.2.1.1.2.4 Connection String Examples . . . 407
8.2.1.1.2.5 Java Program Sample . . . 410

8.2.1.1.3 Prepared Statements . . . 423
8.2.1.1.3.1 Prepared Statement Sample . . . 424
8.2.1.2 Python (pysqream) . . . 425
8.2.1.2.1 Installing the Python Connector . . . 426
8.2.1.2.1.1 Prerequisites . . . 426
8.2.1.2.1.2 Python . . . 427
8.2.1.2.1.3 PIP . . . 428
8.2.1.2.1.4 OpenSSL for Linux . . . 432
8.2.1.2.1.5 Installing via PIP with an internet connection . . . 434
8.2.1.2.1.6 Installing via PIP without an internet connection . . . 436
8.2.1.2.1.7 Upgrading an Existing Installation . . . 438
8.2.1.2.2 SQLAlchemy . . . 439
8.2.1.2.2.1 Limitations . . . 440
8.2.1.2.2.2 Creating a Standard Connection . . . 441
8.2.1.2.2.3 Pulling a Table into Pandas . . . 445
8.2.1.2.3 API . . . 449
8.2.1.2.3.1 Using the Cursor . . . 450
8.2.1.2.3.2 Reading Result Metadata . . . 460
8.2.1.2.3.3 Loading Data into a Table . . . 466
8.2.1.2.3.4 Using SQLAlchemy ORM to Create and Populate Tables . . . 474
8.2.1.3 Node.JS . . . 480
8.2.1.3.1 Installing the Node.JS driver . . . 483
8.2.1.3.1.1 Prerequisites . . . 483
8.2.1.3.1.2 Install with NPM . . . 483
8.2.1.3.1.3 Install from an offline package . . . 484
8.2.1.3.2 Connect to SQream DB with a Node.JS application . . . 485
8.2.1.3.2.1 Create a simple test . . . 485
8.2.1.3.2.2 Run the test . . . 488
8.2.1.3.3 API reference . . . 489
8.2.1.3.3.1 Connection parameters . . . 489
8.2.1.3.3.2 Events . . . 490
8.2.1.3.3.3 Example . . . 491
8.2.1.3.3.4 Input placeholders . . . 493
8.2.1.3.4 Examples . . . 494
8.2.1.3.4.1 Setting configuration flags . . . 494
8.2.1.3.4.2 Lazyloading . . . 496
8.2.1.3.4.3 Reusing a connection . . . 500
8.2.1.3.4.4 Using placeholders in queries . . . 505
8.2.1.3.5 Troubleshooting and recommended configuration . . . 508
8.2.1.3.5.1 Preventing `heap out of memory` errors . . . 508
8.2.1.3.5.2 BIGINT support . . . 510
8.2.1.4 ODBC . . . 513
8.2.1.4.1 Install and Configure ODBC on Windows . . . 513
8.2.1.4.1.1 Installing the ODBC Driver . . . 515
8.2.1.4.1.2 Prerequisites . . . 515
8.2.1.4.1.3 Visual Studio 2015 Redistributables . . . 515
8.2.1.4.1.4 Administrator Privileges . . . 516
8.2.1.4.1.5 Running the Windows Installer . . . 516
8.2.1.4.1.6 Selecting Components . . . 517
8.2.1.4.1.7 Configuring the ODBC Driver DSN . . . 519
8.2.1.4.1.8 Connection Parameters . . . 524
8.2.1.4.1.9 Troubleshooting . . . 525
8.2.1.4.1.10 Solving “Code 126” ODBC errors . . . 525
8.2.1.4.1.11 Limitations . . . 526

8.2.1.4.2 Install and configure ODBC on Linux . . . 527
8.2.1.4.2.1 Prerequisites . . . 529
8.2.1.4.2.2 unixODBC . . . 529
8.2.1.4.2.3 Install unixODBC on RHEL 7 / CentOS 7 . . . 532
8.2.1.4.2.4 Install unixODBC on Ubuntu . . . 532
8.2.1.4.2.5 Install the ODBC driver with a script . . . 532
8.2.1.4.2.6 Install the ODBC driver manually . . . 535
8.2.1.4.2.7 Install the driver dependencies . . . 542
8.2.1.4.2.8 Testing the connection . . . 543
8.2.1.4.2.9 ODBC DSN Parameters . . . 546
8.2.1.4.2.10 Limitations . . . 547
8.2.1.4.3 Getting the ODBC driver . . . 548
8.2.1.4.4 Install and configure the ODBC driver . . . 550
8.2.1.5 SQreamNET . . . 550
8.2.1.5.1 Before You Begin . . . 551
8.2.1.5.2 Integrating SQreamNET . . . 552
8.2.1.5.3 Connecting to SQream For the First Time . . . 553
8.2.1.5.3.1 Connection String Syntax . . . 554
8.2.1.5.3.2 Connection Parameters . . . 555
8.2.1.5.3.3 Connection String Examples . . . 555
8.2.1.5.3.4 Sample C# Program . . . 557
8.2.1.5.4 Limitations . . . 585

9 Data Ingestion Sources 589
9.1 Overview . . . 589
9.1.1 Getting Started . . . 589
9.1.2 Data Loading Considerations . . . 590
9.1.2.1 Verifying Data and Performance after Loading . . . 590
9.1.2.2 File Soure Location when Loading . . . 590
9.1.2.3 Supported Load Methods . . . 591
9.1.2.4 Unsupported Data Types . . . 591
9.1.2.5 Handing Extended Errors . . . 591
9.1.2.6 Best Practices for CSV . . . 591
9.1.2.7 Best Practices for Parquet . . . 592
9.1.2.7.1 Supported Types and Behavior Notes . . . 592
9.1.2.8 Best Practices for ORC . . . 593
9.1.2.8.1 Type Support and Behavior Notes . . . 593
9.1.3 Further Reading and Migration Guides . . . 594
9.2 Avro . . . 594
9.2.1 Foreign Data Wrapper Prerequisites . . . 594
9.2.2 Making Avro Files Accessible to Workers . . . 595
9.2.3 Preparing Your Table . . . 595
9.2.3.1 Creating a Table . . . 595
9.2.3.2 Creating a Foreign Table . . . 596
9.2.4 Mapping Between SQream and Avro Data Types . . . 598
9.2.4.1 Primitive Data Types . . . 598
9.2.4.2 Complex Data Types . . . 598
9.2.4.3 Logical Data Types . . . 599
9.2.5 Mapping Objects to Rows . . . 599
9.2.6 Ingesting Data into SQream . . . 599
9.2.6.1 Syntax . . . 599
9.2.6.2 Example . . . 600
9.2.7 Parameters . . . 600
9.2.8 Best Practices . . . 600

9.2.9 Additional Examples . . . 601
9.2.9.1 Omitting Unsupported Column Types . . . 601
9.2.9.2 Modifying Data Before Loading . . . 601
9.2.9.3 Loading a Table from a Directory of Avro Files on HDFS . . . 602
9.2.9.4 Loading a Table from a Directory of Avro Files on S3 . . . 602
9.3 CSV . . . 602
9.3.1 Foreign Data Wrapper Prerequisites . . . 603
9.3.2 Prepare CSVs . . . 603
9.3.3 Place CSVs where SQream DB workers can access . . . 604
9.3.4 Figure out the table structure . . . 604
9.3.5 Bulk load the data with COPY FROM . . . 605
9.3.6 Loading different types of CSV files . . . 605
9.3.6.1 Loading a standard CSV file from a local filesystem . . . 605
9.3.6.2 Loading a PSV (pipe separated value) file . . . 605
9.3.6.3 Loading a TSV (tab separated value) file . . . 605
9.3.6.4 Loading a text file with non-printable delimiter . . . 606
9.3.6.5 Loading a text file with multi-character delimiters . . . 606
9.3.6.6 Loading files with a header row . . . 606
9.3.6.7 Loading files formatted for Windows (`\r\n`) . . . 606
9.3.6.8 Loading a file from a public S3 bucket . . . 606
9.3.6.9 Loading files from an authenticated S3 bucket . . . 606
9.3.6.10 Loading files from an HDFS storage . . . 607
9.3.6.11 Saving rejected rows to a file . . . 607
9.3.6.12 Stopping the load if a certain amount of rows were rejected . . . 607
9.3.6.13 Load CSV files from a set of directories . . . 607
9.3.6.14 Rearrange destination columns . . . 607
9.3.6.15 Loading non-standard dates . . . 608
9.4 Parquet . . . 608
9.4.1 Foreign Data Wrapper Prerequisites . . . 608
9.4.2 Preparing Your Parquet Files . . . 609
9.4.3 Making Parquet Files Accessible to Workers . . . 609
9.4.4 Creating a Table . . . 610
9.4.5 Ingesting Data into SQream . . . 611
9.4.5.1 Syntax . . . 611
9.4.5.2 Examples . . . 611
9.4.5.2.1 Omitting Unsupported Column Types . . . 611
9.4.5.2.2 Modifying Data Before Loading . . . 611
9.4.5.2.3 Loading a Table from a Directory of Parquet Files on HDFS . . . 612
9.4.5.2.4 Loading a Table from a Directory of Parquet Files on S3 . . . 612
9.4.6 Best Practices . . . 612
9.5 ORC . . . 613
9.5.1 Foreign Data Wrapper Prerequisites . . . 613
9.5.2 Prepare the files . . . 614
9.5.3 Place ORC files where SQream DB workers can access them . . . 615
9.5.4 Figure out the table structure . . . 615
9.5.5 Verify table contents . . . 616
9.5.6 Copying data into SQream DB . . . 617
9.5.6.1 Working around unsupported column types . . . 617
9.5.6.2 Modifying data during the copy process . . . 617
9.5.7 Further ORC loading examples . . . 617
9.5.7.1 Loading a table from a directory of ORC files on HDFS . . . 618
9.5.7.2 Loading a table from a bucket of files on S3 . . . 618
9.6 JSON . . . 618
9.6.1 Foreign Data Wrapper Prerequisites . . . 619

9.6.2 Making JSON Files Accessible to Workers . . . 619
9.6.3 Mapping between JSON and SQream . . . 619
9.6.3.1 Character Escaping . . . 620
9.6.4 Ingesting JSON Data into SQream . . . 620
9.6.4.1 Syntax . . . 620
9.6.4.2 Parameters . . . 620
9.6.4.3 Automatic Schema Inference . . . 622
9.6.4.4 Examples . . . 622
9.7 External Databases . . . 623
9.7.1 Before You Begin . . . 623
9.7.1.1 Minimum Hardware Requirements . . . 623
9.7.1.2 Sizing Guidelines . . . 624
9.7.2 Installation and Connection . . . 624
9.7.2.1 Getting the SQLoader Configuration and JAR Files . . . 624
9.7.2.2 Establishing a Connection . . . 624
9.7.2.3 Starting SQLoader . . . 626
9.7.3 Loading Data into SQreamDB . . . 626
9.7.3.1 Customizing Data Load . . . 626
9.7.3.2 Data Types and Mapping . . . 633
9.7.3.2.1 Automatic Mapping . . . 633
9.7.3.2.1.1 Oracle . . . 633
9.7.3.2.1.2 Postgresql . . . 634
9.7.3.2.1.3 Teradata . . . 634
9.7.3.2.1.4 Microsoft SQL Server . . . 634
9.7.3.2.1.5 SAP HANA . . . 635
9.7.3.2.2 Manual Mapping . . . 635
9.7.3.2.2.1 `names` Method . . . 635
9.7.4 Creating Summary Tables . . . 637
9.7.4.1 Examples . . . 637
9.7.4.1.1 A Summary Table . . . 637
9.7.4.2 Change Data Capture Summary Tables . . . 638
9.7.5 CLI Examples . . . 638

10 External Storage Platforms **641**
10.1 Azure Blob Storage . . . 641
10.1.1 ABS Bucket File Location . . . 641
10.1.2 Connection String . . . 641
10.1.3 Examples . . . 642
10.2 Google Cloud Platform . . . 642
10.2.1 GCP Bucket File Location . . . 642
10.2.2 GCP Access . . . 642
10.2.2.1 Before You Begin . . . 642
10.2.2.2 Granting GCP Access . . . 642
10.2.3 Examples . . . 643
10.3 HDFS Environment . . . 643
10.3.1 Configuring an HDFS Environment for the User **sqream** . . . 643
10.3.2 Authenticating Hadoop Servers that Require Kerberos . . . 644
10.4 Amazon Web Services . . . 646
10.4.1 S3 Bucket File Location . . . 646
10.4.2 S3 Access . . . 647
10.4.3 Authentication . . . 647
10.4.4 Connecting to S3 Using SQreamDB Legacy Configuration File . . . 647
10.4.5 Examples . . . 647
10.4.5.1 Creating a Foreign Table . . . 647

10.4.5.2 Querying Foreign Tables . . . 648
10.4.5.3 Bulk Loading a File from a Public S3 Bucket . . . 649
10.4.5.4 Loading Files from an Authenticated S3 Bucket . . . 649

11 Feature Guides **651**
11.1 Automatic Foreign Table DDL Resolution . . . 651
11.1.1 Usage Notes . . . 651
11.1.2 Syntax . . . 652
11.1.3 Example . . . 652
11.1.4 Permissions . . . 652
11.2 Query Healer . . . 652
11.2.1 Configuration . . . 652
11.2.2 Query Log . . . 652
11.2.3 Activating a Graceful Shutdown . . . 653
11.3 Compression . . . 653
11.3.1 Encoding . . . 653
11.3.2 Lossless Compression . . . 653
11.3.2.1 Automatic Compression . . . 654
11.3.2.2 Compression Methods . . . 654
11.3.2.3 Specifying Compression Strategies . . . 655
11.3.2.3.1 Explicitly Specifying Automatic Compression . . . 655
11.3.2.3.2 Forcing No Compression . . . 655
11.3.2.3.3 Forcing Compression . . . 655
11.3.2.4 Examining Compression Effectiveness . . . 656
11.3.2.4.1 Querying the Catalog . . . 656
11.3.2.4.2 Example Subset from “Ontime” Table . . . 657
11.3.2.4.3 Notes on Reading the “Ontime” Table . . . 659
11.3.3 Best Practices . . . 660
11.3.3.1 Letting SQream Determine the Best Compression Strategy . . . 660
11.3.3.2 Maximizing the Advantage of Each Compression Scheme . . . 660
11.3.3.3 Choosing Data Types that Fit Your Data . . . 660
11.4 Python User-Defined Functions . . . 660
11.4.1 Before You Begin . . . 661
11.4.2 SQreamDB’s UDF Support . . . 661
11.4.2.1 Scalar Functions . . . 661
11.4.2.2 Python . . . 661
11.4.2.3 Using Modules . . . 662
11.4.3 Working with Existing UDFs . . . 662
11.4.3.1 Finding Existing UDFs in the Catalog . . . 662
11.4.3.2 Getting Function DDL . . . 662
11.4.3.3 Handling Errors . . . 662
11.4.4 Permissions and Sharing . . . 662
11.4.5 Example . . . 663
11.4.6 Best Practices . . . 664
11.5 Workload Manager . . . 664
11.5.1 Setting Up Service Queues . . . 664
11.5.2 Example - Allocating ETL Resources . . . 664
11.5.2.1 Creating the Configuration . . . 665
11.5.2.2 Verifying the Configuration . . . 665
11.5.3 Configuring a Client Connection to a Specific Service . . . 666
11.5.3.1 Using SQream Studio . . . 666
11.5.3.2 Using the SQream SQL CLI Reference . . . 666
11.5.3.3 Using a JDBC Client Driver . . . 667
11.5.3.4 Using an ODBC Client Driver . . . 667

11.5.3.5 Using a Python Client Driver . . . 667
11.5.3.6 Using a Node.js Client Driver . . . 668
11.6 Concurrency and Locks . . . 668
11.6.1 Locking Modes . . . 668
11.6.2 When are Locks Obtained? . . . 669
11.6.3 Monitoring Locks . . . 669

12 References **671**
12.1 SQL Statements and Syntax . . . 671
12.1.1 SQL Syntax Features . . . 671
12.1.2 SQL Statements . . . 672
12.1.2.1 Data Definition Commands (DDL) . . . 672
12.1.2.2 Data Manipulation Commands (DML) . . . 673
12.1.2.3 Utility Commands . . . 673
12.1.2.4 Workload Management . . . 675
12.1.2.5 Access Control Commands . . . 675
12.1.3 SQL Functions . . . 675
12.1.3.1 Summary of Functions . . . 675
12.1.3.1.1 Built-In Scalar Functions . . . 676
12.1.3.1.1.1 Bitwise Operations . . . 676
12.1.3.1.1.2 Conditionals . . . 676
12.1.3.1.1.3 Conversion . . . 676
12.1.3.1.1.4 Date and Time . . . 677
12.1.3.1.1.5 Numeric . . . 677
12.1.3.1.1.6 Strings . . . 678
12.1.3.1.2 User-Defined Scalar Functions . . . 679
12.1.3.1.3 Aggregate Functions . . . 679
12.1.3.1.4 Window Functions . . . 680
12.1.3.1.5 Workload Management Functions . . . 680
12.1.3.1.5.1 Built-In Scalar Functions . . . 681
12.1.3.1.5.2 User-Defined Functions . . . 681
12.1.3.1.5.3 Aggregate Functions . . . 681
12.1.3.1.5.4 Overview . . . 681
12.1.3.1.5.5 Available Aggregate Functions . . . 681
12.1.3.1.5.6 Window Functions . . . 682
12.2 Catalog Reference . . . 682
12.2.1 What Information Does the Schema Contain? . . . 682
12.2.2 How to Get Table Information? . . . 682
12.2.2.1 Database Management Tables . . . 683
12.2.2.2 Data Storage and Organization Tables . . . 683
12.2.2.2.1 Catalog Tables . . . 683
12.2.2.2.1.1 Clustering Keys . . . 684
12.2.2.2.1.2 Columns . . . 684
12.2.2.2.1.3 Columns . . . 684
12.2.2.2.1.4 External Table Columns . . . 685
12.2.2.2.1.5 Databases . . . 685
12.2.2.2.1.6 Parameters . . . 686
12.2.2.2.1.7 Permissions . . . 686
12.2.2.2.1.8 Permission Types . . . 686
12.2.2.2.1.9 Default Permissions . . . 686
12.2.2.2.1.10 Default Table Permissions . . . 687
12.2.2.2.1.11 Default Schema Permissions . . . 687
12.2.2.2.1.12 Table Permissions . . . 687
12.2.2.2.1.13 Database Permissions . . . 688

12.2.2.2.1.14 Schema Permissions . . . 688
12.2.2.2.1.15 Queries . . . 688
12.2.2.2.1.16 Roles . . . 688
12.2.2.2.1.17 Roles . . . 689
12.2.2.2.1.18 Role Memberships . . . 689
12.2.2.2.1.19 Schemas . . . 689
12.2.2.2.1.20 Tables . . . 689
12.2.2.2.1.21 Tables . . . 690
12.2.2.2.1.22 Foreign Tables . . . 690
12.2.2.2.1.23 Views . . . 690
12.2.2.2.1.24 User Defined Functions . . . 691
12.2.2.2.2 Additional Tables . . . 691
12.2.2.2.2.1 Extents . . . 691
12.2.2.2.2.2 Chunk Columns . . . 692
12.2.2.2.2.3 Chunks . . . 692
12.2.2.2.2.4 Delete Predicates . . . 693
12.2.2.2.3 Examples . . . 693
12.2.2.2.3.1 Listing All Tables in a Database . . . 693
12.2.2.2.3.2 Listing All Schemas in a Database . . . 694
12.2.2.2.3.3 Listing Columns and Their Types for a Specific Table . . . 694
12.2.2.2.3.4 Listing Delete Predicates . . . 694
12.2.2.2.3.5 Listing Saved Queries . . . 694
12.3 Command line programs . . . 694
12.3.1 metadata_server . . . 695
12.3.1.1 Command Line Arguments . . . 695
12.3.1.2 Starting metadata server . . . 695
12.3.1.2.1 Starting temporarily . . . 695
12.3.1.2.2 Starting temporarily with non-default port . . . 695
12.3.1.2.3 Stopping metadata server . . . 696
12.3.2 sqreamd . . . 696
12.3.2.1 Starting SQream DB . . . 696
12.3.2.1.1 Start SQream DB temporarily . . . 696
12.3.2.2 Command line arguments . . . 696
12.3.2.2.1 Positional command arguments . . . 697
12.3.3 sqream Linux console . . . 697
12.3.3.1 Starting the console . . . 697
12.3.3.2 Operations and flag reference . . . 698
12.3.3.2.1 Commands . . . 698
12.3.3.2.2 Master . . . 698
12.3.3.2.2.1 Syntax . . . 699
12.3.3.2.2.2 Common usage . . . 699
12.3.3.2.2.3 Start master node . . . 699
12.3.3.2.2.4 Start master node on different ports . . . 699
12.3.3.2.2.5 Listing active master nodes and workers . . . 699
12.3.3.2.2.6 Stopping all SQream DB workers and master . . . 699
12.3.3.2.3 Workers . . . 700
12.3.3.2.3.1 Syntax . . . 700
12.3.3.2.3.2 Common usage . . . 700
12.3.3.2.3.3 Start 2 workers . . . 700
12.3.3.2.3.4 Stop a single worker . . . 701
12.3.3.2.3.5 Start workers with a different spool size . . . 701
12.3.3.2.3.6 Starting multiple workers on non-dedicated GPUs . . . 701
12.3.3.2.3.7 Overriding default configuration files . . . 701
12.3.3.2.4 Client . . . 702

12.3.3.2.4.1 Syntax . . . 702
12.3.3.2.4.2 Common usage . . . 702
12.3.3.2.4.3 Start a client . . . 702
12.3.3.2.4.4 Start a client to a specific worker . . . 702
12.3.3.2.4.5 Start master node on different ports . . . 703
12.3.3.2.4.6 Listing active master nodes and worker nodes . . . 703
12.3.3.2.5 Editor . . . 703
12.3.3.2.5.1 Syntax . . . 703
12.3.3.2.5.2 Common usage . . . 703
12.3.3.2.5.3 Start the editor UI . . . 703
12.3.3.2.5.4 Stop the editor UI . . . 703
12.3.3.3 Using the console to start SQream DB . . . 704
12.3.3.3.1 Starting a SQream DB cluster for the first time . . . 704
12.3.4 Server Picker . . . 704
12.3.4.1 Command Line Arguments . . . 704
12.3.4.1.1 Parameters . . . 704
12.3.4.1.2 Example . . . 705
12.3.4.2 Starting server picker . . . 705
12.3.4.2.1 Starting temporarily . . . 705
12.3.4.2.2 Starting temporarily with non-default port . . . 705
12.3.4.2.3 Stopping server picker . . . 705
12.3.5 SqreamStorage . . . 705
12.3.5.1 Running SqreamStorage . . . 706
12.3.5.2 Command Line Arguments . . . 706
12.3.5.3 Example . . . 706
12.3.6 Sqream SQL CLI Reference . . . 706
12.3.6.1 Installing Sqream SQL . . . 707
12.3.6.1.1 Troubleshooting Sqream SQL Installation . . . 707
12.3.6.2 Using Sqream SQL . . . 708
12.3.6.2.1 Running Commands Interactively (SQL shell) . . . 708
12.3.6.2.2 Executing Batch Scripts (`-f`) . . . 709
12.3.6.2.3 Executing Commands Immediately (`-c`) . . . 709
12.3.6.3 Examples . . . 710
12.3.6.3.1 Starting a Regular Interactive Shell . . . 710
12.3.6.3.2 Executing Statements in an Interactive Shell . . . 710
12.3.6.3.3 Executing SQL Statements from the Command Line . . . 711
12.3.6.3.4 Controlling the Client Output . . . 711
12.3.6.3.4.1 Exporting SQL Query Results to CSV . . . 711
12.3.6.3.4.2 Changing a CSV to a TSV . . . 711
12.3.6.3.5 Executing a Series of Statements From a File . . . 712
12.3.6.3.6 Connecting Using Environment Variables . . . 712
12.3.6.3.7 Connecting to a Specific Queue . . . 712
12.3.6.4 Operations and Flag References . . . 712
12.3.6.4.1 Command Line Arguments . . . 712
12.3.6.4.1.1 Supported Record Delimiters . . . 713
12.3.6.4.2 Meta-Commands . . . 713
12.3.6.4.3 Basic Commands . . . 714
12.3.6.4.4 Moving Around the Command Line . . . 714
12.3.6.4.5 Searching . . . 714
12.3.7 upgrade_storage . . . 715
12.3.7.1 Running upgrade_storage . . . 715
12.3.7.1.1 Command line arguments and options . . . 715
12.3.7.1.2 Syntax . . . 715
12.3.7.2 Results and error codes . . . 715

12.3.7.3 Examples . . . 715
12.3.7.3.1 Upgrade SQream DB's storage cluster . . . 715
12.4 SQL Feature Checklist . . . 716
12.4.1 Data Types and Values . . . 717
12.4.2 Constraints . . . 717
12.4.3 Transactions . . . 717
12.4.4 Indexes . . . 717
12.4.5 Schema Changes . . . 718
12.4.6 Statements . . . 718
12.4.7 Clauses . . . 718
12.4.8 Table Expressions . . . 719
12.4.9 Scalar Expressions . . . 719
12.4.10 Permissions . . . 719
12.4.11 Extra Functionality . . . 720

13 Data Types **721**
13.1 Supported Data Types . . . 721
13.1.1 Primitive Data Types . . . 721
13.1.2 Array . . . 723
13.1.2.1 Limitation . . . 723
13.1.2.1.1 Syntax . . . 723
13.1.2.1.2 Supported Operators . . . 725
13.1.2.1.3 Examples . . . 726
13.1.2.2 `ARRAY` Statements . . . 726
13.1.2.3 Ingesting Arrayed Data from External Files . . . 726
13.1.2.3.1 Limitations . . . 727
13.1.2.4 Connectors . . . 727
13.1.2.4.1 Limitations . . . 727
13.1.2.5 Casting . . . 728
13.1.2.6 Function . . . 728
13.2 Casts and Conversions . . . 728
13.2.1 Supported Casts . . . 729
13.3 Supported Casts . . . 729
13.3.1 Numeric . . . 729
13.3.1.1 Numeric Examples . . . 729
13.3.2 Boolean . . . 730
13.3.2.1 Boolean Examples . . . 730
13.3.2.2 Boolean Casts and Conversions . . . 730
13.3.3 Integer . . . 730
13.3.3.1 Integer Types . . . 731
13.3.3.2 Integer Examples . . . 731
13.3.3.3 Integer Casts and Conversions . . . 731
13.3.4 Floating Point . . . 732
13.3.4.1 Floating Point Types . . . 732
13.3.4.2 Floating Point Examples . . . 732
13.3.4.3 Floating Point Casts and Conversions . . . 733
13.3.5 String . . . 733
13.3.5.1 Length . . . 733
13.3.5.2 Syntax . . . 733
13.3.5.3 Size . . . 733
13.3.5.4 String Examples . . . 733
13.3.5.5 String Casts and Conversions . . . 734
13.3.6 Date . . . 734
13.3.6.1 Date Types . . . 734

13.3.6.2 Aliases . . . 735
13.3.6.3 Syntax . . . 735
13.3.6.4 Size . . . 735
13.3.6.5 Date Examples . . . 735
13.3.6.6 Date Casts and Conversions . . . 736

14 Release Notes **737**
14.1 4.0 Release Notes . . . 738
14.1.1 Release Notes 4.0 . . . 738
14.1.1.1 New Features . . . 738
14.1.1.2 Storage Version . . . 738
14.1.1.3 SQream Studio Updates and Improvements . . . 738
14.1.1.4 Known Issues . . . 739
14.1.1.5 Version 4.0 resolved Issues . . . 739
14.1.1.6 Configuration Changes . . . 739
14.1.1.7 Naming Changes . . . 739
14.1.1.8 Deprecated Features . . . 739
14.1.1.9 End of Support . . . 739
14.1.1.10 Upgrading to version 4.0 . . . 740
14.1.2 Release Notes 4.1 . . . 740
14.1.2.1 New Features . . . 741
14.1.2.2 Newly Released Connector Drivers . . . 741
14.1.2.3 Storage Version . . . 741
14.1.2.4 SQream Studio Updates and Improvements . . . 741
14.1.2.5 Known Issues . . . 741
14.1.2.6 Version 4.1 resolved Issues . . . 741
14.1.2.7 Configuration Changes . . . 742
14.1.2.8 Naming Changes . . . 742
14.1.2.9 Deprecated Features . . . 742
14.1.2.10 End of Support . . . 742
14.1.2.11 Upgrading to v4.1 . . . 742
14.1.3 Release Notes 4.2 . . . 743
14.1.3.1 New Features . . . 743
14.1.3.2 Newly Released Connector Drivers . . . 744
14.1.3.3 Compatibility Matrix . . . 744
14.1.3.4 SQream Studio Updates and Improvements . . . 744
14.1.3.5 Known Issues . . . 745
14.1.3.6 Version 4.2 Resolved Issues . . . 745
14.1.3.7 Configuration Changes . . . 745
14.1.3.8 Naming Changes . . . 745
14.1.3.9 Deprecated Features . . . 745
14.1.3.10 End of Support . . . 746
14.1.3.11 Upgrading to v4.2 . . . 746
14.1.4 Release Notes 4.3 . . . 746
14.1.4.1 Compatibility Matrix . . . 747
14.1.4.2 New Features and Enhancements . . . 747
14.1.4.3 SQreamDB Studio Updates and Improvements . . . 747
14.1.4.4 Known Issues . . . 747
14.1.4.5 Version 4.3 resolved Issues . . . 748
14.1.4.6 Configuration Adjustments . . . 748
14.1.4.7 Deprecations . . . 748
14.1.4.8 Upgrading to v4.3 . . . 749
14.1.5 Release Notes 4.4 . . . 750
14.1.5.1 Compatibility Matrix . . . 750

14.1.5.2 New Features and Enhancements . . . 751
14.1.5.3 Newly Released Connector Drivers . . . 751
14.1.5.4 SQreamDB Studio Updates and Improvements . . . 751
14.1.5.5 Known Issues . . . 751
14.1.5.6 Version 4.4 resolved Issues . . . 751
14.1.5.7 Configuration Adjustments . . . 752
14.1.5.8 Deprecations . . . 752
14.1.5.9 Upgrading to Version 4.4 . . . 752
14.1.6 Release Notes 4.5 . . . 753
14.1.6.1 Compatibility Matrix . . . 753
14.1.6.2 New Features and Enhancements . . . 754
14.1.6.3 Known Issues . . . 754
14.1.6.4 Version 4.5 resolved Issues . . . 754
14.1.6.5 Deprecations . . . 754
14.1.6.6 Upgrading to Version 4.5 . . . 754
14.1.7 Release Notes 4.6 . . . 755
14.1.7.1 Compatibility Matrix . . . 755
14.1.7.2 New Features and Enhancements . . . 756
14.1.7.3 Known Issues . . . 756
14.1.7.4 Version 4.6 resolved Issues . . . 756
14.1.7.5 Deprecations . . . 756
14.1.7.6 Upgrading to Version 4.6 . . . 757
14.2 Release Notes 2022.1 . . . 757
14.2.1 Release Notes 2022.1.7 . . . 757
14.2.1.1 New Features . . . 758
14.2.1.2 Storage Version . . . 758
14.2.1.3 Known Issues . . . 758
14.2.1.4 Version 2022.1.7 resolved Issues . . . 758
14.2.1.5 Configuration Changes . . . 758
14.2.1.6 Naming Changes . . . 758
14.2.1.7 Deprecated Features . . . 758
14.2.1.8 End of Support . . . 759
14.2.1.9 Upgrading to v2022.1.7 . . . 759
14.2.2 Release Notes 2022.1.6 . . . 759
14.2.2.1 New Features . . . 760
14.2.2.2 Storage Version . . . 760
14.2.2.3 Known Issues . . . 760
14.2.2.4 Version 2022.1.6 resolved Issues . . . 760
14.2.2.5 Configuration Changes . . . 760
14.2.2.6 Naming Changes . . . 760
14.2.2.7 Deprecated Features . . . 761
14.2.2.8 End of Support . . . 761
14.2.2.9 Upgrading to v2022.1.6 . . . 761
14.2.3 Release Notes 2022.1.5 . . . 762
14.2.3.1 New Features . . . 762
14.2.3.2 Storage Version . . . 762
14.2.3.3 Known Issues . . . 762
14.2.3.4 Resolved Issues . . . 763
14.2.3.5 Operations and Configuration Changes . . . 763
14.2.3.6 Naming Changes . . . 763
14.2.3.7 Deprecated Features . . . 763
14.2.3.8 End of Support . . . 763
14.2.3.9 Upgrading to v2022.1.5 . . . 763
14.2.4 Release Notes 2022.1.4 . . . 764

14.2.4.1 Version Content . . . 764
14.2.4.2 Storage Version . . . 765
14.2.4.3 Known Issues . . . 765
14.2.4.4 Resolved Issues . . . 765
14.2.4.5 Operations and Configuration Changes . . . 765
14.2.4.6 Naming Changes . . . 765
14.2.4.7 Deprecated Features . . . 765
14.2.4.8 End of Support . . . 765
14.2.4.9 Upgrading to v2022.1.4 . . . 766
14.2.5 Release Notes 2022.1.3 . . . 766
14.2.5.1 Version Content . . . 767
14.2.5.2 Storage Version . . . 767
14.2.5.3 Known Issues . . . 767
14.2.5.4 Resolved Issues . . . 767
14.2.5.5 Operations and Configuration Changes . . . 767
14.2.5.6 Naming Changes . . . 768
14.2.5.7 Deprecated Features . . . 768
14.2.5.8 End of Support . . . 768
14.2.5.9 Upgrading to v2022.1.3 . . . 768
14.2.6 Release Notes 2022.1.2 . . . 769
14.2.6.1 Version Content . . . 769
14.2.6.2 Storage Version . . . 769
14.2.6.3 New Features . . . 769
14.2.6.3.1 Parquet Read Optimization . . . 769
14.2.6.4 Resolved Issues . . . 770
14.2.6.5 Operations and Configuration Changes . . . 770
14.2.6.6 Naming Changes . . . 770
14.2.6.7 Deprecated Features . . . 770
14.2.6.8 End of Support . . . 770
14.2.6.9 Upgrading to v2022.1.2 . . . 770
14.2.7 Release Notes 2022.1.1 . . . 771
14.2.7.1 Version Content . . . 771
14.2.7.2 Storage Version . . . 771
14.2.7.3 New Features . . . 771
14.2.7.3.1 Password Security Compliance . . . 772
14.2.7.4 Known Issues . . . 772
14.2.7.5 Resolved Issues . . . 772
14.2.7.6 Operations and Configuration Changes . . . 772
14.2.7.7 Naming Changes . . . 772
14.2.7.8 Deprecated Features . . . 772
14.2.7.9 End of Support . . . 772
14.2.7.10 Upgrading to v2022.1.1 . . . 772
14.2.8 Release Notes 2022.1 . . . 773
14.2.8.1 Version Content . . . 773
14.2.8.2 Storage Version . . . 774
14.2.8.3 New Features . . . 774
14.2.8.3.1 Data Encryption . . . 774
14.2.8.3.2 Update Feature . . . 774
14.2.8.3.3 Avro Ingestion . . . 774
14.2.8.4 Known Issues . . . 774
14.2.8.5 Resolved Issues . . . 775
14.2.8.6 Operations and Configuration Changes . . . 775
14.2.8.7 Naming Changes . . . 775
14.2.8.8 Deprecated Features . . . 775

14.2.8.9 End of Support . . . 775
14.2.8.10 Upgrading to v2022.1 . . . 775
14.3 Release Notes 2021.2 . . . 776
14.3.1 Release Notes 2021.2.1.24 . . . 776
14.3.1.1 Version Content . . . 776
14.3.1.2 New Features . . . 776
14.3.1.2.1 Query Healer . . . 777
14.3.1.3 Resolved Issues . . . 777
14.3.1.4 Known Issues . . . 777
14.3.1.5 Operations and Configuration Changes . . . 777
14.3.1.6 Naming Changes . . . 778
14.3.1.7 Deprecated Features . . . 778
14.3.1.8 End of Support . . . 778
14.3.2 Release Notes 2021.2.1 . . . 778
14.3.2.1 New Features . . . 778
14.3.2.1.1 CREATE TABLE . . . 779
14.3.2.1.2 PERCENTILE FUNCTIONS . . . 779
14.3.2.1.3 REGEX REPLACE . . . 779
14.3.2.1.4 Delete Optimization . . . 779
14.3.2.2 Performance Enhancements . . . 779
14.3.2.3 Resolved Issues . . . 779
14.3.2.4 Known Issues . . . 780
14.3.2.5 Naming Convention Modifications . . . 780
14.3.2.6 End of Support . . . 780
14.3.2.7 Deprecated Features . . . 780
14.3.3 Release Notes 2021.2 . . . 780
14.3.3.1 New Features . . . 780
14.3.3.1.1 New Driver Compatibility . . . 781
14.3.3.1.2 Centralized Configuration System . . . 781
14.3.3.1.3 Qualifying Schemas Without Providing an Alias . . . 781
14.3.3.1.4 Double-Quotations Supported When Importing and Exporting CSVs . . . 781
14.3.3.2 Performance Enhancements . . . 782
14.3.3.3 Resolved Issues . . . 782
14.3.3.4 Known Issues . . . 782
14.3.3.5 Naming Convention Modifications . . . 782
14.3.3.5.1 NVARCHAR Data Type Renamed TEXT . . . 783
14.3.3.6 End of Support . . . 783
14.3.3.7 Deprecated Features . . . 783
14.3.3.8 Upgrading Your SQream Version . . . 783
14.3.3.8.1 Upgrading Your Storage Version . . . 783
14.3.3.8.2 Upgrading Your Client Drivers . . . 783
14.3.3.8.3 Configuring Your Instance of SQream . . . 783

15 Troubleshooting **785**
15.1 Remedying Slow Queries . . . 785
15.2 Resolving Common Issues . . . 787
15.2.1 Troubleshooting Cluster Setup and Configuration . . . 787
15.2.2 Troubleshooting Connectivity Issues . . . 787
15.2.3 Troubleshooting Query Performance . . . 787
15.2.4 Troubleshooting Query Behavior . . . 787
15.2.5 File an issue with SQream support . . . 788
15.3 Identifying Configuration Issues . . . 788
15.4 Lock Related Issues . . . 788
15.5 Log Related Issues . . . 788

15.5.1 Loading Logs with Foreign Tables . . . 789
15.5.2 Counting Message Types . . . 789
15.5.3 Finding Fatal Errors . . . 790
15.5.4 Countng Error Events Within a Certain Timeframe . . . 790
15.5.5 Tracing Errors to Find Offending Statements . . . 790
15.6 Core Dumping Related Issues . . . 791
15.7 Retrieving Execution Plan Output Using SQreamDB Studio . . . 791
15.7.1 Retrieving Execution Plan Output . . . 792
15.8 Gathering Information for SQream Support . . . 792
15.8.1 Getting Support and Reporting Bugs . . . 792
15.8.2 How SQream Debugs Issues . . . 793
15.8.2.1 Reproduce . . . 793
15.8.2.2 Logs . . . 793
15.8.2.3 Fix . . . 793
15.8.3 Collecting a Reproducible Example of a Problematic Statement . . . 793
15.8.3.1 SQL Syntax . . . 793
15.8.3.2 Parameters . . . 794
15.8.3.3 Example . . . 794
15.8.4 Collecting Logs and Metadata Database . . . 794
15.8.4.1 Examples . . . 794
15.8.5 Using the Command Line Utility: . . . 794

16 Glossary **795**

SQreamDB is a columnar analytic SQL database management system. SQreamDB supports regular SQL including *a substantial amount of ANSI SQL*, uses *serializable transactions*, and *scales horizontally* for concurrent statements. Even a *basic SQreamDB machine* can support tens to hundreds of terabytes of data. SQreamDB easily plugs in to third-party tools like *Tableau* comes with standard SQL client drivers, including *JDBC*, *ODBC*, and *Python DB-API*.

Topic		Description
Getting Started		
Preparing Your Machine to Install SQreamDB		Set up your local machine according to SQreamDB's recommended pre-installation configurations.
Executing Statements in SQreamDB		Provides more information about the available methods for executing statements in SQreamDB.
Performing Basic SQream Operations		Provides more information on performing basic operations.
Hardware Guide | Describes SQreamDB's mandatory and recommended hardware settings, designed for a technical audience.		
Installation Guides		
Installing and Launching SQreamDB		Refers to SQreamDB's installation guides.
Installing SQream Studio		Refers to all installation guides required for installations related to Studio.
Ingesting Data		
CSV	*Avro*	
Parquet	*ORC*	
JSON	*External Databases*	
Connecting to SQreamDB		
Client Platforms		Describes how to install and connect a variety of third party connection platforms and tools.
Client Drivers		Describes how to use the SQreamDB client drivers and client applications with SQreamDB.
External Storage Platforms		
Inserting Data Using Amazon S3		Describes how to insert data over a native S3 connector.
Using SQream in an HDFS Environment		Describes how to configure an HDFS environment for the user sqream and is only relevant for users with an HDFS environment.

Need help?

If you couldn't find what you're looking for, we're always happy to help. Visit SQreamDB's support portal for additional support.

CHAPTER

ONE

GETTING STARTED

The **Getting Started** page describes the following things you need to start using SQreamDB:

1.1 Preparing Your Machine to Install SQreamDB

To prepare your machine to install SQreamDB, do the following:

- Set up your local machine according to SQreamDB's recommended pre-installation configurations.

- Verify you have an NVIDIA-capable server, either on-premise or on supported cloud platforms:
  - Red Hat Enterprise Linux v7.x / v8.6 - 8.8
  - CentOS v7.x
  - Amazon Linux 7
- Verify that you have the following:
  - An NVIDIA GPU - SQreamDB recommends using a Tesla GPU.
  - An SSH connection to your server.
  - SUDO permissions for installation and configuration purposes.
  - A SQreamDB license - Contact SQreamDB Support for your license key.

For more information, see the following:

- *Pre-Installation Configuration*
- *Hardware Guide*

1.2 Installing SQreamDB

Method	Description
Installing SQreamDB natively	Describes installing SQreamDB using binary packages provided by SQreamDB

1.3 Executing Statements in SQreamDB

You may choose one of the following tools for executing statements in SQreamDB:

Tool	Description
SQream SQL CLI	A command line interface
SQreamDB Acceleration Studio	An intuitive and easy-to-use interface

1.4 Performing Basic SQream Operations

After installing SQream you can perform the operations described on this page:

1.4.1 Running the SQream SQL Client

The following example shows how to run the SQream SQL client:

```
$ sqream sql --port=5000 --username=rhendricks -d master
Password:

Interactive client mode
To quit, use ^D or \q.
```

Running the SQream SQL client prompts you to provide your password. Use the username and password that you have set up, or your DBA has provided.

Tip:

- You can exit the shell by typing `\q` or `Ctrl-d`.
- A new SQream cluster contains a database named *master,* which is the database used in the examples on this page.

1.4.2 Creating Your First Table

The **Creating Your First Table** section describes the following:

- *Creating a table*
- *Replacing a table*
- *Listing a CREATE TABLE statement*
- *Dropping a table*

Creating a Table

The `CREATE TABLE` syntax is used to create your first table. This table includes a table name and column specifications, as shown in the following example:

```
CREATE TABLE cool_animals (
   id INT NOT NULL,
   name TEXT(20),
```

(continues on next page)

(continued from previous page)

```
   weight INT
);
```

For more information on creating a table, see create_table.

Replacing a Table

You can drop an existing table and create a new one by adding the `OR REPLACE` parameter after the `CREATE` keyword, as shown in the following example:

```
CREATE OR REPLACE TABLE cool_animals (
   id INT NOT NULL,
   name TEXT(20),
   weight INT
);
```

Listing a CREATE TABLE Statement

You can list the full, verbose `CREATE TABLE` statement for a table by using the **GET DDL** function with the table name as shown in the following example:

```
test=> SELECT GET_DDL('cool_animals');
create table "public"."cool_animals" (
"id" int not null,
"name" text(20),
"weight" int
);
```

Note:

- SQream DB identifier names such as table names and column names are not case sensitive. SQreamDB lowercases all identifiers by default. If you want to maintain case, enclose the identifiers with double-quotes.
- SQream DB places all tables in the *public* schema, unless another schema is created and specified as part of the table name.

For information on listing a `CREATE TABLE` statement, see get_ddl.

Dropping a Table

When you have finished working with your table, you can drop the table to remove it table and its content, as shown in the following example:

```
test=> DROP TABLE cool_animals;

executed
```

For more information on dropping tables, see drop_table.

1.4.3 Listing Tables

To see the tables in the current database you can query the catalog, as shown in the following example:

```
SELECT table_name FROM sqream_catalog.tables;
cool_animals
```

1.4.4 Inserting Rows

The **Inserting Rows** section describes the following:

- *Inserting basic rows*
- *Changing value order*
- *Inserting multiple rows*
- *Omitting columns*

Inserting Basic Rows

You can insert basic rows into a table using the `INSERT` statement. The inserted statement includes the table name, an optional list of column names, and column values listed in the same order as the column names, as shown in the following example:

```
INSERT INTO cool_animals VALUES (1, 'Dog', 7);
```

Changing Value Order

You can change the order of values by specifying the column order, as shown in the following example:

```
INSERT INTO cool_animals(weight, id, name) VALUES (3, 2, 'Possum');
```

Inserting Multiple Rows

You can insert multiple rows using the `INSERT` statement by using sets of parentheses separated by commas, as shown in the following example:

```
INSERT INTO cool_animals VALUES
  (3, 'Cat', 5) ,
  (4, 'Elephant', 6500) ,
  (5, 'Rhinoceros', 2100);
```

Note: You can load large data sets using bulk loading methods instead. For more information, see inserting_data.

Omitting Columns

Omitting columns that have a default values (including default `NULL` values) uses the default value, as shown in the following example:

```
INSERT INTO cool_animals (id) VALUES (6);
```

```
INSERT INTO cool_animals (id) VALUES (6);


SELECT * FROM cool_animals;
```

(continues on next page)

(continued from previous page)

```
1,Dog                  ,7
2,Possum               ,3
3,Cat                  ,5
4,Elephant             ,6500
5,Rhinoceros           ,2100
6,\N,\N
```

Note: Null row values are represented as `\N`

For more information on inserting rows, see insert.

For more information on default values, see default value.

1.4.5 Running Queries

The **Running Queries** section describes the following:

- *Running basic queries*
- *Outputting all columns*
- *Outputting shorthand table values*
- *Filtering results*
- *Sorting results*
- *Filtering null rows*

Running Basic Queries

You can run a basic query using the `SELECT` keyword, followed by a list of columns and values to be returned, and the table to get the data from, as shown in the following example:

```
SELECT id, name, weight FROM cool_animals;
1,Dog                  ,7
2,Possum               ,3
3,Cat                  ,5
4,Elephant             ,6500
5,Rhinoceros           ,2100
6,\N,\N
```

For more information on the `SELECT` keyword, see select.

To Output All Columns

You can output all columns without specifying them using the star operator `*`, as shown in the following example:

```
SELECT * FROM cool_animals;
1,Dog                  ,7
2,Possum               ,3
3,Cat                  ,5
4,Elephant             ,6500
5,Rhinoceros           ,2100
6,\N,\N
```

Outputting Shorthand Table Values

You can output the number of values in a table without getting the full result set by using the `COUNT` statement:

```
SELECT COUNT(*) FROM cool_animals;
6
```

Filtering Results

You can filter results by adding a `WHERE` clause and specifying the filter condition, as shown in the following example:

```
SELECT id, name, weight FROM cool_animals WHERE weight > 1000;
4,Elephant            ,6500
5,Rhinoceros          ,2100
```

Sorting Results

You can sort results by adding an `ORDER BY` clause and specifying ascending (`ASC`) or descending (`DESC`) order, as shown in the following example:

```
SELECT * FROM cool_animals ORDER BY weight DESC;
4,Elephant            ,6500
5,Rhinoceros          ,2100
1,Dog                 ,7
3,Cat                 ,5
2,Possum              ,3
6,\N,\N
```

Filtering Null Rows

You can filter null rows by adding an `IS NOT NULL` filter, as shown in the following example:

```
SELECT * FROM cool_animals WHERE weight IS NOT NULL ORDER BY weight DESC;
4,Elephant            ,6500
5,Rhinoceros          ,2100
1,Dog                 ,7
3,Cat                 ,5
2,Possum              ,3
```

For more information, see the following:

- Outputting the number of values in a table without getting the full result set - COUNT(*).
- Filtering results - WHERE
- Sorting results - ORDER BY
- Filtering rows - IS NOT NULL

1.4.6 Deleting Rows

The **Deleting Rows** section describes the following:

- *Deleting selected rows*
- *Deleting all rows*

Deleting Selected Rows

You can delete rows in a table selectively using the `DELETE` command. You must include a table name and *WHERE* clause to specify the rows to delete, as shown in the following example:

```
DELETE FROM cool_animals WHERE weight is null;

executed
SELECT  * FROM cool_animals;
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
4,Elephant            ,6500
5,Rhinoceros          ,2100
```

Deleting All Rows

You can delete all rows in a table using the `TRUNCATE` command followed by the table name, as shown in the following example:

```
TRUNCATE TABLE cool_animals;
```

Note: While truncate deletes data from disk immediately, delete does not physically remove the deleted rows.

For more information, see the following:

- Deleting selected rows - DELETE
- Deleting all rows - TRUNCATE

1.4.7 Saving Query Results to a CSV or PSV File

You can save query results to a CSV or PSV file using the `sqream sql` command from a CLI client. This saves your query results to the selected delimited file format, as shown in the following example:

```
$ sqream sql --username=mjordan --database=nba --host=localhost --port=5000 -c
↪"SELECT * FROM nba LIMIT 5" --results-only --delimiter='|' > nba.psv
$ cat nba.psv
Avery Bradley           |Boston Celtics        |0|PG|25|6-2 |180|Texas              ␣
↪|7730337
Jae Crowder             |Boston Celtics        |99|SF|25|6-6 |235|Marquette          ␣
↪ |6796117
John Holland            |Boston Celtics        |30|SG|27|6-5 |205|Boston University  ␣
↪ |\N
R.J. Hunter             |Boston Celtics        |28|SG|22|6-5 |185|Georgia State      ␣
↪ |1148640
Jonas Jerebko           |Boston Celtics        |8|PF|29|6-10|231|\N|5000000
```

For more output options, see *Controlling the Client Output*.

What's next?

- Explore all of SQream DB's *SQL Syntax*.
- See the full *SQream SQL CLI reference*.
- Connect a third party tool to start analyzing data.

For more information on other basic SQream operations, see the following:

- Creating a Database
- *Data Ingestion Sources*

1.5 Hardware Guide

The **Hardware Guide** describes the SQreamDB reference architecture, emphasizing the benefits to the technical audience, and provides guidance for end-users on selecting the right configuration for a SQreamDB installation.

Need help?

This page is intended as a "reference" to suggested hardware. However, different workloads require different solution sizes. SQreamDB's experienced customer support has the experience to advise on these matters to ensure the best experience.

Visit SQreamDB's support portal for additional support.

- *Cluster Architectures*
  - *Single-Node Cluster*
  - *Multi-Node Cluster*
  - *Metadata Server*
  - *SQreamDB Studio Server*
- *Cluster Design Considerations*
  - *Balancing Cost and Performance*
  - *CPU Compute*
  - *GPU Compute and RAM*
  - *RAM*
  - *Operating System*
  - *Storage*

1.5.1 Cluster Architectures

SQreamDB recommends rackmount servers by server manufacturers Dell, Lenovo, HP, Cisco, Supermicro, IBM, and others.

A typical SQreamDB cluster includes one or more nodes, consisting of:

- Two-socket enterprise processors, such as Intel® Xeon® Gold processors or the IBM® POWER9 processors, providing the high performance required for compute-bound database workloads.

- NVIDIA Tesla GPU accelerators, with up to 5,120 CUDA and Tensor cores, running on PCIe or fast NVLINK busses, delivering high core count, and high-throughput performance on massive datasets.

- High density chassis design, offering between 2 and 4 GPUs in a 1U, 2U, or 3U package, for best-in-class performance per cm^2.

1.5.1.1 Single-Node Cluster

A single-node SQreamDB cluster can handle between 1 and 8 concurrent users, with up to 1PB of data storage (when connected via NAS).

An average single-node cluster can be a rackmount server or workstation, containing the following components:

Component	Type
Server	Dell R750, Dell R940xa, HP ProLiant DL380 Gen10 or similar (Intel only)
Processors	2x Intel Xeon Gold 6348 (28C/56HT) 3.5GHz or similar
RAM	1.5 TB
Onboard storage	• 2x 960GB SSD 2.5in hot plug for OS, RAID1 • 2x 2TB SSD or NVMe, for temporary spooling, RAID0 • 10x 3.84TB SSD 2.5in Hot plug for storage, RAID6
GPU	NVIDIA 2x A100, H100, or L40S
Operating System	Red Hat Enterprise Linux v8.8 or Amazon Linux

Note: If you are using internal storage, your volumes must be formatted as xfs.

In this system configuration, SQreamDB can store about 100TB of raw data (assuming an average compression ratio and ~30TB of usable raw storage).

If a NAS is used, the 10x SSD drives can be omitted, but SQreamDB recommends 2TB of local spool space on SSD or NVMe drives.

1.5.1.2 Multi-Node Cluster

Multi-node clusters can handle any number of concurrent users. A typical SQreamDB cluster relies on a minimum of two GPU-enabled servers and shared storage connected over a network fabric, such as InfiniBand EDR, 40GbE, or 100GbE.

The **Multi-Node Cluster Examples** section describes the following specifications:

The following table shows SQreamDB's recommended hardware specifications:

Component	Type
Server	Dell R750, Dell R940xa, HP ProLiant DL380 Gen10 or similar (Intel only)
Processors	2x Intel Xeon Gold 6348 (28C/56HT) 3.5GHz or similar
RAM	2 TB
Onboard storage	• 2x 960GB SSD 2.5in hot plug for OS, RAID1 • 2x 2TB SSD or NVMe, for temporary spooling, RAID0
Network Card (Storage)	2x Mellanox ConnectX-6 Single Port HDR VPI InfiniBand Adapter cards at 100GbE or similar.
Network Card (Client)	2x 1 GbE cards or similar
External Storage	• Mellanox Connectx5/6 100G NVIDIA Network Card (if applicable) or other high-speed network card minimum 40G compatible with customer's infrastructure • 50 TB (NAS connected over GPFS, Lustre, Weka, or VAST) GPFS recommended
GPU	NVIDIA 2x A100, H100, or L40S
Operating System	Red Hat Enterprise Linux v8.8 or Amazon Linux

1.5.1.3 Metadata Server

The following table shows SQreamDB's recommended metadata server specifications:

Component	Type
Server	Dell R750, Dell R940xa, HP ProLiant DL380 Gen10 or similar (Intel only)
Processors	2x Intel Xeon Gold 6342 2.8 Ghz 24C processors or similar
RAM	512GB DDR4 RAM 8x64GB RDIMM or similar
Onboard storage	2x 960 GB MVMe SSD drives in RAID 1 or similar
Network Card (Storage)	2x Mellanox ConnectX-6 Single Port HDR VPI InfiniBand Adapter cards at 100GbE or similar.
Network Card (Client)	2x 1 GbE cards or similar
Operating System	Red Hat Enterprise Linux v8.8 or Amazon Linux

Note: With a NAS connected over GPFS, Lustre, Weka, or VAST, each SQreamDB worker can read data at 5GB/s or more.

1.5.1.4 SQreamDB Studio Server

The following table shows SQreamDB's recommended Studio server specifications:

Component	Type
Server	Physical or virtual machine
Processor	1x Intel Core i7
RAM	16 GB
Onboard storage	50 GB SSD 2.5in Hot-plug for OS, RAID1
Operating System	Red Hat Enterprise Linux v7.9 or CentOS v7.9

1.5.2 Cluster Design Considerations

This section describes the following cluster design considerations:

- In a SQreamDB installation, the storage and computing are logically separated. While they may reside on the same machine in a standalone installation, they may also reside on different hosts, providing additional flexibility and scalability.

- SQreamDB uses all resources in a machine, including CPU, RAM, and GPU to deliver the best performance. At least 256GB of RAM per physical GPU is recommended.

- Local disk space is required for good temporary spooling performance, particularly when performing intensive operations exceeding the available RAM, such as sorting. SQreamDB recommends an SSD or NVMe drive in RAID0 configuration with about twice the RAM size available for temporary storage. This can be shared with the operating system drive if necessary.

- When using NAS devices, SQreamDB recommends approximately 5GB/s of burst throughput from storage per GPU.

1.5.2.1 Balancing Cost and Performance

Prior to designing and deploying a SQreamDB cluster, a number of important factors must be considered.

The **Balancing Cost and Performance** section provides a breakdown of deployment details to ensure that this installation exceeds or meets the stated requirements. The rationale provided includes the necessary information for modifying configurations to suit the customer use-case scenario, as shown in the following table:

Component	Value
Compute - CPU	Balance price and performance
Compute – GPU	Balance price with performance and concurrency
Memory – GPU RAM	Balance price with concurrency and performance.
Memory - RAM	Balance price and performance
Operating System	Availability, reliability, and familiarity
Storage	Balance price with capacity and performance
Network	Balance price and performance

1.5.2.2 CPU Compute

SQreamDB relies on multi-core Intel Gold Xeon processors or IBM POWER9 processors and recommends a dual-socket machine populated with CPUs with 18C/36HT or better. While a higher core count may not necessarily affect query performance, more cores will enable higher concurrency and better load performance.

1.5.2.3 GPU Compute and RAM

The NVIDIA Tesla range of high-throughput GPU accelerators provides the best performance for enterprise environments. Most cards have ECC memory, which is crucial for delivering correct results every time. SQreamDB recommends the NVIDIA Tesla A100 80GB GPU for the best performance and highest concurrent user support.

GPU RAM, sometimes called GRAM or VRAM, is used for processing queries. It is possible to select GPUs with less RAM. However, the smaller GPU RAM results in reduced concurrency, as the GPU RAM is used extensively in operations like JOINs, ORDER BY, GROUP BY, and all SQL transforms.

1.5.2.4 RAM

SQreamDB requires using **Error-Correcting Code memory (ECC)**, standard on most enterprise servers. Large amounts of memory are required for improved performance for heavy external operations, such as sorting and joining.

SQreamDB recommends at least 256GB of RAM per GPU on your machine.

1.5.2.5 Operating System

SQreamDB can run on the following 64-bit Linux operating systems:

- Red Hat Enterprise Linux (RHEL) v7.9
- CentOS v7.9
- Amazon Linux 2018.03

1.5.2.6 Storage

For clustered scale-out installations, SQreamDB relies on NAS storage. For stand-alone installations, SQreamDB relies on redundant disk configurations, such as RAID 5, 6, or 10. These RAID configurations replicate blocks of data between disks to avoid data loss or system unavailability.

SQreamDB recommends using enterprise-grade SAS SSD or NVMe drives. For a 32-user configuration, the number of GPUs should roughly match the number of users. SQreamDB recommends 1 Tesla A100 / H100 or L40S GPU per 2 users, for full, uninterrupted dedicated access.

1.6 Staging and Development Hardware Guide

The **Staging and Development Hardware Guide** describes the SQream recommended HW for development, staging and or QA desktop and servers.

> **Warning:** The HW specification in this page are not intended for production use!

1.6.1 Development Desktop

Component	Type
Server	PC
Processor	Intel i7
RAM	64GB RAM
Onboard storage	2TB SSD
GPU	1x NVIDIA RTX A4000 16GB
Operating System	Red Hat Enterprise Linux v7.9 or CentOS v7.9

1.6.2 Lab Server

Component	Type
Server	Dell R640 or similar
Processor	x2 Intel(R) Xeon(R) Silver 4112 CPU @ 2.60GHz
RAM	128 or 256 GB
Onboard storage	"2x 960GB SSD 2.5in hot plug for OS, RAID1
	1(or more)x 3.84TB SSD 2.5in Hot plug for storage, RAID5"
GPU	1xNVIDIA T4 or A40 or A10
Operating System	Red Hat Enterprise Linux v7.9 or CentOS v7.9

CHAPTER

TWO

INSTALLATION GUIDES

Before you get started using SQreamDB, consider your business needs and available resources. SQreamDB was designed to run in a number of environments, and to be installed using different methods depending on your requirements. This determines which installation method to use.

The **Installation Guides** section describes the following installation guide sets:

2.1 Installing and Launching SQreamDB

The **Installing and Launching SQreamDB** page includes the following installation guides:

2.1.1 Pre-Installation Configuration

Before installing SQreamDB, it is recommended that you to tune your system for better performance and stability.

- *Recommended Basic Input/Output System Settings*
- *Installing the Operating System*
- *Configuring the Operating System*
- *Installing the NVIDIA CUDA Driver*
- *Enabling Core Dumps*

2.1.1.1 Recommended Basic Input/Output System Settings

The first step when setting your pre-installation configurations is to use the recommended basic input/output system (BIOS) settings.

The BIOS settings may have a variety of names, or may not exist on your system. Each system vendor has a different set of settings and variables. It is safe to skip any and all of the configuration steps, but this may impact performance.

If any doubt arises, consult the documentation for your server or your hardware vendor for the correct way to apply the settings.

Item	Setting	Rationale
Management console access	**Connected**	Connection to Out-of-band (OOB) required to preserve continuous network uptime.
All drives	**Connected and displayed on RAID interface**	Prerequisite for cluster or OS installation.
RAID volumes	**Configured according to project guidelines. Must be rebooted to take effect.**	Clustered to increase logical volume and provide redundancy.
Fan speed Thermal Configuration.	Dell fan speed: **High Maximum**. Specified minimum setting: **60**. HPe thermal configuration: **Increased cooling**.	NVIDIA Tesla GPUs are passively cooled and require high airflow to operate at full performance.
Power regulator or iDRAC power unit policy	HPe: **HP static high performance** mode enabled. Dell: **iDRAC power unit policy** (power cap policy) disabled.	Other power profiles (such as "balanced") throttle the CPU and diminishes performance. Throttling may also cause GPU failure.
System Profile, **Power Profile**, or **Performance Profile**	**High Performance**	The Performance profile provides potentially increased performance by maximizing processor frequency, and the disabling certain power saving features such as C-states. Use this setting for environments that are not sensitive to power consumption.
Power Cap Policy or **Dynamic power capping**	**Disabled**	Other power profiles (like "balanced") throttle the CPU and may diminish performance or cause GPU failure. This setting may appear together with the above (Power profile or Power regulator). This setting allows disabling system ROM power calibration during the boot process. Power regulator settings are named differently in BIOS and iLO/iDRAC.
Intel Turbo Boost	**Enabled**	Intel Turbo Boost enables overclocking the processor to boost CPU-bound operation performance. Overclocking may risk computational jitter due to changes in the processor's turbo frequency. This causes brief pauses in processor operation, introducing uncertainty into application processing time. Turbo operation is a function of power consumption, processor temperature, and the number of active cores.
Logical Processor	**HPe**: Enable **Hyperthreading Dell**: Enable **Logical Processor**	Hyperthreading doubles the amount of logical processors, which may improve performance by ~5-10% for CPU-bound operations.
Intel Virtualization Technology (VT-d)	**Disable**	VT-d is optimal for running VMs. However, when running Linux natively, disabling VT-d boosts performance by up to 10%.
Logical Processor	**HPe**: Enable **Hyperthreading Dell**: Enable **Logical Processor**	Hyperthreading doubles the amount of logical processors, which may improve performance by ~5-10% for CPU-bound operations.
Intel Virtualization Technology (VT-d)	**Disable**	VT-d is optimal for running VMs. However, when running Linux natively, disabling VT-d boosts performance by up to 10%.
Processor C-States (Minimum processor idle power core state)	**Disable**	Processor C-States reduce server power when the system is in an idle state. This causes slower cold-starts when the system transitions from an idle to a load state, and may reduce query performance by up to 15%.
HPe: **Energy/Performance bias**	**Maximum performance**	Configures processor sub-systems for high-performance and low-latency. Other power profiles (like "balanced") throttle the CPU and may diminish performance. Use this setting for environments that are not sensitive to power

2.1.1.2 Installing the Operating System

2.1.1.2.1 Before You Begin

- Your system must have at least 200 gigabytes of free space on the root / mount.
- For a multi-node cluster, you must have external shared storage provided by systems like General Parallel File System (GPFS), Weka, or VAST.
- Once the BIOS settings have been set, you must install the operating system.
- A typical SQreamDB installation requires RHEL7.9/CentOS 7.9 or RHEL8.X (RHEL8.8 recommended)
- Verify the exact RHEL7/RHEL8 version with your storage vendor to avoid driver incompatibility.

2.1.1.2.2 Installation

1. Select a language (English recommended).
2. From **Software Selection**, select **Minimal** and check the **Development Tools** group checkbox.

   Selecting the **Development Tools** group installs the following tools:

   - autoconf
   - automake
   - binutils
   - bison
   - flex
   - gcc
   - gcc-c++
   - gettext
   - libtool
   - make
   - patch
   - pkgconfig
   - redhat-rpm-config
   - rpm-build
   - rpm-sign
3. Continue the installation.
4. Set up the necessary drives and users as per the installation process.

   The OS shell is booted up.

2.1.1.3 Configuring the Operating System

When configuring the operating system, several basic settings related to creating a new server are required. Configuring these as part of your basic set-up increases your server's security and usability.

2.1.1.3.1 Creating a `sqream` User

The sqream user must have the same UID and GID across all servers in your cluster.

If the `sqream` user does not have the same UID and GID across all servers and there is no critical data stored under `/home/sqream`, it is recommended to delete the `sqream` user and sqream group from your servers. Subsequently, create new ones with the same ID, using the following command:

```
sudo userdel sqream
sudo rm /var/spool/mail/sqream
```

Before adding a user with a specific UID and GID, it is crucial to verify that such Ids do not already exist.

The steps below guide you on creating a `sqream` user with an exemplary ID of `1111`.

1. Verify that a `1111` UID does not already exists:

   ```
   cat /etc/passwd |grep 1111
   ```

2. Verify that a `1111` GID does not already exists:

   ```
   cat /etc/group |grep 1111
   ```

3. Add a user with an identical UID on all cluster nodes:

   ```
   useradd -u 1111 sqream
   ```

4. Add a `sqream` user to the `wheel` group.

   ```
   sudo usermod -aG wheel sqream
   ```

   You can remove the `sqream` user from the `wheel` group when the installation and configuration are complete:

   ```
   passwd sqream
   ```

5. Log out and log back in as `sqream`.
6. If you deleted the `sqream` user and recreated it to have a new ID, you must change its ownership to `/home/sqream` in order to avoid permission errors.

   ```
   sudo chown -R sqream:sqream /home/sqream
   ```

2.1.1.3.2 Setting Up A Locale

SQreamDB enables you to set up a locale using your own location. To find out your current time-zone, run the `timedatectl list-timezones` command.

1. Set the language of the locale:

```
sudo localectl set-locale LANG=en_US.UTF-8
```

2. Set the time stamp (time and date) of the locale:

```
sudo timedatectl set-timezone Asia/Jerusalem
```

2.1.1.3.3 Installing Required Software

- *Installing EPEL Repository*
- *Installing Required Packages*
- *Installing Recommended Tools*
- *Installing Python*
- *Installing NodeJS*

2.1.1.3.3.1 Installing EPEL Repository

CentOS7/RHEL8

```
sudo yum install epel-release
```

RHEL7

```
sudo rpm -Uvh http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
↪noarch.rpm
```

RHEL8

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.
↪noarch.rpm
```

2.1.1.3.3.2 Enabling Additional Red Hat Repositories

Enabling additional Red Hat repositories is essential to install the required packages in the subsequent procedures.

RHEL7

```
sudo subscription-manager repos --enable rhel-7-server-optional-rpms
```

RHEL8

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-
↪rpms
sudo subscription-manager repos --enable rhel-8-for-x86_64-appstream-rpms
sudo subscription-manager repos --enable rhel-8-for-x86_64-baseos-rpms
```

2.1.1.3.3.3 Installing Required Packages

RHEL7/CentOS7

```
sudo yum install ntp pciutils monit zlib-devel openssl-devel kernel-devel-
↪$(uname -r) kernel-headers-$(uname -r) gcc net-tools wget jq libffi-devel␣
↪gdbm-devel tk-devel xz-devel sqlite-devel readline-devel bzip2-devel␣
↪ncurses-devel zlib-devel
```

RHEL8

```
sudo dnf install chrony pciutils monit zlib-devel openssl-devel kernel-devel-
↪$(uname -r) kernel-headers-$(uname -r) gcc net-tools wget jq libffi-devel␣
↪xz-devel ncurses-compat-libs libnsl gdbm-devel tk-devel sqlite-devel␣
↪readline-devel texinfo
```

2.1.1.3.3.4 Installing Recommended Tools

RHEL7/CentOS7

```
sudo yum install bash-completion.noarch vim-enhanced vim-common net-tools␣
↪iotop htop psmisc screen xfsprogs wget yum-utils deltarpm dos2unix
```

RHEL8

```
sudo dnf install bash-completion.noarch vim-enhanced vim-common net-tools␣
↪iotop htop psmisc screen xfsprogs wget yum-utils dos2unix
```

2.1.1.3.3.5 Installing Python

For SQreamDb version 4.3 or older, install Python 3.6.7.

1. Download the Python 3.6.7 source code tarball file from the following URL into the `/home/sqream` directory:

```
wget https://www.python.org/ftp/python/3.6.7/Python-3.6.7.tar.xz
```

2. Extract the Python 3.6.7 source code into your current directory:

```
tar -xf Python-3.6.7.tar.xz
```

3. Navigate to the Python 3.6.7 directory:

```
cd Python-3.6.7
```

4. Run the `./configure` script:

```
./configure --enable-loadable-sqlite-extensions
```

5. Build the software:

```
make -j30
```

6. Install the software:

```
sudo make install
```

7. Verify that Python 3.6.7 has been installed:

```
python3 --version
```

For SQreamDB version 4.4 or newer, install Python 3.9.13.

1. Download the Python 3.9.13 source code tarball file from the following URL into the `/home/sqream` directory:

```
wget https://www.python.org/ftp/python/3.9.13/Python-3.9.13.tar.xz
```

2. Extract the Python 3.9.13 source code into your current directory:

```
tar -xf Python-3.9.13.tar.xz
```

3. Navigate to the Python 3.9.13 directory:

```
cd Python-3.9.13
```

4. Run the `./configure` script:

```
./configure --enable-loadable-sqlite-extensions
```

5. Build the software:

```
make -j30
```

6. Install the software:

```
sudo make install
```

7. Verify that Python 3.9.13 has been installed:

```
python3 --version
```

2.1.1.3.3.6 Installing NodeJS

NodeJS is necessary only when the UI runs on the same server as SqreamDB. If not, you can skip this step.

1. Download the NodeJS source code tarball file from the following URL into the `/home/sqream` directory:

```
wget https://nodejs.org/dist/v16.20.0/node-v16.20.0-linux-x64.tar.xz
tar -xf node-v16.20.0-linux-x64.tar.xz
```

2. Move the node-v16.20.0-linux-x64 file to the */usr/local* directory.

```
sudo mv  node-v16.20.0-linux-x64 /usr/local
```

3. Navigate to the `/usr/bin/` directory:

```
cd /usr/bin
```

4. Create a symbolic link to the `/local/node-v16.20.0-linux-x64/bin/node node` directory:

```
sudo ln -s ../local/node-v16.20.0-linux-x64//bin/node node
```

5. Create a symbolic link to the `/local/node-v16.20.0-linux-x64/bin/npm npm` directory:

```
sudo ln -s ../local/node-v16.20.0-linux-x64/bin/npm npm
```

6. Create a symbolic link to the `/local/node-v16.20.0-linux-x64/bin/npx npx` directory:

```
sudo ln -s ../local/node-v16.20.0-linux-x64/bin/npx npx
```

7. Install the `pm2` process management:

```
sudo npm install pm2 -g
```

8. If installing the `pm2` process management fails, install it offline:

   a. On a machine with internet access, install the following:

   - nodejs
   - npm
   - pm2

   b. Extract the pm2 module to the correct directory:

```
cd /usr/local/node-v16.20.0-linux-x64/lib/node_modules
tar -czvf pm2_x86.tar.gz pm2
```

   c. Copy the `pm2_x86.tar.gz` file to a server without access to the internet and extract it.

   d. Move the `pm2` folder to the `/usr/local/node-v16.20.0-linux-x64/lib/node_modules` directory:

```
sudo mv pm2 /usr/local/node-v16.20.0-linux-x64/lib/node_modules
```

   e. Navigate back to the `/usr/bin` directory:

```
cd /usr/bin
```

   f. Create a symbolink to the `pm2` service:

```
sudo ln -s /usr/local/node-v16.20.0-linux-x64/lib/node_modules/pm2/bin/
↪pm2 pm2
```

   g. Verify that installation was successful without using `sudo`:

```
pm2 list
```

   h. Verify that the node versions for the above are correct:

```
node --version
```

2.1.1.3.4 Configuring the Network Time Protocol

This Network Time Protocol (NTP) configuration is intended for use on systems running RHEL7 and CentOS7 exclusively.

If you don't have internet access, see Configure NTP Client to Synchronize with NTP Server.

1. Install the NTP file.

```
sudo yum install ntp
```

2. Enable the **ntpd** program.

```
sudo systemctl enable ntpd
```

3. Start the **ntdp** program.

```
sudo systemctl start ntpd
```

4. Print a list of peers known to the server and a summary of their states.

```
sudo ntpq -p
```

2.1.1.3.5 Configuring the Network Time Protocol Server

If your organization has a Network Time Protocol (NTP) server, configure it by adding records to `/etc/ntpd.conf`, reloading the service, and checking that synchronization is enabled.

1. Output your NTP server address and append `/etc/ntpd.conf` to the output:

```
echo -e "\nserver <your NTP server address>\n" | sudo tee -a /etc/ntp.conf
```

2. Restart the service:

```
sudo systemctl restart ntpd
```

3. Check that synchronization is enabled:

```
sudo timedatectl
```

Checking that synchronization is enabled generates the following output:

```
  Local time: Sat 2019-10-12 17:26:13 EDT
 Universal time: Sat 2019-10-12 21:26:13 UTC
       RTC time: Sat 2019-10-12 21:26:13
      Time zone: America/New_York (EDT, -0400)
    NTP enabled: yes
NTP synchronized: yes
RTC in local TZ: no
     DST active: yes
Last DST change: DST began at
                 Sun 2019-03-10 01:59:59 EST
```

(continues on next page)

(continued from previous page)

```
                 Sun 2019-03-10 03:00:00 EDT
Next DST change: DST ends (the clock jumps one hour backwards) at
                 Sun 2019-11-03 01:59:59 EDT
                 Sun 2019-11-03 01:00:00 EST
```

2.1.1.3.6 Configuring Chrony for RHEL8 Only

1. Start the Chrony service:

```
sudo systemctl start chronyd
```

2. Enable the Chrony service to start automatically at boot time:

```
sudo systemctl enable chronyd
```

3. Check the status of the Chrony service:

```
sudo systemctl status chronyd
```

2.1.1.3.7 Configuring the Server to Boot Without Linux GUI

We recommend that you configure your server to boot without a Linux GUI by running the following command:

```
sudo systemctl set-default multi-user.target
```

Running this command activates the **NO-UI** server mode.

2.1.1.3.8 Configuring the Security Limits

The security limits refer to the number of open files, processes, etc.

```
sudo bash
```

```
echo -e "sqream soft nproc 1000000\nsqream hard nproc 1000000\nsqream soft␣
↪nofile 1000000\nsqream hard nofile 1000000\nroot soft nproc 1000000\nroot␣
↪hard nproc 1000000\nroot soft nofile 1000000\nroot hard nofile 1000000\
↪nsqream soft core unlimited\nsqream hard core unlimited" >> /etc/security/
↪limits.conf
```

2.1.1.3.9 Configuring the Kernel Parameters

1. Insert a new line after each kernel parameter:

```
echo -e "vm.dirty_background_ratio = 5 \n vm.dirty_ratio = 10 \n vm.swappiness =␣
↪10 \n vm.vfs_cache_pressure = 200 \n vm.zone_reclaim_mode = 0 \n" >> /etc/
↪sysctl.conf
```

2. Check the maximum value of the `fs.file`:

```
sysctl -n fs.file-max
```

3. **Only** if the maximum value of the `fs.file` is smaller than `2097152`, run the following command:

```
echo "fs.file-max=2097152" >> /etc/sysctl.conf
```

2.1.1.3.10 Configuring the Firewall

The example in this section shows the open ports for four `sqreamd` sessions. If more than four are required, open the required ports as needed. Port 8080 in the example below is a new UI port.

The ports listed below are required, and the same logic applies to all additional SQreamDB Worker ports.

Port	Use
8080	UI port
443	UI over HTTPS (requires nginx installation)
3105	SqreamDB metadataserver service
3108	SqreamDB serverpicker service
3109	SqreamDB serverpicker service over ssl
5000	SqreamDB first worker default port
5100	SqreamDB first worker over ssl default port
5001	SqreamDB second worker default port
5101	SqreamDB second worker over ssl default port

1. Start the service and enable FirewallID on boot:

```
systemctl start firewalld
```

2. Add the following ports to the permanent firewall:

```
firewall-cmd --zone=public --permanent --add-port=8080/tcp
firewall-cmd --zone=public --permanent --add-port=3105/tcp
firewall-cmd --zone=public --permanent --add-port=3108/tcp
firewall-cmd --zone=public --permanent --add-port=5000-5003/tcp
firewall-cmd --zone=public --permanent --add-port=5100-5103/tcp
firewall-cmd --permanent --list-all
```

3. Reload the firewall:

```
firewall-cmd --reload
```

4. Enable FirewallID on boot:

```
systemctl enable firewalld
```

If you do not need the firewall, you can disable it:

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```

2.1.1.3.11 Disabling SELinux

Disabling SELinux is a recommended action.

1. Show the status of `selinux`:

```
sudo sestatus
```

2. If the output is not `disabled`, edit the `/etc/selinux/config` file:

```
sudo vim /etc/selinux/config
```

3. Change `SELINUX=enforcing` to `SELINUX=disabled`:

   The above changes will only take effect after rebooting the server.

   You can disable selinux immediately after rebooting the server by running the following command:

```
sudo setenforce 0
```

2.1.1.3.12 Configuring the `/etc/hosts` File

1. Edit the `/etc/hosts` file:

```
sudo vim /etc/hosts
```

2. Call your local host:

```
127.0.0.1 localhost
<server1 ip>      <server_name>
<server2 ip>      <server_name>
```

2.1.1.4 Installing the NVIDIA CUDA Driver

After configuring your operating system, you must install the NVIDIA CUDA driver.

Warning: If your Linux GUI runs on the server, it must be stopped before installing the CUDA drivers.

1. Verify that the NVIDIA card has been installed and is detected by the system:

```
lspci | grep -i nvidia
```

2. Verify that `gcc` has been installed:

```
gcc --version
```

3. If `gcc` has not been installed, install it for RHEL or CentOS:

```
sudo yum install -y gcc
```

1. Update the kernel headers on RHEL or CentOS:

```
sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r)
```

2. Make sure kernel-devel and kernel-headers match installed kernel:

```
uname -r
rpm -qa |grep kernel-devel-$(uname -r)
rpm -qa |grep kernel-headers-$(uname -r)
```

Disable Nouveau, which is the default operating system driver.

1. Check if the Nouveau driver has been loaded:

```
lsmod | grep nouveau
```

   If the Nouveau driver has been loaded, the command above generates output. If the Nouveau driver has not been loaded, you may skip step 2 and 3.

2. Blacklist the Nouveau driver to disable it:

```
cat <<EOF | sudo tee /etc/modprobe.d/blacklist-nouveau.conf
blacklist nouveau
options nouveau modeset=0
EOF
```

3. Regenerate the kernel `initramfs` directory set:

a. Modify the `initramfs` directory set:

```
sudo dracut --force
```

b. Reboot the server:

```
sudo reboot
```

2.1.1.4.1 Installing the CUDA Driver

- *Installing the CUDA Driver from the Repository*
- *Tuning Up NVIDIA Performance*

The current recommendation is for CUDA 12.2.1.

For A100 GPU and other A series GPUs, you must install CUDA driver **11.4.3** or a later version.

For H100 GPU and other H series GPUs, you must install CUDA driver **11.8** or a later version.

For L40S GPU and other L series GPUs, you must install CUDA driver **12.0** or a later version.

For questions related to which driver to install, contact SqreamDB support.

2.1.1.4.1.1 Installing the CUDA Driver from the Repository

Installing the CUDA driver from the Repository is the recommended installation method.

1. Install the CUDA dependencies for one of the following operating systems:
   - For RHEL7:

     ```
     sudo rpm -Uvh http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
     ↪noarch.rpm
     ```

   - For RHEL8:

     ```
     sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.
     ↪noarch.rpm
     ```

   - For CentOS:

     ```
     sudo yum install epel-release
     ```

2. (Optional) Install the CUDA dependencies from the `epel` repository:

   ```
   sudo yum install dkms libvdpau
   ```

   Installing the CUDA depedendencies from the `epel` repository is only required for installing `runfile`.

3. Download and install the required local repository:
   - **CentOS7/RHEL7 CUDA 11.4.3 repository (INTEL) installation (Required for A-Series GPU models):**

     ```
     wget https://developer.download.nvidia.com/compute/cuda/11.4.3/local_
     ↪installers/cuda-repo-rhel7-11-4-local-11.4.3_470.82.01-1.x86_64.rpm
     sudo yum localinstall cuda-repo-rhel7-11-4-local-11.4.3_470.82.01-1.x86_64.rpm
     ```

     ```
     sudo yum clean all
     sudo yum -y install nvidia-driver-latest-dkms
     ```

   - **RHEL8.6 CUDA 11.4.3 repository (INTEL) installation (Required for A-Series GPU models):**

     ```
     wget https://developer.download.nvidia.com/compute/cuda/11.4.3/local_
     ↪installers/cuda-repo-rhel8-11-4-local-11.4.3_470.82.01-1.x86_64.rpm
     sudo dnf localinstall cuda-repo-rhel8-11-4-local-11.4.3_470.82.01-1.
     ↪x86_64.rpm
     ```

   - **RHEL8.8 CUDA 12.2.1 repository (INTEL) installation (Required for H/L Series GPU models):**

     ```
     wget https://developer.download.nvidia.com/compute/cuda/12.2.1/local_
     ↪installers/cuda-repo-rhel8-12-2-local-12.2.1_535.86.10-1.x86_64.rpm
     sudo dnf localinstall cuda-repo-rhel8-12-2-local-12.2.1_535.86.10-1.
     ↪x86_64.rpm
     ```

     ```
     sudo dnf clean all
     sudo dnf -y module install nvidia-driver:latest-dkms
     ```

Power9 with V100 GPUs supports only cuda 10.1 driver on RHEL7

- **IBM Power9 - CUDA 10.1 for RHEL7:**

```
wget https://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_
↪installers/cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.ppc64le.rpm
sudo yum localinstall cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.ppc64le.
↪rpm
```

If you are installing the CUDA driver on IBM Power9, please follow the NVIDIA documentation for additionally required steps.

2.1.1.4.1.2 Tuning Up NVIDIA Performance

The following procedures exclusively relate to Intel.

- *Tune Up NVIDIA Performance when Driver Installed from the Repository*
- *Tune Up NVIDIA Performance when Driver Installed from the Runfile*

Note: Setting up the NVIDIA POWER9 CUDA driver includes additional set-up requirements. The NVIDIA POWER9 CUDA driver will not function properly if the additional set-up requirements are not followed. See POWER9 Setup for the additional set-up requirements.

2.1.1.4.1.3 Tune Up NVIDIA Performance when Driver Installed from the Repository

1. Check the service status:

   ```
   sudo systemctl status nvidia-persistenced
   ```

   If the service exists, it will be stopped by default.

2. Start the service:

   ```
   sudo systemctl start nvidia-persistenced
   ```

3. Verify that no errors have occurred:

   ```
   sudo systemctl status nvidia-persistenced
   ```

4. Enable the service to start up on boot:

   ```
   sudo systemctl enable nvidia-persistenced
   ```

5. For **H100/A100**, add the following lines:

   ```
   nvidia-persistenced
   ```

6. Reboot the server and run the **NVIDIA System Management Interface (NVIDIA SMI)**:

   ```
   nvidia-smi
   ```

2.1.1.4.1.4 Tune Up NVIDIA Performance when Driver Installed from the Runfile

1. Change the permissions on the `rc.local` file to `executable`:

```
sudo chmod +x /etc/rc.local
```

2. Edit the `/etc/yum.repos.d/cuda-10-1-local.repo` file:

```
sudo vim /etc/rc.local
```

3. Add the following lines:

   - **For H100/A100**:

```
nvidia-persistenced
```

   - **For IBM (mandatory)**:

```
sudo systemctl start nvidia-persistenced
sudo systemctl enable nvidia-persistenced
```

   - **For K80**:

```
nvidia-persistenced
nvidia-smi -pm 1
nvidia-smi -acp 0
nvidia-smi --auto-boost-permission=0
nvidia-smi --auto-boost-default=0
```

4. Reboot the server and run the `NVIDIA System Management Interface (NVIDIA SMI)`:

```
nvidia-smi
```

2.1.1.5 Enabling Core Dumps

While this procedure is optional, SQreamDB recommends that core dumps be enabled. Note that the default `abrt` format is not `gdb` compatible, and that for SQreamDB support to be able to analyze your core dumps, they must be `gdb` compatible.

- *Checking the `abrtd` Status*
- *Setting the Limits*
- *Creating the Core Dump Directory*
- *Setting the Output Directory on the `/etc/sysctl.conf` File*
- *Verifying that the Core Dumps Work*
- *Verify Your SQreamDB Installation*
- *Troubleshooting Core Dumping*

2.1.1.5.1 Checking the `abrtd` Status

1. Check if `abrtd` is running:

```
sudo ps -ef |grep abrt
```

2. If **abrtd** is running, stop it:

```
for i in abrt-ccpp.service abrtd.service abrt-oops.service abrt-pstoreoops.
↪service abrt-vmcore.service abrt-xorg.service ; do sudo systemctl disable $i;␣
↪sudo systemctl stop $i; done
```

2.1.1.5.2 Setting the Limits

1. Set the limits:

```
ulimit -c
```

2. If the output is `0`, add the following lines to the `/etc/security/limits.conf` file:

```
*          soft     core           unlimited
*          hard     core           unlimited
```

3. To apply the limit changes, log out and log back in.

2.1.1.5.3 Creating the Core Dump Directory

Because the core dump file may be the size of total RAM on the server, verify that you have sufficient disk space. In the example above, the core dump is configured to the `/tmp/core_dumps` directory. If necessary, replace path according to your own environment and disk space.

1. Make the `/tmp/core_dumps` directory:

```
mkdir /tmp/core_dumps
```

2. Set the ownership of the `/tmp/core_dumps` directory:

```
sudo chown sqream.sqream /tmp/core_dumps
```

3. Grant read, write, and execute permissions to all users:

```
sudo chmod -R 777 /tmp/core_dumps
```

2.1.1.5.4 Setting the Output Directory on the `/etc/sysctl.conf` File

1. Open the `/etc/sysctl.conf` file in the Vim text editor:

```
sudo vim /etc/sysctl.conf
```

2. Add the following to the bottom of the file:

```
kernel.core_uses_pid = 1
kernel.core_pattern = /tmp/core_dumps/core-%e-%s-%u-%g-%p-%t
fs.suid_dumpable = 2
```

3. To apply the changes without rebooting the server, run the following:

```
sudo sysctl -p
```

4. Check that the core output directory points to the following:

```
sudo cat /proc/sys/kernel/core_pattern
```

The following shows the correct generated output:

```
/tmp/core_dumps/core-%e-%s-%u-%g-%p-%t
```

2.1.1.5.5 Verifying that the Core Dumps Work

You can verify that the core dumps work only after installing and running SQreamDB. This causes the server to crash and a new `core.xxx` file to be included in the folder that is written in `/etc/sysctl.conf`.

1. Stop and restart all SQreamDB services.

2. Connect to SQreamDB with ClientCmd and run the following command:

```
select abort_server();
```

2.1.1.5.6 Verify Your SQreamDB Installation

1. Verify that the `sqream` user exists and has the same ID on all cluster servers.

```
id sqream
```

2. please verify that the storage is mounted on all cluster servers.

```
mount
```

3. make sure that the driver is properly installed.

```
nvidia-smi
```

4. Verify that the kernel file-handles allocation is greater than or equal to `2097152`:

```
sysctl -n fs.file-max
```

5. Verify limits (run this command as a `sqream` user):

```
ulimit -c -u -n

Desired output:
core file size (blocks, -c) unlimited
max user processes (-u) 1000000
open files (-n) 1000000
```

2.1.1.5.7 Troubleshooting Core Dumping

This section describes the troubleshooting procedure to be followed if all parameters have been configured correctly, but the cores have not been created.

1. Reboot the server.

2. Verify that you have folder permissions:

```
sudo chmod -R 777 /tmp/core_dumps
```

3. Verify that the limits have been set correctly:

```
ulimit -c
```

   If all parameters have been configured correctly, the correct output is:

```
core file size          (blocks, -c) unlimited
```

4. If all parameters have been configured correctly, but running `ulimit -c` outputs `0`, run the following:

```
sudo vim /etc/profile
```

5. Search for the following line and disable it using the # symbol:

```
ulimit -S -c 0 > /dev/null 2>&1
```

6. Log out and log back in.

7. Run the `ulimit -c` command:

```
ulimit -a
```

8. If the line is not found in `/etc/profile`, do the following:

   a. Run the following command:

```
sudo vim /etc/init.d/functions
```

   b. Search for the following line disable it using the # symbol and reboot the server.

```
ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>&1
```

2.1.2 Installing SQream Using Binary Packages

This procedure describes how to install SQream using Binary packages and must be done on all servers.

To install SQream using Binary packages:

1. Copy the SQream package to the **/home/sqream** directory for the current version:

```
$ tar -xf sqream-db-v<2020.2>.tar.gz
```

2. Append the version number to the name of the SQream folder. The version number in the following example is **v2020.2**:

```
$ mv sqream sqream-db-v<2020.2>
```

3. Move the new version of the SQream folder to the **/usr/local/** directory:

```
$ sudo mv sqream-db-v<2020.2> /usr/local/
```

4. Change the ownership of the folder to **sqream folder**:

```
$ sudo chown -R sqream:sqream  /usr/local/sqream-db-v<2020.2>
```

5. Navigate to the **/usr/local/** directory and create a symbolic link to SQream:

```
$ cd /usr/local
$ sudo ln -s sqream-db-v<2020.2> sqream
```

6. Verify that the symbolic link that you created points to the folder that you created:

```
$ ls -l
```

7. Verify that the symbolic link that you created points to the folder that you created:

```
$ sqream -> sqream-db-v<2020.2>
```

8. Create the SQream configuration file destination folders and set their ownership to **sqream**:

```
$ sudo mkdir /etc/sqream
$ sudo chown -R sqream:sqream /etc/sqream
```

9. Create the SQream service log destination folders and set their ownership to **sqream**:

```
$ sudo mkdir /var/log/sqream
$ sudo chown -R sqream:sqream /var/log/sqream
```

10. Navigate to the **/usr/local/** directory and copy the SQream configuration files from them:

```
$ cd /usr/local/sqream/etc/
$ cp * /etc/sqream
```

The configuration files are **service configuration files**, and the JSON files are **SQream configuration files**, for a total of four files. The number of SQream configuration files and JSON files must be identical.

Note: Verify that the JSON files have been configured correctly and that all required flags have been set to the correct values.

In each JSON file, the following parameters **must be updated**:

- instanceId
- machineIP
- metadataServerIp
- spoolMemoryGB
- limitQueryMemoryGB
- gpu
- port

- ssl_port

See how to *configure* the Spool Memory and Limit Query Memory.

Note the following:

- The value of the **metadataServerIp** parameter must point to the IP that the metadata is running on.
- The value of the **machineIP** parameter must point to the IP of your local machine.

It would be same on server running metadataserver and different on other server nodes.

11. **Optional** - To run additional SQream services, copy the required configuration files and create additional JSON files:

```
$ cp sqream2_config.json sqream3_config.json
$ vim sqream3_config.json
```

Note: A unique **instanceID** must be used in each JSON file. IN the example above, the instanceID **sqream_2** is changed to **sqream_3**.

12. **Optional** - If you created additional services in **Step 11**, verify that you have also created their additional configuration files:

```
$ cp sqream2-service.conf sqream3-service.conf
$ vim sqream3-service.conf
```

13. For each SQream service configuration file, do the following:
    1. Change the **SERVICE_NAME=sqream2** value to **SERVICE_NAME=sqream3**.
    2. Change **LOGFILE=/var/log/sqream/sqream2.log** to **LOGFILE=/var/log/sqream/sqream3.log**.

Note: If you are running SQream on more than one server, you must configure the `serverpicker` and `metadatserver` services to start on only one of the servers. If **metadataserver** is running on the first server, the `metadataServerIP` value in the second server's /etc/sqream/sqream1_config.json file must point to the IP of the server on which the `metadataserver` service is running.

14. Set up **servicepicker**:
    1. Do the following:

    ```
    $ vim /etc/sqream/server_picker.conf
    ```

    2. Change the IP **127.0.0.1** to the IP of the server that the **metadataserver** service is running on.
    3. Change the **CLUSTER** to the value of the cluster path.
15. Set up your service files:

```
$ cd /usr/local/sqream/service/
$ cp sqream2.service sqream3.service
$ vim sqream3.service
```

16. Increment each **EnvironmentFile=/etc/sqream/sqream2-service.conf** configuration file for each SQream service file, as shown below:

```
$ EnvironmentFile=/etc/sqream/sqream<3>-service.conf
```

17. Copy and register your service files into systemd:

```
$ sudo cp metadataserver.service /usr/lib/systemd/system/
$ sudo cp serverpicker.service /usr/lib/systemd/system/
$ sudo cp sqream*.service /usr/lib/systemd/system/
```

18. Verify that your service files have been copied into systemd:

```
$ ls -l /usr/lib/systemd/system/sqream*
$ ls -l /usr/lib/systemd/system/metadataserver.service
$ ls -l /usr/lib/systemd/system/serverpicker.service
$ sudo systemctl daemon-reload
```

19. Copy the license into the **/etc/license** directory:

```
$ cp license.enc /etc/sqream/
```

If you have an HDFS environment, see *Configuring an HDFS Environment for the User sqream*.

2.1.3 Installing Monit

2.1.3.1 Getting Started

Before installing SQream with Monit, verify that you have followed the required *recommended pre-installation configurations*.

The procedures in the **Installing Monit** guide must be performed on each SQream cluster node.

2.1.3.2 Overview

Monit is a free open source supervision utility for managing and monitoring Unix and Linux. Monit lets you view system status directly from the command line or from a native HTTP web server. Monit can be used to conduct automatic maintenance and repair, such as executing meaningful causal actions in error situations.

SQream uses Monit as a watchdog utility, but you can use any other utility that provides the same or similar functionality.

The **Installing Monit** procedures describes how to install, configure, and start Monit.

You can install Monit in one of the following ways:

- *Installing Monit on CentOS*
- *Installing Monit on CentOS offline*
- *Installing Monit on Ubuntu*
- *Installing Monit on Ubuntu offline*

2.1.3.2.1 Installing Monit on CentOS:

To install Monit on CentOS:

1. Install Monit as a superuser on CentOS:

```
$ sudo yum install monit
```

2.1.3.2.2 Installing Monit on CentOS Offline:

Installing Monit on CentOS offline can be done in either of the following ways:

- *Building Monit from Source Code*
- *Building Monit from Pre-Built Binaries*

2.1.3.2.2.1 Building Monit from Source Code

To build Monit from source code:

1. Copy the Monit package for the current version:

```
$ tar zxvf monit-<x.y.z>.tar.gz
```

   The value `x.y.z` denotes the version numbers.

2. Navigate to the directory where you want to store the package:

```
$ cd monit-x.y.z
```

3. Configure the files in the package:

```
$ ./configure (use ./configure --help to view available options)
```

4. Build and install the package:

```
$ make && make install
```

The following are the default storage directories:

- The Monit package: **/usr/local/bin/**
- The **monit.1 man-file**: **/usr/local/man/man1/**

5. **Optional** - To change the above default location(s), use the **–prefix** option to ./configure.
6. **Optional** - Create an RPM package for CentOS directly from the source code:

```
$ rpmbuild -tb monit-x.y.z.tar.gz
```

2.1.3.2.2.2 Building Monit from Pre-Built Binaries

To build Monit from pre-built binaries:

1. Copy the Monit package for the current version:

```
$ tar zxvf monit-x.y.z-linux-x64.tar.gz
```

   The value `x.y.z` denotes the version numbers.

2. Navigate to the directory where you want to store the package:
3. Copy the **bin/monit** and **/usr/local/bin/** directories:

```
$ cp bin/monit /usr/local/bin/
```

4. Copy the **conf/monitrc** and **/etc/** directories:

```
$ cp conf/monitrc /etc/
```

For examples of pre-built Monit binarties, see Download Precompiled Binaries.

Back to top

2.1.3.2.3 Installing Monit on Ubuntu:

To install Monit on Ubuntu:

1. Install Monit as a superuser on Ubuntu:

```
$ sudo apt-get install monit
```

Back to top

2.1.3.2.4 Installing Monit on Ubuntu Offline:

You can install Monit on Ubuntu when you do not have an internet connection.

To install Monit on Ubuntu offline:

1. Compress the required file:

```
$ tar zxvf monit-<x.y.z>-linux-x64.tar.gz
```

   NOTICE: *<x.y.z>* denotes the version number.

2. Navigate to the directory where you want to save the file:

```
$ cd monit-x.y.z
```

3. Copy the **bin/monit** directory into the **/usr/local/bin/** directory:

```
$ cp bin/monit /usr/local/bin/
```

4. Copy the **conf/monitrc** directory into the **/etc/** directory:

```
$ cp conf/monitrc /etc/
```

Back to top

2.1.3.3 Configuring Monit

When the installation is complete, you can configure Monit. You configure Monit by modifying the Monit configuration file, called **monitrc**. This file contains blocks for each service that you want to monitor.

The following is an example of a service block:

```
$ #SQREAM1-START
$ check process sqream1 with pidfile /var/run/sqream1.pid
$ start program = "/usr/bin/systemctl start sqream1"
$ stop program = "/usr/bin/systemctl stop sqream1"
$ #SQREAM1-END
```

For example, if you have 16 services, you can configure this block by copying the entire block 15 times and modifying all service names as required, as shown below:

```
$ #SQREAM2-START
$ check process sqream2 with pidfile /var/run/sqream2.pid
$ start program = "/usr/bin/systemctl start sqream2"
$ stop program = "/usr/bin/systemctl stop sqream2"
$ #SQREAM2-END
```

For servers that don't run the **metadataserver** and **serverpicker** commands, you can use the block example above, but comment out the related commands, as shown below:

```
$ #METADATASERVER-START
$ #check process metadataserver with pidfile /var/run/metadataserver.pid
$ #start program = "/usr/bin/systemctl start metadataserver"
$ #stop program = "/usr/bin/systemctl stop metadataserver"
$ #METADATASERVER-END
```

To configure Monit:

1. Copy the required block for each required service.
2. Modify all service names in the block.
3. Copy the configured **monitrc** file to the **/etc/monit.d/** directory:

   ```
   $ cp monitrc /etc/monit.d/
   ```

4. Set file permissions to **600** (full read and write access):

   ```
   $ sudo chmod 600 /etc/monit.d/monitrc
   ```

5. Reload the system to activate the current configurations:

   ```
   $ sudo systemctl daemon-reload
   ```

6. **Optional** - Navigate to the **/etc/sqream** directory and create a symbolic link to the **monitrc** file:

   ```
   $ cd /etc/sqream
   $ sudo ln -s /etc/monit.d/monitrc monitrc
   ```

2.1.3.4 Starting Monit

After configuring Monit, you can start it.

To start Monit:

1. Start Monit as a super user:

```
$ sudo systemctl start monit
```

2. View Monit's service status:

```
$ sudo systemctl status monit
```

3. If Monit is functioning correctly, enable the Monit service to start on boot:

```
$ sudo systemctl enable monit
```

2.1.4 Launching SQream with Monit

This procedure describes how to launch SQream using Monit.

2.1.4.1 Launching SQream

After doing the following, you can launch SQream according to the instructions on this page.

1. *Installing Monit*
2. *Installing SQream with Binary*

The following is an example of a working monitrc file configured to monitor the ***metadataserver** and **serverpicker** commands, and **four sqreamd services**. The **monitrc** configuration file is located in the **conf/monitrc** directory.

Note that the **monitrc** in the following example is configured for eight `sqreamd` services, but that only the first four are enabled:

```
$ set daemon  5              # check services at 30 seconds intervals
$ set logfile syslog
$
$ set httpd port 2812 and
$      use address localhost  # only accept connection from localhost
$      allow localhost        # allow localhost to connect to the server and
$      allow admin:monit      # require user 'admin' with password 'monit'
$
$  ##set mailserver smtp.gmail.com port 587
$  ##        using tlsv12
$  #METADATASERVER-START
$  check process metadataserver with pidfile /var/run/metadataserver.pid
$  start program = "/usr/bin/systemctl start metadataserver"
$  stop program = "/usr/bin/systemctl stop metadataserver"
$  #METADATASERVER-END
$  #      alert user@domain.com on {nonexist, timeout}
$  #                      with mail-format {
$  #                            from:     Monit@$HOST
$  #                            subject:  metadataserver $EVENT - $ACTION
$  #                            message:  This is an automate mail, sent from monit.
$  #                    }
```

(continues on next page)

(continued from previous page)

```
$  #SERVERPICKER-START
$  check process serverpicker with pidfile /var/run/serverpicker.pid
$  start program = "/usr/bin/systemctl start serverpicker"
$  stop program = "/usr/bin/systemctl stop serverpicker"
$  #SERVERPICKER-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                                    with mail-format {
$  #                                          from:     Monit@$HOST
$  #                                          subject:  serverpicker $EVENT - $ACTION
$  #                                         message:  This is an automate mail, sent␣
↪from monit.
$  #
$  #
$  #SQREAM1-START
$  check process sqream1 with pidfile /var/run/sqream1.pid
$  start program = "/usr/bin/systemctl start sqream1"
$  stop program = "/usr/bin/systemctl stop sqream1"
$  #SQREAM1-END
$  #        alert user@domain.com on {nonexist, timeout}
$  #               with mail-format {
$  #                     from:     Monit@$HOST
$  #                     subject:  sqream1 $EVENT - $ACTION
$  #                     message:  This is an automate mail, sent from monit.
$  #              }
$  #SQREAM2-START
$  check process sqream2 with pidfile /var/run/sqream2.pid
$  start program = "/usr/bin/systemctl start sqream2"
$  #SQREAM2-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #               with mail-format {
$  #                     from:     Monit@$HOST
$  #                     subject:  sqream1 $EVENT - $ACTION
$  #                     message:  This is an automate mail, sent from monit.
$  #              }
$  #SQREAM3-START
$  check process sqream3 with pidfile /var/run/sqream3.pid
$  start program = "/usr/bin/systemctl start sqream3"
$  stop program = "/usr/bin/systemctl stop sqream3"
$  #SQREAM3-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #               with mail-format {
$  #                     from:     Monit@$HOST
$  #                     subject:  sqream2 $EVENT - $ACTION
$  #                     message:  This is an automate mail, sent from monit.
$  #              }
$  #SQREAM4-START
$  check process sqream4 with pidfile /var/run/sqream4.pid
$  start program = "/usr/bin/systemctl start sqream4"
$  stop program = "/usr/bin/systemctl stop sqream4"
$  #SQREAM4-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                      with mail-format {
$  #                            from:     Monit@$HOST
$  #                            subject:  sqream2 $EVENT - $ACTION
$  #                            message:  This is an automate mail, sent from monit.
$  #                    }
$  #
```

(continues on next page)

(continued from previous page)

```
$  #SQREAM5-START
$  #check process sqream5 with pidfile /var/run/sqream5.pid
$  #start program = "/usr/bin/systemctl start sqream5"
$  #stop program = "/usr/bin/systemctl stop sqream5"
$  #SQREAM5-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                     with mail-format {
$  #                           from:     Monit@$HOST
$  #                           subject:  sqream2 $EVENT - $ACTION
$  #                           message:  This is an automate mail, sent from monit.
$  #                    }
$  #
$  #SQREAM6-START
$  #check process sqream6 with pidfile /var/run/sqream6.pid
$  #start program = "/usr/bin/systemctl start sqream6"
$  #stop program = "/usr/bin/systemctl stop sqream6"
$  #SQREAM6-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                     with mail-format {
$  #                           from:     Monit@$HOST
$  #                           subject:  sqream2 $EVENT - $ACTION
$  #                           message:  This is an automate mail, sent from monit.
$  #                    }
$  #
$  #SQREAM7-START
$  #check process sqream7 with pidfile /var/run/sqream7.pid
$  #start program = "/usr/bin/systemctl start sqream7"
$  #stop program = "/usr/bin/systemctl stop sqream7"
$  #SQREAM7-END
$  #                     with mail-format {
$  #                           from:     Monit@$HOST
$  #                           subject:  sqream2 $EVENT - $ACTION
$  #                           message:  This is an automate mail, sent from monit.
$  #                    }
$  #
$  #SQREAM8-START
$  #check process sqream8 with pidfile /var/run/sqream8.pid
$  #start program = "/usr/bin/systemctl start sqream8"
$  #stop program = "/usr/bin/systemctl stop sqream8"
$  #SQREAM8-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                     with mail-format {
$  #                           from:     Monit@$HOST
$  #                           subject:  sqream2 $EVENT - $ACTION
$  #                           message:  This is an automate mail, sent from monit.
$  #                    }
```

2.1.4.2 Monit Usage Examples

This section shows examples of two methods for stopping the **sqream3** service use Monit's command syntax:

- *Stopping Monit and SQream separately*
- *Stopping SQream using a Monit command*

2.1.4.2.1 Stopping Monit and SQream Separately

You can stop the Monit service and SQream separately as follows:

```
$ sudo systemctl stop monit
$ sudo systemctl stop sqream3
```

You can restart Monit as follows:

```
$ sudo systemctl start monit
```

Restarting Monit automatically restarts the SQream services.

2.1.4.2.2 Stopping SQream Using a Monit Command

You can stop SQream using a Monit command as follows:

```
$ sudo monit stop sqream3
```

This command stops SQream only (and not Monit).

You can restart SQream as follows:

```
$ sudo monit start sqream3
```

2.1.4.2.3 Monit Command Line Options

The **Monit Command Line Options** section describes some of the most commonly used Monit command options.

You can show the command line options by running:

```
$ monit --help
```

```
$ start all             - Start all services
$ start <name>          - Only start the named service
$ stop all              - Stop all services
$ stop <name>           - Stop the named service
$ restart all           - Stop and start all services
$ restart <name>        - Only restart the named service
$ monitor all           - Enable monitoring of all services
$ monitor <name>        - Only enable monitoring of the named service
$ unmonitor all         - Disable monitoring of all services
$ unmonitor <name>      - Only disable monitoring of the named service
$ reload                - Reinitialize monit
$ status [name]         - Print full status information for service(s)
$ summary [name]        - Print short status information for service(s)
```

(continues on next page)

(continued from previous page)

```
$ report [up|down|..]   - Report state of services. See manual for options
$ quit                  - Kill the monit daemon process
$ validate              - Check all services and start if not running
$ procmatch <pattern>   - Test process matching pattern
```

2.1.4.3 Using Monit While Upgrading Your Version of SQream

While upgrading your version of SQream, you can use Monit to avoid conflicts (such as service start). This is done by pausing or stopping all running services while you manually upgrade SQream. When you finish successfully upgrading SQream, you can use Monit to restart all SQream services

To use Monit while upgrading your version of SQream:

1. Stop all actively running SQream services:

```
$ sudo monit stop all
```

2. Verify that SQream has stopped listening on ports **500X**, **510X**, and **310X**:

```
$ sudo netstat -nltp    #to make sure sqream stopped listening on 500X, 510X and␣
↪310X ports.
```

The example below shows the old version `sqream-db-v2020.2` being replaced with the new version `sqream-db-v2025.200`.

```
$ cd /home/sqream
$ mkdir tempfolder
$ mv sqream-db-v2025.200.tar.gz tempfolder/
$ tar -xf sqream-db-v2025.200.tar.gz
$ sudo mv sqream /usr/local/sqream-db-v2025.200
$ cd /usr/local
$ sudo chown -R sqream:sqream sqream-db-v2025.200
$ sudo rm sqream   #This only should remove symlink
$ sudo ln -s sqream-db-v2025.200 sqream   #this will create new symlink named
↪"sqream" pointing to new version
$ ls -l
```

The symbolic SQream link should point to the real folder:

```
$ sqream -> sqream-db-v2025.200
```

4. Restart the SQream services:

```
$ sudo monit start all
```

5. Verify that the latest version has been installed:

```
$ SELECT SHOW_VERSION();
```

The correct version is output.

6. Restart the UI:

```
$ pm2 start all
```

2.2 Installing SQream Studio

The **Installing SQream Studio** page incudes the following installation guides:

2.2.1 Installing Prometheus Exporter

The **Installing Prometheus Exporters** guide includes the following sections:

- *Overview*
- *Adding a User and Group*
- *Cloning the Prometheus GIT Project*
- *Installing the Node Exporter and NVIDIA Exporter*
- *Installing the Process Exporter*
- *Opening the Firewall Ports*

2.2.1.1 Overview

The **Prometheus** exporter is an open-source systems monitoring and alerting toolkit. It is used for collecting metrics from an operating system and exporting them to a graphic user interface.

The Installing Prometheus Exporters guide describes how to installing the following exporters:

- The **Node_exporter** - the basic exporter used for displaying server metrics, such as CPU and memory.
- The **Nvidia_exporter** - shows Nvidia GPU metrics.
- The **process_exporter** - shows data belonging to the server's running processes.

For information about more exporters, see Exporters and Integration

2.2.1.2 Adding a User and Group

Adding a user and group determines who can run processes.

You can add users with the following command:

```
$ sudo groupadd --system prometheus
```

You can add groups with the following command:

```
$ sudo useradd -s /sbin/nologin --system -g prometheus prometheus
```

2.2.1.3 Cloning the Prometheus GIT Project

After adding a user and group you must clone the Prometheus GIT project.

You can clone the Prometheus GIT project with the following command:

```
$ git clone http://gitlab.sq.l/IT/promethues.git prometheus
```

Note: If you experience difficulties cloning the Prometheus GIT project or receive an error, contact your IT department.

The following shows the result of cloning your Prometheus GIT project:

```
$ prometheus/
$ ├── node_exporter
$ |   └── node_exporter
$ ├── nvidia_exporter
$ |   └── nvidia_exporter
$ ├── process_exporter
$ |   └── process-exporter_0.5.0_linux_amd64.rpm
$ ├── README.md
$ └── services
$     ├── node_exporter.service
$     └── nvidia_exporter.service
```

2.2.1.4 Installing the Node Exporter and NVIDIA Exporter

After cloning the Prometheus GIT project you must install the **node_exporter** and **NVIDIA_exporter**.

To install the node_exporter and NVIDIA_exporter:

1. Navigate to the cloned folder:

   ```
   $ cd prometheus
   ```

2. Copy **node_exporter** and **nvidia_exporter** to **/usr/bin/**.

   ```
   $ sudo cp node_exporter/node_exporter /usr/bin/
   $ sudo cp nvidia_exporter/nvidia_exporter /usr/bin/
   ```

3. Copy the **services** files to the services folder:

   ```
   $ sudo cp services/node_exporter.service /etc/systemd/system/
   $ sudo cp services/nvidia_exporter.service /etc/systemd/system/
   ```

4. Reload the services so that they can be run:

   ```
   $ sudo systemctl daemon-reload
   ```

5. Set the permissions and group for both service files:

   ```
   $ sudo chown prometheus:prometheus /usr/bin/node_exporter
   $ sudo chmod u+x /usr/bin/node_exporter
   $ sudo chown prometheus:prometheus /usr/bin/nvidia_exporter
   $ sudo chmod u+x /usr/bin/nvidia_exporter
   ```

6. Start both services:

```
$ sudo systemctl start node_exporter && sudo systemctl enable node_exporter
```

7. Set both services to start automatically when the server is booted up:

```
$ sudo systemctl start nvidia_exporter && sudo systemctl enable nvidia_exporter
```

8. Verify that the server's status is **active (running)**:

```
$ sudo systemctl status node_exporter && sudo systemctl status nvidia_exporter
```

The following is the correct output:

```
$ ● node_exporter.service - Node Exporter
$    Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor␣
↪preset: disabled)
$    Active: active (running) since Wed 2019-12-11 12:28:31 IST; 1 months 5 days␣
↪ago
$  Main PID: 28378 (node_exporter)
$    CGroup: /system.slice/node_exporter.service
$
$ ● nvidia_exporter.service - Nvidia Exporter
$    Loaded: loaded (/etc/systemd/system/nvidia_exporter.service; enabled; vendor␣
↪preset: disabled)
$    Active: active (running) since Wed 2020-01-22 13:40:11 IST; 31min ago
$  Main PID: 1886 (nvidia_exporter)
$    CGroup: /system.slice/nvidia_exporter.service
$            └─1886 /usr/bin/nvidia_exporter
```

2.2.1.5 Installing the Process Exporter

After installing the **node_exporter** and **Nvidia_exporter** you must install the **process_exporter**.

To install the process_exporter:

1. Do one of the following:
   - For **CentOS**, run `sudo rpm -i process_exporter/process-exporter_0.5.0_linux_amd64.rpm`.
   - For **Ubuntu**, run `sudo dpkg -i process_exporter/process-exporter_0.6.0_linux_amd64.deb`.
2. Verify that the process_exporter is running:

```
$ sudo systemctl status process-exporter
```

3. Set the process_exporter to start automatically when the server is booted up:

```
$ sudo systemctl enable process-exporter
```

2.2.1.6 Opening the Firewall Ports

After installing the **process_exporter** you must open the firewall ports for the following services:

- **node_exporter** - port: 9100
- **nvidia_exporter** - port: 9445
- **process-exporter** - port: 9256

Note: This procedure is only relevant if your firwall is running.

To open the firewall ports:

1. Run the following command:

```
$ sudo firewall-cmd --zone=public --add-port=<PORT NUMBER>/tcp --permanent
```

2. Reload the firewall:

```
$ sudo firewall-cmd --reload
```

3. Verify that the changes have taken effect.

2.2.2 Installing Prometheus Using Binary Packages

Prometheus is an application used for event monitoring and alerting.

- *Installing Prometheus*
- *Configuring Your Prometheus Settings*
- *Configuring Your Prometheus Service File*
- *Accessing the Prometheus User Interface*

2.2.2.1 Installing Prometheus

You must install Prometheus before installing the Dashboard Data Collector.

To install Prometheus:

1. Verify the following:
    1. That you have **sudo** access to your Linux server.
    2. That your server has access to the internet (for downloading the Prometheus binary package).
    3. That your firewall rules are opened for accessing Prometheus Port 9090.
2. Navigate to the Prometheus Download page and download the **prometheus-2.32.0-rc.1.linux-amd64.tar.gz** package.
3. Do the following:
    1. Download the source using the `curl` command:

```
$ curl -LO url -LO https://github.com/prometheus/prometheus/releases/download/
↪v2.22.0/prometheus-2.22.0.linux-amd64.tar.gz
```

2. Extract the file contents:

```
$ tar -xvf prometheus-2.22.0.linux-amd64.tar.gz
```

3. Rename the extracted folder **prometheus-files**:

```
$ mv prometheus-2.22.0.linux-amd64 prometheus-files
```

4. Create a Prometheus user:

```
$ sudo useradd --no-create-home --shell /bin/false prometheus
```

5. Create your required directories:

```
$ sudo mkdir /etc/prometheus
$ sudo mkdir /var/lib/prometheus
```

6. Set the Prometheus user as the owner of your required directories:

```
$ sudo chown prometheus:prometheus /etc/prometheus
$ sudo chown prometheus:prometheus /var/lib/prometheus
```

7. Copy the Prometheus and Promtool binary packages from the **prometheus-files** folder to **/usr/local/bin**:

```
$ sudo cp prometheus-files/prometheus /usr/local/bin/
$ sudo cp prometheus-files/promtool /usr/local/bin/
```

8. Change the ownership to the prometheus user:

```
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
$ sudo chown prometheus:prometheus /usr/local/bin/promtool
```

9. Move the **consoles** and **consoles_libraries** directories from **prometheus-files** folder to **/etc/prometheus** folder:

```
$ sudo cp -r prometheus-files/consoles /etc/prometheus
    $ sudo cp -r prometheus-files/console_libraries /etc/prometheus
```

10. Change the ownership to the prometheus user:

```
$ sudo chown -R prometheus:prometheus /etc/prometheus/consoles
$ sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

For more information on installing the Dashboard Data Collector, see *Installing the Dashboard Data Collector*.

Back to *Installing Prometheus Using Binary Packages*

2.2.2.2 Configuring Your Prometheus Settings

After installing Prometheus you must configure your Prometheus settings. You must perform all Prometheus configurations in the **/etc/prometheus/prometheus.yml** file.

To configure your Prometheus settings:

1. Create your **prometheus.yml** file:

```
$ sudo vi /etc/prometheus/prometheus.yml
```

2. Copy the contents below into your prometheus.yml file:

```
$ #node_exporter port : 9100
$ #nvidia_exporter port: 9445
$ #process-exporter port: 9256
$
$ global:
$   scrape_interval: 10s
$
$ scrape_configs:
$   - job_name: 'prometheus'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <prometheus server IP>:9090
$   - job_name: 'processes'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <process exporters iP>:9256
$         - <another process exporters iP>:9256
$   - job_name: 'nvidia'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <nvidia exporter IP>:9445
$         - <another nvidia exporter IP>:9445
$   - job_name: 'nodes'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <node exporter IP>:9100
$         - <another node exporter IP>:9100
```

3. Change the ownership of the file to the prometheus user:

```
$ sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Back to *Installing Prometheus Using Binary Packages*

2.2.2.3 Configuring Your Prometheus Service File

After configuring your Prometheus settings you must configure your Prometheus service file.

To configure your Prometheus service file:

1. Create your **prometheus.yml** file:

```
$ sudo vi /etc/systemd/system/prometheus.service
```

2. Copy the contents below into your prometheus service file:

```
$ [Unit]
$ Description=Prometheus
$ Wants=network-online.target
$ After=network-online.target
$
$ [Service]
$ User=prometheus
$ Group=prometheus
$ Type=simple
$ ExecStart=/usr/local/bin/prometheus \
$     --config.file /etc/prometheus/prometheus.yml \
$     --storage.tsdb.path /var/lib/prometheus/ \
$     --web.console.templates=/etc/prometheus/consoles \
$     --web.console.libraries=/etc/prometheus/console_libraries
$
$ [Install]
$ WantedBy=multi-user.target
```

3. Register the prometheus service by reloading the **systemd** service:

```
$ sudo systemctl daemon-reload
```

4. Start the prometheus service:

```
$ sudo systemctl start prometheus
```

5. Check the status of the prometheus service:

```
$ sudo systemctl status prometheus
```

   If the status is `active (running)`, you have configured your Prometheus service file correctly.

Back to *Installing Prometheus Using Binary Packages*

2.2.2.4 Accessing the Prometheus User Interface

After configuring your Prometheus service file, you can access the Prometheus user interface.

You can access the Prometheus user interface by running the following command:

```
$ http://<prometheus-ip>:9090/graph
```

Once the Prometheus user interface is displayed, go to the **Query** tab and query metrics.

2.2.3 Installing the Dashboard Data Collector

2.2.3.1 Installing the Dashboard Data Collector

After accessing the Prometheus user interface, you can install the **Dashboard Data Collector**. You must install the Dashboard Data Collector to enable the Dashboard in Studio.

Note: Before installing the Dashboard Data collector, verify that Prometheus has been installed and configured for the cluster.

How to install Prometheus from tarball - **Comment - this needs to be its own page.**

To install the Dashboard Data Collector:

1. Store the Data Collector Package obtained from SQream Artifactory.

2. Extract and rename the package:

```
$ tar -xvf dashboard-data-collector-0.5.2.tar.gz
$ mv package dashboard-data-collector
```

3. Change your directory to the location of the package folder:

```
$ cd dashboard-data-collector
```

4. Set up the data collection by modifying the SQream and Data Collector IPs, ports, user name, and password according to the cluster:

```
$ npm run setup -- \
$         --host=127.0.0.1 \
$         --port=3108 \
$         --database=master \
$         --is-cluster=true \
$         --service=sqream \
$         --dashboard-user=sqream \
$         --dashboard-password=sqream \
$         --prometheus-url=http://127.0.0.1:9090/api/v1/query
```

5. Debug the Data Collector: (**Comment** - *using the npm project manager*).

```
$ npm start
```

A json file is generated in the log, as shown below:

```
$ {
$   "machines": [
$     {
$       "machineId": "dd4af489615",
$       "name": "Server 0",
$       "location": "192.168.4.94",
$       "totalMemory": 31.19140625,
$       "gpus": [
$         {
$           "gpuId": "GPU-b17575ec-eeba-3e0e-99cd-963967e5ee3f",
$           "machineId": "dd4af489615",
```

(continues on next page)

(continued from previous page)

```
$           "name": "GPU 0",
$           "totalMemory": 3.9453125
$         }
$       ],
$       "workers": [
$         {
$           "workerId": "sqream_01",
$           "gpuId": "",
$           "name": "sqream_01"
$         }
$       ],
$       "storageWrite": 0,
$       "storageRead": 0,
$       "freeStorage": 0
$     },
$     {
$       "machineId": "704ec607174",
$       "name": "Server 1",
$       "location": "192.168.4.95",
$       "totalMemory": 31.19140625,
$       "gpus": [
$         {
$           "gpuId": "GPU-8777c14f-7611-517a-e9c7-f42eeb21700b",
$           "machineId": "704ec607174",
$           "name": "GPU 0",
$           "totalMemory": 3.9453125
$         }
$       ],
$       "workers": [
$         {
$           "workerId": "sqream_02",
$           "gpuId": "",
$           "name": "sqream_02"
$         }
$       ],
$       "storageWrite": 0,
$       "storageRead": 0,
$       "freeStorage": 0
$     }
$   ],
$   "clusterStatus": true,
$   "storageStatus": {
$     "dataStorage": 49.9755859375,
$     "totalDiskUsage": 52.49829018075231,
$     "storageDetails": {
$       "data": 0,
$       "freeData": 23.7392578125,
$       "tempData": 0,
$       "deletedData": 0,
$       "other": 26.236328125
$     },
$     "avgThroughput": {
$       "read": 0,
$       "write": 0
$     },
$     "location": "/"
$   },
```

(continues on next page)

(continued from previous page)

```
$   "queues": [
$     {
$       "queueId": "sqream",
$       "name": "sqream",
$       "workerIds": [
$         "sqream_01",
$         "sqream_02"
$       ]
$     }
$   ],
$   "queries": [],
$   "collected": true,
$   "lastCollect": "2021-11-17T12:46:31.601Z"
$ }
```

Note: Verify that all machines and workers are correctly registered.

6. Press **CTRL + C** to stop `npm start` (**Comment** - *It may be better to refer to it as the npm project manager*).

7. Start the Data Collector with the `pm2` service:

```
$ pm2 start ./index.js --name=dashboard-data-collector
```

8. Add the following parameter to the SQream Studio setup defined in *Step 4* in **Installing Studio** below.

```
--data-collector-url=http://127.0.0.1:8100/api/dashboard/data
```

Back to *Installing Studio on a Stand-Alone Server*

2.2.4 Installing Studio on a Stand-Alone Server

A stand-alone server is a server that does not run SQreamDB based on binary files.

- *Installing NodeJS Version 12 on the Server*
- *Installing Studio*
- *Starting Studio Manually*
- *Starting Studio as a Service*
- *Accessing Studio*
- *Maintaining Studio with the Process Manager (PM2)*
- *Upgrading Studio*

2.2.4.1 Installing NodeJS Version 12 on the Server

Before installing Studio you must install NodeJS version 12 on the server.

To install NodeJS version 12 on the server:

1. Check if a version of NodeJS older than version *12.<x.x>* has been installed on the target server.

```
$ node -v
```

The following is the output if a version of NodeJS has already been installed on the target server:

```
bash: /usr/bin/node: No such file or directory
```

2. If a version of NodeJS older than *12.<x.x>* has been installed, remove it as follows:
   - On CentOS:

```
$ sudo yum remove -y nodejs
```

   - On Ubuntu:

```
$ sudo apt remove -y nodejs
```

3. If you have not installed NodeJS version 12, run the following commands:
   - On CentOS:

```
$ curl -sL https://rpm.nodesource.com/setup_12.x | sudo bash -
$ sudo yum clean all && sudo yum makecache fast
$ sudo yum install -y nodejs
```

   - On Ubuntu:

```
$ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

The following output is displayed if your installation has completed successfully:

```
Transaction Summary
================================================================================================
Install  1 Package

Total download size: 22 M
Installed size: 67 M
Downloading packages:
warning: /var/cache/yum/x86_64/7/nodesource/packages/nodejs-12.22.1-
↪1nodesource.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID 34fa74dd:␣
↪NOKEY
Public key for nodejs-12.22.1-1nodesource.x86_64.rpm is not installed
nodejs-12.22.1-1nodesource.x86_64.rpm                                       ␣
↪                           |  22 MB  00:00:02
Retrieving key from file:///etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Importing GPG key 0x34FA74DD:
 Userid     : "NodeSource <gpg-rpm@nodesource.com>"
 Fingerprint: 2e55 207a 95d9 944b 0cc9 3261 5ddb e8d4 34fa 74dd
 Package    : nodesource-release-el7-1.noarch (installed)
 From       : /etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Running transaction check
```

(continues on next page)

(continued from previous page)

```
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
  Installing : 2:nodejs-12.22.1-1nodesource.x86_64                          ␣
↪                                                 1/1
  Verifying  : 2:nodejs-12.22.1-1nodesource.x86_64                          ␣
↪                                                 1/1

Installed:
  nodejs.x86_64 2:12.22.1-1nodesource

Complete!
```

4. Confirm the Node version.

```
$ node -v
```

The following is an example of the correct output:

```
v12.22.1
```

5. Install Prometheus using binary packages.

For more information on installing Prometheus using binary packages, see *Installing Prometheus Using Binary Packages*.

Back to *Installing Studio on a Stand-Alone Server*

2.2.4.2 Installing Studio

After installing the Dashboard Data Collector, you can install Studio.

To install Studio:

1. Copy the SQream Studio package from SQream Artifactory into the target server. For access to the Sqream Studio package, contact SQream Support.
2. Extract the package:

```
$ tar -xvf sqream-acceleration-studio-<version number>.x86_64.tar.gz
```

3. Navigate to the new package folder.

```
$ cd sqream-admin
```

4. Build the configuration file to set up Sqream Studio. You can use IP address **127.0.0.1** on a single server.

```
$ npm run setup -- -y --host=<SQreamD IP> --port=3108 --data-collector-url=http://
↪<data collector IP address>:8100/api/dashboard/data
```

The above command creates the **sqream-admin-config.json** configuration file in the **sqream-admin** folder and shows the following output:

```
Config generated successfully. Run `npm start` to start the app.
```

For more information about the available set-up arguments, see Set-Up Arguments.

5. To access Studio over a secure connection, in your configuration file do the following:
   1. Change your `port` value to **3109**.
   2. Change your `ssl` flag value to **true**.

      The following is an example of the correctly modified configuration file:

```
{
  "debugSqream": false,
  "webHost": "localhost",
  "webPort": 8080,
  "webSslPort": 8443,
  "logsDirectory": "",
  "clusterType": "standalone",
  "dataCollectorUrl": "",
  "connections": [
    {
      "host": "127.0.0.1",
      "port":3109,
      "isCluster": true,
      "name": "default",
      "service": "sqream",
      "ssl":true,
      "networkTimeout": 60000,
      "connectionTimeout": 3000
    }
  ]
}
```

5. If you have installed Studio on a server where SQream is already installed, move the **sqream-admin-config.json** file to **/etc/sqream/**:

```
$ mv sqream-admin-config.json /etc/sqream
```

Back to *Installing Studio on a Stand-Alone Server*

2.2.4.3 Starting Studio Manually

You can start Studio manually by running the following command:

```
$ cd /home/sqream/sqream-admin
$ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio -- start
```

The following output is displayed:

```
[PM2] Starting /home/sqream/sqream-admin/server/build/main.js in fork_mode (1␣
↪instance)
[PM2] Done.
┌────┬────────────────────┬─────────────┬─────────┬───────┬──────────┬────────┬─────┬──────────┬
│ id │ name               │ namespace   │ version │ mode  │ pid      │ uptime │ ↺   │
↪│ status    │ cpu      │ mem      │ user     │ watching │
├────┼────────────────────┼─────────────┼─────────┼───────┼──────────┼────────┼─────┼──────────┼
│ 0  │ sqream-studio      │ default     │ 0.1.0   │ fork  │ 11540    │ 0s     │ 0   │
```

(continues on next page)

(continued from previous page)

```
↪| online     | 0%         | 15.6mb    | sqream    | disabled |
```

2.2.4.4 Starting Studio as a Service

Sqream uses the **Process Manager (PM2)** to maintain Studio.

To start Studio as a service:

1. Run the following command:

```
$ sudo npm install -g pm2
```

2. Verify that the PM2 has been installed successfully.

```
$ pm2 list
```

The following is the output:

```
┌────┬──────────────────┬─────────────┬─────────┬───────┬──────────┬────────┬──────┬
│ id │ name             │ namespace   │ version │ mode  │ pid      │ uptime │␣
↪↺   │ status   │ cpu      │ mem      │ user    │ watching │
├────┼──────────────────┼─────────────┼─────────┼───────┼──────────┼────────┼──────┼
│ 0  │ sqream-studio    │ default     │ 0.1.0   │ fork  │ 11540    │ 2m     │␣
↪0   │ online   │ 0%       │ 31.5mb   │ sqream  │ disabled │
└────┴──────────────────┴─────────────┴─────────┴───────┴──────────┴────────┴──────┴
```

2. Start the service with PM2:

   - If the **sqream-admin-config.json** file is located in **/etc/sqream/**, run the following command:

```
$ cd /home/sqream/sqream-admin
$ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio --
↪ start --config-location=/etc/sqream/sqream-admin-config.json
```

   - If the **sqream-admin-config.json** file is not located in **/etc/sqream/**, run the following command:

```
$ cd /home/sqream/sqream-admin
$ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio --
↪ start
```

3. Verify that Studio is running.

```
$ netstat -nltp
```

4. Verify that SQream_studio is listening on port 8080, as shown below:

```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       ␣
↪PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      -
tcp6       0      0 :::8080                 :::*                    LISTEN      ␣
↪11540/sqream-studio
```

(continues on next page)

(continued from previous page)

```
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:25                  :::*                    LISTEN      -
```

5. Verify the following:
   1. That you can access Studio from your browser (`http://<IP_Address>:8080`).
   2. That you can log in to SQream.
6. Save the configuration to run on boot.

```
$ pm2 startup
```

The following is an example of the output:

```
$ sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -
↪u sqream --hp /home/sqream
```

7. Copy and paste the output above and run it.
8. Save the configuration.

```
$ pm2 save
```

Back to *Installing Studio on a Stand-Alone Server*

2.2.4.5 Accessing Studio

The Studio page is available on port 8080: `http://<server ip>:8080`.

If port 8080 is blocked by the server firewall, you can unblock it by running the following command:

```
$ firewall-cmd --zone=public --add-port=8080/tcp --permanent
$ firewall-cmd --reload
```

Back to *Installing Studio on a Stand-Alone Server*

2.2.4.6 Maintaining Studio with the Process Manager (PM2)

Sqream uses the **Process Manager (PM2)** to maintain Studio.

You can use PM2 to do one of the following:

- To check the PM2 service status: `pm2 list`
- To restart the PM2 service: `pm2 reload sqream-studio`
- To see the PM2 service logs: `pm2 logs sqream-studio`

Back to *Installing Studio on a Stand-Alone Server*

2.2.4.7 Upgrading Studio

To upgrade Studio you need to stop the version that you currently have.

To stop the current version of Studio:

1. List the process name:

```
$ pm2 list
```

   The process name is displayed.

```
<process name>
```

2. Run the following command with the process name:

```
$ pm2 stop <process name>
```

3. If only one process is running, run the following command:

```
$ pm2 stop all
```

4. Change the name of the current **sqream-admin** folder to the old version.

```
$ mv sqream-admin sqream-admin-<old_version>
```

5. Extract the new Studio version.

```
$ tar -xf sqream-acceleration-studio-<version>tar.gz
```

6. Rebuild the configuration file. You can use IP address **127.0.0.1** on a single server.

```
$ npm run setup -- -y --host=<SQreamD IP> --port=3108
```

   The above command creates the **sqream-admin-config.json** configuration file in the **sqream_admin** folder.

7. Copy the **sqream-admin-config.json** configuration file to **/etc/sqream/** to overwrite the old configuration file.
8. Start PM2.

```
$ pm2 start all
```

Back to *Installing Studio on a Stand-Alone Server*

2.2.5 Installing an NGINX Proxy Over a Secure Connection

Configuring your NGINX server to use a strong encryption for client connections provides you with secure servers requests, preventing outside parties from gaining access to your traffic.

The **Installing an NGINX Proxy Over a Secure Connection** page describes the following:

- *Overview*
- *Prerequisites*
- *Installing NGINX and Adjusting the Firewall*
- *Creating Your SSL Certificate*

- *Configuring NGINX to use SSL*
- *Redirecting Studio Access from HTTP to HTTPS*
- *Activating Your NGINX Configuration*
- *Verifying that NGINX is Running*

2.2.5.1 Overview

The Node.js platform that SQream uses with our Studio user interface is susceptible to web exposure. This page describes how to implement HTTPS access on your proxy server to establish a secure connection.

TLS (Transport Layer Security), and its predecessor **SSL (Secure Sockets Layer)**, are standard web protocols used for wrapping normal traffic in a protected, encrypted wrapper. This technology prevents the interception of server-client traffic. It also uses a certificate system for helping users verify the identity of sites they visit. The **Installing an NGINX Proxy Over a Secure Connection** guide describes how to set up a self-signed SSL certificate for use with an NGINX web server on a CentOS 7 server.

Note: A self-signed certificate encrypts communication between your server and any clients. However, because it is not signed by trusted certificate authorities included with web browsers, you cannot use the certificate to automatically validate the identity of your server.

A self-signed certificate may be appropriate if your domain name is not associated with your server, and in cases where your encrypted web interface is not user-facing. If you do have a domain name, using a CA-signed certificate is generally preferrable.

For more information on setting up a free trusted certificate, see How To Secure Nginx with Let's Encrypt on CentOS 7.

2.2.5.2 Prerequisites

The following prerequisites are required for installing an NGINX proxy over a secure connection:

- Super user privileges
- A domain name to create a certificate for

2.2.5.3 Installing NGINX and Adjusting the Firewall

After verifying that you have the above preriquisites, you must verify that the NGINX web server has been installed on your machine.

Though NGINX is not available in the default CentOS repositories, it is available from the **EPEL (Extra Packages for Enterprise Linux)** repository.

To install NGINX and adjust the firewall:

1. Enable the EPEL repository to enable server access to the NGINX package:

```
$ sudo yum install epel-release
```

2. Install NGINX:

```
$ sudo yum install nginx
```

3. Start the NGINX service:

```
$ sudo systemctl start nginx
```

4. Verify that the service is running:

```
$ systemctl status nginx
```

The following is an example of the correct output:

```
Output● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor␣
↪preset: disabled)
   Active: active (running) since Fri 2017-01-06 17:27:50 UTC; 28s ago

. . .

Jan 06 17:27:50 centos-512mb-nyc3-01 systemd[1]: Started The nginx HTTP and␣
↪reverse proxy server.
```

5. Enable NGINX to start when your server boots up:

```
$ sudo systemctl enable nginx
```

6. Verify that access to **ports 80 and 443** are not blocked by a firewall.

7. Do one of the following:

   - If you are not using a firewall, skip to *Creating Your SSL Certificate*.

   - If you have a running firewall, open ports 80 and 443:

```
$ sudo firewall-cmd --add-service=http
$ sudo firewall-cmd --add-service=https
$ sudo firewall-cmd --runtime-to-permanent
```

8. If you have a running **iptables firewall**, for a basic rule set, add HTTP and HTTPS access:

```
$ sudo iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
$ sudo iptables -I INPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

Note: The commands in Step 8 above are highly dependent on your current rule set.

9. Verify that you can access the default NGINX page from a web browser.

2.2.5.4 Creating Your SSL Certificate

After installing NGINX and adjusting your firewall, you must create your SSL certificate.

TLS/SSL combines public certificates with private keys. The SSL key, kept private on your server, is used to encrypt content sent to clients, while the SSL certificate is publicly shared with anyone requesting content. In addition, the SSL certificate can be used to decrypt the content signed by the associated SSL key. Your public certificate is located in the **/etc/ssl/certs** directory on your server.

This section describes how to create your **/etc/ssl/private directory**, used for storing your private key file. Because the privacy of this key is essential for security, the permissions must be locked down to prevent unauthorized access:

To create your SSL certificate:

1. Set the following permissions to **private**:

```
$ sudo mkdir /etc/ssl/private
$ sudo chmod 700 /etc/ssl/private
```

2. Create a self-signed key and certificate pair with OpenSSL with the following command:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/
↪private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

The following list describes the elements in the command above:

- **openssl** - The basic command line tool used for creating and managing OpenSSL certificates, keys, and other files.
- **req** - A subcommand for using the X.509 **Certificate Signing Request (CSR)** management. A public key infrastructure standard, SSL and TLS adhere X.509 key and certificate management regulations.
- **-x509** - Used for modifying the previous subcommand by overriding the default functionality of generating a certificate signing request with making a self-signed certificate.
- **-nodes** - Sets **OpenSSL** to skip the option of securing our certificate with a passphrase, letting NGINX read the file without user intervention when the server is activated. If you don't use **-nodes** you must enter your passphrase after every restart.
- **-days 365** - Sets the certificate's validation duration to one year.
- **-newkey rsa:2048** - Simultaneously generates a new certificate and new key. Because the key required to sign the certificate was not created in the previous step, it must be created along with the certificate. The **rsa:2048** generates an RSA 2048 bits long.
- **-keyout** - Determines the location of the generated private key file.
- **-out** - Determines the location of the certificate.

After creating a self-signed key and certificate pair with OpenSSL, a series of prompts about your server is presented to correctly embed the information you provided in the certificate.

3. Provide the information requested by the prompts.

   The most important piece of information is the **Common Name**, which is either the server **FQDN** or **your** name. You must enter the domain name associated with your server or your server's public IP address.

   The following is an example of a filled out set of prompts:

```
OutputCountry Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:New York City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Bouncy Castles, Inc.
Organizational Unit Name (eg, section) []:Ministry of Water Slides
Common Name (e.g. server FQDN or YOUR name) []:server_IP_address
Email Address []:admin@your_domain.com
```

   Both files you create are stored in their own subdirectories of the **/etc/ssl** directory.

   Although SQream uses OpenSSL, in addition we recommend creating a strong **Diffie-Hellman** group, used for negotiating **Perfect Forward Secrecy** with clients.

4. Create a strong Diffie-Hellman group:

```
$ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

   Creating a Diffie-Hellman group takes a few minutes, which is stored as the **dhparam.pem** file in the **/etc/ssl/certs** directory. This file can use in the configuration.

2.2.5.5 Configuring NGINX to use SSL

After creating your SSL certificate, you must configure NGINX to use SSL.

The default CentOS NGINX configuration is fairly unstructured, with the default HTTP server block located in the main configuration file. NGINX checks for files ending in **.conf** in the **/etc/nginx/conf.d** directory for additional configuration.

SQream creates a new file in the **/etc/nginx/conf.d** directory to configure a server block. This block serves content using the certificate files we generated. In addition, the default server block can be optionally configured to redirect HTTP requests to HTTPS.

Note: The example on this page uses the IP address **127.0.0.1**, which you should replace with your machine's IP address.

To configure NGINX to use SSL:

1. Create and open a file called **ssl.conf** in the **/etc/nginx/conf.d** directory:

```
$ sudo vi /etc/nginx/conf.d/ssl.conf
```

2. In the file you created in Step 1 above, open a server block:

   1. Listen to **port 443**, which is the TLS/SSL default port.

   2. Set the `server_name` to the server's domain name or IP address you used as the Common Name when generating your certificate.

   3. Use the `ssl_certificate`, `ssl_certificate_key`, and `ssl_dhparam` directives to set the location of the SSL files you generated, as shown in the **/etc/nginx/conf.d/ssl.conf** file below:

```
    upstream ui {
        server 127.0.0.1:8080;
    }
server {
    listen 443 http2 ssl;
    listen [::]:443 http2 ssl;

    server_name nginx.sq.l;

    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

root /usr/share/nginx/html;

#    location / {
#    }

  location / {
        proxy_pass http://ui;
        proxy_set_header           X-Forwarded-Proto https;
        proxy_set_header           X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header           X-Real-IP       $remote_addr;
        proxy_set_header           Host $host;
                add_header                 Front-End-Https   on;
        add_header                 X-Cache-Status $upstream_cache_status;
        proxy_cache                off;
        proxy_cache_revalidate     off;
        proxy_cache_min_uses       1;
        proxy_cache_valid          200 302 1h;
        proxy_cache_valid          404 3s;
        proxy_cache_use_stale      error timeout invalid_header updating http_500␣
↪http_502 http_503 http_504;
        proxy_no_cache             $cookie_nocache $arg_nocache $arg_comment
↪$http_pragma $http_authorization;
        proxy_redirect             default;
        proxy_max_temp_file_size   0;
        proxy_connect_timeout      90;
        proxy_send_timeout         90;
        proxy_read_timeout         90;
        proxy_buffer_size          4k;
        proxy_buffering            on;
        proxy_buffers              4 32k;
        proxy_busy_buffers_size    64k;
        proxy_temp_file_write_size 64k;
        proxy_intercept_errors     on;

        proxy_set_header           Upgrade $http_upgrade;
        proxy_set_header           Connection "upgrade";
    }

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
```

(continues on next page)

(continued from previous page)

```
    }
}
```

4. Open and modify the **nginx.conf** file located in the **/etc/nginx/conf.d** directory as follows:

```
$ sudo vi /etc/nginx/conf.d/nginx.conf
```

```
server {
    listen       80;
    listen       [::]:80;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

2.2.5.6 Redirecting Studio Access from HTTP to HTTPS

After configuring NGINX to use SSL, you must redirect Studio access from HTTP to HTTPS.

According to your current configuration, NGINX responds with encrypted content for requests on port 443, but with **unencrypted** content for requests on **port 80**. This means that our site offers encryption, but does not enforce its usage. This may be fine for some use cases, but it is usually better to require encryption. This is especially important when confidential data like passwords may be transferred between the browser and the server.

The default NGINX configuration file allows us to easily add directives to the default port 80 server block by adding files in the /etc/nginx/default.d directory.

To create a redirect from HTTP to HTTPS:

1. Create a new file called **ssl-redirect.conf** and open it for editing:

```
$ sudo vi /etc/nginx/default.d/ssl-redirect.conf
```

2. Copy and paste this line:

```
$ return 301 https://$host$request_uri:8080/;
```

2.2.5.7 Activating Your NGINX Configuration

After redirecting from HTTP to HTTPs, you must restart NGINX to activate your new configuration.

To activate your NGINX configuration:

1. Verify that your files contain no syntax errors:

```
$ sudo nginx -t
```

The following output is generated if your files contain no syntax errors:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

2. Restart NGINX to activate your configuration:

```
$ sudo systemctl restart nginx
```

2.2.5.8 Verifying that NGINX is Running

After activating your NGINX configuration, you must verify that NGINX is running correctly.

To verify that NGINX is running correctly:

1. Check that the service is up and running:

```
$ systemctl status nginx
```

The following is an example of the correct output:

```
Output● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor␣
↪preset: disabled)
   Active: active (running) since Fri 2017-01-06 17:27:50 UTC; 28s ago

. . .

Jan 06 17:27:50 centos-512mb-nyc3-01 systemd[1]: Started The nginx HTTP and␣
↪reverse proxy server.
```

2. Run the following command:

```
$ sudo netstat -nltp |grep nginx
```

The following is an example of the correct output:

```
[sqream@dorb-pc etc]$ sudo netstat -nltp |grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      ␣
↪15486/nginx: master
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      ␣
↪15486/nginx: master
tcp6       0      0 :::80                   :::*                    LISTEN      ␣
```

(continues on next page)

(continued from previous page)

```
↪15486/nginx: master
tcp6       0      0 :::443                  :::*                    LISTEN      ␣
↪15486/nginx: master
```

CHAPTER

THREE

UPGRADE GUIDES

Refer to the *Version Upgrade* guide to upgrade from your current SQreamDB version and explore the *Upgrade-Related Configuration Changes* guide. It provides a breakdown of the necessary system modifications for the specific version you're upgrading to, ensuring a thorough and effective upgrade process.

3.1 Version Upgrade

Upgrading your SQreamDB version requires stopping all running services.

1. Stop all actively running SQreamDB services.

   Ensuring that SQreamDB services are at a halt depends on the tool being used.

2. Verify that SQreamDB has stopped listening on ports **500X**, **510X**, and **310X**:

```
$ sudo netstat -nltp    #to make sure SQreamDB stopped listening on 500X, 510X and␣
↪310X ports.
```

3. Replace the old SQreamDB version with the new version, such as in the following example:

```
$ cd /home/sqream
$ mkdir tempfolder
$ mv sqream-db-v2021.1.tar.gz tempfolder/
$ cd tempfolder/
$ tar -xf sqream-db-v2021.1.tar.gz
$ sudo mv sqream /usr/local/sqream-db-v2021.1
$ cd /usr/local
$ sudo chown -R sqream:sqream sqream-db-v2021.1
```

4. Remove the symbolic link:

```
$ sudo rm sqream
```

5. Create a new symbolic link named "sqream" pointing to the new version:

```
$ sudo ln -s sqream-db-v2021.1 sqream
```

6. Verify that the symbolic SQreamDB link points to the real folder:

```
$ ls -l

-- Output example:

$ sqream -> sqream-db-v2021.1
```

7. Upgrade your version of SQreamDB storage.

   a. SQreamDB recommends storing the generated back-up locally in case needed. To generate a back-up of the metadata, run the following command:

```
$ select backup_metadata('out_path');
```

   SQreamDB runs the Garbage Collector and creates a clean backup tarball package.

   b. Shut down all SQreamDB services.

   c. Extract the recently created back-up file.

   d. Replace your current metadata with the metadata you stored in the back-up file.

   e. Navigate to the new SQreamDB package bin folder.

   f. Get the cluster path

```
$ cat /etc/sqream/sqream1_config.json |grep cluster
```

   g. Run the following command:

```
$ ./upgrade_storage <RocksDB path>

-- Output example:

    get_leveldb_version path{<cluster path>}
    current storage version 23
upgrade_v24
upgrade_storage to 24
    upgrade_storage to 24 - Done
    upgrade_v25
    upgrade_storage to 25
    upgrade_storage to 25 - Done
    upgrade_v26
    upgrade_storage to 26
    upgrade_storage to 26 - Done
    validate_leveldb
    ...
upgrade_v37
    upgrade_storage to 37
    upgrade_storage to 37 - Done
    validate_leveldb
storage has been upgraded successfully to version 37
```

8. Verify that the latest version has been installed:

```
$ ./sqream sql --username sqream --password sqream --host localhost --databasename␣
↪master -c "SELECT SHOW_VERSION();"

-- Output example:

v2021.1
1 row
time: 0.050603s
```

For more information, see the *upgrade_storage* command line program.

9. After completing the upgrade process, ensure that ALL *operational and configuration* changes introduced in versions newer than the version you are upgrading from are applied before returning to regular SQreamDB operations.

3.2 Upgrade-Related Configuration Changes

SQreamDB Version	Storage Version	Configurations and Changes
4.4	49	**New Releases:** • Pysqream 5.0.0 Connector is released • JDBC 5.0.0 Connector is released
4.3	49	**Configuration:** • Two new *AWS S3* object access style and endpoint URL with Virtual Private Cloud (VPC) configuration flags: `AwsEndpointOverride`, `AwsObjectAccessStyle` • **REHL 8.x** is now officially supported
4.2	46	**New Releases:** • Pysqream 3.2.5 Connector is released • ODBC 4.4.4 Connector is released • JDBC 4.5.8 Connector is released • Apache Spark 5.0.0 Connector is released • The `INT96` data type is deprecated Configuration: • *Access control permissions* in SQreamDB have been expanded. Learn how to reconfigure access control permissions when *upgrading from version 4.2*
4.1	45	**New Releases:** • JDBC 4.5.7 Connector • SQream Studio v5.5.4
4.0	45	None
2022.1.7	43	None
2022.1.6	42	None
2022.1.5	42	None
2022.1.4	42	None
2022.1.3	42	The `VARCHAR` data type has been deprecated and replaced with `TEXT`.
2022.1.2	41	None
2022.1.1	40	• In compliance with GDPR standards, version 2022.1.1 re-

CHAPTER

FOUR

OPERATIONAL GUIDES

The **Operational Guides** section describes processes that SQream users can manage to affect the way their system operates, such as creating storage clusters and monitoring query performance.

This section summarizes the following operational guides:

4.1 Access Control

4.1.1 Overview

Access control refers to SQream's authentication and authorization operations, managed using a **Role-Based Access Control (RBAC)** system, such as ANSI SQL or other SQL products. SQream's default permissions system is similar to Postgres, but is more powerful. SQream's method lets administrators prepare the system to automatically provide objects with their required permissions.

SQream users can log in from any worker, which verify their roles and permissions from the metadata server. Each statement issues commands as the role that you're currently logged into. Roles are defined at the cluster level, and are valid for all databases in the cluster. To bootstrap SQream, new installations require one `SUPERUSER` role, typically named `sqream`. You can only create new roles by connecting as this role.

Access control refers to the following basic concepts:

- **Role** - A role can be a user, a group, or both. Roles can own database objects (such as tables) and can assign permissions on those objects to other roles. Roles can be members of other roles, meaning a user role can inherit permissions from its parent role.

- **Authentication** - Verifies the identity of the role. User roles have usernames (or **role names**) and passwords.

- **Authorization** - Checks that a role has permissions to perform a particular operation, such as the grant command.

4.1.2 Password Policy

The **Password Policy** describes the following:

- *Password Strength Requirements*
- *Brute Force Prevention*

4.1.2.1 Password Strength Requirements

As part of our compliance with GDPR standards SQream relies on a strong password policy when accessing the CLI or Studio, with the following requirements:

- At least eight characters long.
- Mandatory upper and lowercase letters.
- At least one numeric character.
- May not include a username.
- Must include at least one special character, such as **?**, **!**, **$**, etc.

You can create a password by using the Studio graphic interface or using the CLI, as in the following example command:

```
CREATE ROLE user_a ;
GRANT LOGIN to user_a ;
GRANT PASSWORD 'BBAu47?fqPL' to user_a ;
```

Creating a password which does not comply with the password policy generates an error message with a request to include any of the missing above requirements:

```
The password you attempted to create does not comply with SQream's security␣
↪requirements.

Your password must:

* Be at least eight characters long.

* Contain upper and lowercase letters.

* Contain at least one numeric character.

* Not include a username.

* Include at least one special character, such as **?**, **!**, **$**, etc.
```

4.1.2.2 Brute Force Prevention

Unsuccessfully attempting to log in five times displays the following message:

```
The user is locked. Please contact your system administrator to reset the password␣
↪and regain access functionality.
```

You must have superuser permissions to release a locked user to grant a new password:

```
GRANT PASSWORD '<password>' to <blocked_user>;
```

For more information, see login_max_retries.

> **Warning:** Because superusers can also be blocked, **you must have** at least two superusers per cluster.

4.1.3 Managing Roles

Roles are used for both users and groups, and are global across all databases in the SQream cluster. For a `ROLE` to be used as a user, it requires a password and log-in and connect permissionss to the relevant databases.

The Managing Roles section describes the following role-related operations:

- *Creating New Roles (Users)*
- *Dropping a User*
- *Altering a User Name*
- *Changing a User Password*
- *Altering Public Role Permissions*
- *Altering Role Membership (Groups)*

4.1.3.1 Creating New Roles (Users)

A user role logging in to the database requires `LOGIN` permissions and a password.

The following is the syntax for creating a new role:

```
CREATE ROLE <role_name> ;
GRANT LOGIN to <role_name> ;
GRANT PASSWORD <'new_password'> to <role_name> ;
GRANT CONNECT ON DATABASE <database_name> to <role_name> ;
```

The following is an example of creating a new role:

```
CREATE  ROLE  new_role_name  ;
GRANT  LOGIN  TO  new_role_name;
GRANT  PASSWORD  'Passw0rd!' to new_role_name;
GRANT  CONNECT  ON  DATABASE  master to new_role_name;
```

A database role may have a number of permissions that define what tasks it can perform, which are assigned using the grant command.

4.1.3.2 Dropping a User

The following is the syntax for dropping a user:

```
DROP ROLE <role_name> ;
```

The following is an example of dropping a user:

```
DROP ROLE  admin_role ;
```

4.1.3.3 Altering a User Name

The following is the syntax for altering a user name:

```
ALTER ROLE <role_name> RENAME TO <new_role_name> ;
```

The following is an example of altering a user name:

```
ALTER ROLE admin_role RENAME TO copy_role ;
```

4.1.3.4 Changing a User Password

You can change a user role's password by granting the user a new password.

The following is an example of changing a user password:

```
GRANT  PASSWORD  <'new_password'>  TO  rhendricks;
```

Note: Granting a new password overrides any previous password. Changing the password while the role has an active running statement does not affect that statement, but will affect subsequent statements.

4.1.3.5 Altering Public Role Permissions

The database has a predefined `PUBLIC` role that cannot be deleted. Each user role is automatically granted membership in the `PUBLIC` role public group, and this membership cannot be revoked. However, you have the capability to adjust the permissions associated with this `PUBLIC` role.

The `PUBLIC` role has `USAGE` and `CREATE` permissions on `PUBLIC` schema by default, therefore, newly created user roles are granted `CREATE` (databases, schemas, roles, functions, views, and tables) on the public schema. Other permissions, such as insert, delete, select, and update on objects in the public schema are not automatically granted.

4.1.3.6 Altering Role Membership (Groups)

Many database administrators find it useful to group user roles together. By grouping users, permissions can be granted to, or revoked from a group with one command. In SQream DB, this is done by creating a group role, granting permissions to it, and then assigning users to that group role.

To use a role purely as a group, omit granting it `LOGIN` and `PASSWORD` permissions.

The `CONNECT` permission can be given directly to user roles, and/or to the groups they are part of.

```
CREATE ROLE my_group;
```

Once the group role exists, you can add user roles (members) using the `GRANT` command. For example:

```
-- Add my_user to this group
GRANT my_group TO my_user;
```

To manage object permissions like databases and tables, you would then grant permissions to the group-level role (see the permissions table below.

All member roles then inherit the permissions from the group. For example:

```
-- Grant all group users connect permissions
GRANT  CONNECT  ON  DATABASE  a_database  TO  my_group;

-- Grant all permissions on tables in public schema
GRANT  ALL  ON  all  tables  IN  schema  public  TO  my_group;
```

Removing users and permissions can be done with the `REVOKE` command:

```
-- remove my_other_user from this group
REVOKE my_group FROM my_other_user;
```

4.1.4 Permissions

SQreamDB's primary permission object is a role. The role operates in a dual capacity as both a user and a group. As a user, a role may have permissions to execute operations like creating tables, querying data, and administering the database. The group attribute may be thought of as a membership. As a group, a role may extend its permissions to other roles defined as its group members. This becomes handy when privileged roles wish to extend their permissions and grant multiple permissions to multiple roles. The information about all system role permissions is stored in the metadata.

There are two types of permissions: global and object-level. Global permissions belong to `SUPERUSER` roles, allowing unrestricted access to all system and database activities. Object-level permissions apply to non-`SUPERUSER` roles and can be assigned to databases, schemas, tables, functions, views, foreign tables, catalogs, and services.

The following table describe the required permissions for performing and executing operations on various SQreamDB objects.

Permission	Description
All Databases	
`LOGIN`	Use role to log into the system (the role also needs connect permission on the
`PASSWORD`	The password used for logging into the system
`SUPERUSER`	No permission restrictions on any activity
Database	
`SUPERUSER`	No permission restrictions on any activity within that database (this does not

Table 1 – continued from previous page

Permission	Description
CONNECT	Connect to the database
CREATE	Create schemas in the database
CREATEFUNCTION	Create and drop functions
DDL	Drop and alter tables within the database
ALL	All database permissions except for a SUPERUSER permission
Schema	
USAGE	Grants access to schema objects
CREATE	Create tables in the schema
SUPERUSER	No permission restrictions on any activity within the schema (this does not in
DDL	Drop and alter tables within the schema
ALL	All schema permissions
Table	
SELECT	select from the table
INSERT	insert into the table
UPDATE	update the value of certain columns in existing rows
DELETE	delete and truncate on the table
DDL	Drop and alter on the table
ALL	All table permissions
Function	
EXECUTE	Use the function
DDL	Drop and alter on the function
ALL	All function permissions
View	
SELECT	Select from view
DDL	DDL operations of view results
ALL	All views permissions
Foreign Table	
SELECT	Select from foreign table
DDL	Foreign table DDL operations
ALL	All foreign table permissions
Catalog	
SELECT	Select from catalog
ALL	All catalog permissions
Services	
USAGE	Using a specific service
ALL	All services permissions
Saved Query	
SELECT	Executing saved query statements and utility functions
DDL	Saved query DDL operations
USAGE	Grants access to saved query objects
ALL	All saved query permissions

4.1.4.1 Syntax

Permissions may be granted or revoked using the following syntax.

4.1.4.1.1 GRANT

```
-- Grant permissions to all databases:
GRANT {
SUPERUSER
| LOGIN
| PASSWORD '<password>' }
TO <role> [, ...]

-- Grant permissions at the database level:
GRANT {
CREATE
| CONNECT
| DDL
| SUPERUSER
| CREATE FUNCTION } [, ...]
| ALL [PERMISSIONS]
ON DATABASE <database> [, ...]
TO <role> [, ...]

-- Grant permissions at the schema level:
GRANT {
CREATE
| DDL
| USAGE
| SUPERUSER } [, ...]
| ALL [PERMISSIONS]
ON SCHEMA <schema> [, ...]
TO <role> [, ...]

-- Grant permissions at the object level:
GRANT {
SELECT
| INSERT
| DELETE
| DDL
| UPDATE } [, ...]
| ALL [PERMISSIONS]
ON {TABLE <table_name> [, ...]
| ALL TABLES IN SCHEMA <schema_name> [, ...]}
TO <role> [, ...]

-- Grant permissions at the catalog level:
GRANT {
{SELECT } [, ...]
| ALL [PERMISSIONS] }
ON { CATALOG <catalog_name> [, ...] }
TO <role> [, ...]

-- Grant permissions on the foreign table level:

GRANT {
```

(continues on next page)

(continued from previous page)

```
{SELECT
| DDL } [, ...]
| ALL [PERMISSIONS] }
ON { FOREIGN TABLE <table_name> [, ...]
| ALL FOREIGN TABLE IN SCHEMA <schema_name> [, ...]}
TO <role> [, ...]

-- Grant function execution permission:
GRANT {
ALL
| EXECUTE
| DDL }
ON FUNCTION <function_name>
TO <role>

-- Grant permissions on the view level
GRANT {
{SELECT
| DDL } [, ...]
| ALL [PERMISSIONS] }
ON { VIEW <view_name> [, ...]
| ALL VIEWS IN SCHEMA <schema_name> [, ...]}
TO <role> [, ...]

-- Grant permissions at the Service level:
GRANT {
{USAGE} [, ...]
| ALL [PERMISSIONS] }
ON { SERVICE <service_name> [, ...]
| ALL SERVICES IN SYSTEM }
TO <role> [, ...]

-- Grant saved query permissions
GRANT
SELECT
| DDL
| USAGE
| ALL
ON SAVED QUERY <saved_query> [,...]
TO <role> [,...]

-- Allows role2 to use permissions granted to role1
GRANT <role1> [, ...]
TO <role2>

-- Also allows the role2 to grant role1 to other roles:
GRANT <role1> [, ...]
TO <role2> [,...] [WITH ADMIN OPTION]
```

4.1.4.1.2 REVOKE

```
-- Revoke permissions from all databases:
REVOKE {
SUPERUSER
| LOGIN
| PASSWORD '<password>' }
FROM <role> [, ...]

-- Revoke permissions at the database level:
REVOKE {
CREATE
| CONNECT
| DDL
| SUPERUSER
| CREATE FUNCTION } [, ...]
| ALL [PERMISSIONS]
ON DATABASE <database> [, ...]
FROM <role> [, ...]

-- Revoke permissions at the schema level:
REVOKE {
CREATE
| DDL
| USAGE
| SUPERUSER } [, ...]
| ALL [PERMISSIONS]
ON SCHEMA <schema> [, ...]
FROM <role> [, ...]

-- Revoke permissions at the object level:
REVOKE {
SELECT
| INSERT
| DELETE
| DDL
| UPDATE } [, ...]
| ALL [PERMISSIONS]
ON {TABLE <table_name> [, ...]
| ALL TABLES IN SCHEMA <schema_name> [, ...]}
FROM <role> [, ...]

-- Revoke permissions at the catalog level:
REVOKE {
{SELECT } [, ...]
| ALL [PERMISSIONS] }
ON { CATALOG <catalog_name> [, ...] }
FROM <role> [, ...]

-- Revoke permissions on the foreign table level:

REVOKE {
{SELECT
| DDL } [, ...]
| ALL [PERMISSIONS] }
ON { FOREIGN TABLE <table_name> [, ...]
| ALL FOREIGN TABLE IN SCHEMA <schema_name> [, ...]}
```

(continues on next page)

(continued from previous page)

```
FROM <role> [, ...]

-- Revoke function execution permission:
REVOKE {
ALL
| EXECUTE
| DDL }
ON FUNCTION <function_name>
FROM <role>

-- Revoke permissions on the view level
REVOKE {
{SELECT
| DDL } [, ...]
| ALL [PERMISSIONS] }
ON { VIEW <view_name> [, ...]
| ALL VIEWS IN SCHEMA <schema_name> [, ...]}
FROM <role> [, ...]

-- Revoke permissions at the Service level:
REVOKE {
{USAGE} [, ...]
| ALL [PERMISSIONS] }
ON { SERVICE <service_name> [, ...]
| ALL SERVICES IN SYSTEM }
FROM <role> [, ...]

-- Revoke saved query permissions
REVOKE
SELECT
| DDL
| USAGE
| ALL
ON SAVED QUERY <saved_query> [,...]
FROM <role> [,...]

-- Removes access to permissions in role1 by role 2
REVOKE [ADMIN OPTION FOR] <role1> [, ...]
FROM <role2> [, ...]

-- Removes permissions to grant role1 to additional roles from role2
REVOKE [ADMIN OPTION FOR] <role1> [, ...]
FROM <role2> [, ...]
```

4.1.4.1.3 Altering Default Permissions

The default permissions system (See alter_default_permissions) can be used to automatically grant permissions to newly created objects (See the departmental example below for one way it can be used).

A default permissions rule looks for a schema being created, or a table (possibly by schema), and is table to grant any permission to that object to any role. This happens when the create table or create schema statement is run.

```
ALTER DEFAULT PERMISSIONS FOR modifying_role
[IN <schema_name> [, ...]
FOR {
```

(continues on next page)

(continued from previous page)

```
     SCHEMAS
     | TABLES
     | FOREIGN TABLES
     | VIEWS
     | CATALOGS
     | SERVICES
     | SAVED_QUERIES
    }
     { grant_clause
     | DROP grant_clause }
     TO ROLE { role_name | public
            }

grant_clause ::=
GRANT
   { CREATE FUNCTION
    | SUPERUSER
    | CONNECT
    | CREATE
    | USAGE
    | SELECT
    | INSERT
    | DELETE
    | DDL
    | UPDATE
    | EXECUTE
    | ALL
   }
```

4.1.4.2 Examples

4.1.4.2.1 GRANT

Grant superuser privileges and login capability to a role:

```
GRANT SUPERUSER, LOGIN TO role_name;
```

Grant specific permissions on a database to a role:

```
GRANT CREATE, CONNECT, DDL, SUPERUSER, CREATE FUNCTION ON DATABASE database_name TO␣
↪role_name;
```

Grant various permissions on a schema to a role:

```
GRANT CREATE, USAGE, SUPERUSER ON SCHEMA schema_name TO role_name;
```

Grant permissions on specific objects (table, view, foreign table, or catalog) to a role:

```
GRANT SELECT, INSERT, DELETE, DDL, UPDATE ON TABLE schema_name.table_name TO role_
↪name;
```

Grant execute function permission to a role:

```
GRANT EXECUTE ON FUNCTION function_name TO role_name;
```

Grant usage permissions on a service to a role:

```
GRANT USAGE ON SERVICE service_name TO role_name;
```

Grant role2 the ability to use permissions granted to role1:

```
GRANT role1 TO role2;
```

Grant role2 the ability to grant role1 to other roles:

```
GRANT role1 TO role2 WITH ADMIN OPTION;
```

4.1.4.2.2 REVOKE

Revoke superuser privileges or login capability from a role:

```
REVOKE SUPERUSER, LOGIN FROM role_name;
```

Revoke specific permissions on a database from a role:

```
REVOKE CREATE, CONNECT, DDL, SUPERUSER, CREATE FUNCTION ON DATABASE database_name␣
↪FROM role_name;
```

Revoke permissions on a schema from a role:

```
REVOKE CREATE, USAGE, SUPERUSER ON SCHEMA schema_name FROM role_name;
```

Revoke permissions on specific objects (table, view, foreign table, or catalog) from a role:

```
REVOKE SELECT, INSERT, DELETE, DDL, UPDATE ON TABLE schema_name.table_name FROM role_
↪name;
```

Revoke usage permissions on a service from a role:

```
REVOKE USAGE ON SERVICE service_name FROM role_name;
```

Remove access to permissions in role1 by role2:

```
REVOKE role1 FROM role2 ;
```

Remove permissions to grant role1 to additional roles from role2:

```
REVOKE ADMIN OPTION FOR role1 FROM role2 ;
```

4.1.5 Departmental Example

You work in a company with several departments.

The example below shows you how to manage permissions in a database shared by multiple departments, where each department has different roles for the tables by schema. It walks you through how to set the permissions up for existing objects and how to set up default permissions rules to cover newly created objects.

The concept is that you set up roles for each new schema with the correct permissions, then the existing users can use these roles.

A superuser must do new setup for each new schema which is a limitation, but superuser permissions are not needed at any other time, and neither are explicit grant statements or object ownership changes.

In the example, the database is called `my_database`, and the new or existing schema being set up to be managed in this way is called `my_schema`.

Our departmental example has four user group roles and seven users roles

There will be a group for this schema for each of the following:

Group	Activities
database designers	create, alter and drop tables
updaters	insert and delete data
readers	read data
security officers	add and remove users from these groups

4.1.5.1 Setting up the department permissions

As a superuser, you connect to the system and run the following:

```
-- create the groups

CREATE ROLE my_schema_security_officers;
CREATE ROLE my_schema_database_designers;
CREATE ROLE my_schema_updaters;
CREATE ROLE my_schema_readers;

-- grant permissions for each role
-- we grant permissions for existing objects here too,
-- so you don't have to start with an empty schema

-- security officers

GRANT connect ON DATABASE my_database TO my_schema_security_officers;
GRANT usage ON SCHEMA my_schema TO my_schema_security_officers;

GRANT my_schema_database_designers TO my_schema_security_officers WITH ADMIN OPTION;
GRANT my_schema_updaters TO my_schema_security_officers WITH ADMIN OPTION;
GRANT my_schema_readers TO my_schema_security_officers WITH ADMIN OPTION;

-- database designers

GRANT connect ON DATABASE my_database TO my_schema_database_designers;
GRANT usage ON SCHEMA my_schema TO my_schema_database_designers;

GRANT create,ddl ON SCHEMA my_schema TO my_schema_database_designers;

-- updaters

GRANT connect ON DATABASE my_database TO my_schema_updaters;
GRANT usage ON SCHEMA my_schema TO my_schema_updaters;

GRANT SELECT,INSERT,DELETE ON ALL TABLES IN SCHEMA my_schema TO my_schema_updaters;

-- readers
```

(continues on next page)

(continued from previous page)

```
GRANT connect ON DATABASE my_database TO my_schema_readers;
GRANT usage ON SCHEMA my_schema TO my_schema_readers;

GRANT SELECT ON ALL TABLES IN SCHEMA my_schema TO my_schema_readers;
GRANT EXECUTE ON ALL FUNCTIONS TO my_schema_readers;


-- create the default permissions for new objects

ALTER DEFAULT PERMISSIONS FOR my_schema_database_designers IN my_schema
 FOR TABLES GRANT SELECT,INSERT,DELETE TO my_schema_updaters;

-- For every table created by my_schema_database_designers, give access to my_schema_
↪readers:

ALTER DEFAULT PERMISSIONS FOR my_schema_database_designers IN my_schema
 FOR TABLES GRANT SELECT TO my_schema_readers;
```

Note:

- This process needs to be repeated by a user with `SUPERUSER` permissions each time a new schema is brought into this permissions management approach.
- By default, any new object created will not be accessible by our new `my_schema_readers` group. Running a `GRANT SELECT ...` only affects objects that already exist in the schema or database.

  If you're getting a `Missing the following permissions: SELECT on table 'database.public.tablename'` error, make sure that you've altered the default permissions with the `ALTER DEFAULT PERMISSIONS` statement.

4.1.5.2 Creating new users in the departments

After the group roles have been created, you can now create user roles for each of your users.

```
-- create the new database designer users

CREATE  ROLE  ecodd;
GRANT  LOGIN  TO  ecodd;
GRANT  PASSWORD  'Passw0rd!'  TO ecodd;
GRANT  CONNECT  ON  DATABASE  my_database  TO  ecodd;
GRANT my_schema_database_designers TO ecodd;

CREATE  ROLE  ebachmann;
GRANT  LOGIN  TO  ebachmann;
GRANT  PASSWORD  'Passw0rd!!!'  TO ebachmann;
GRANT  CONNECT  ON  DATABASE  my_database  TO  ebachmann;
GRANT my_database_designers TO ebachmann;

-- If a user already exists, we can assign that user directly to the group

GRANT my_schema_updaters TO rhendricks;

-- Create users in the readers group
```

(continues on next page)

(continued from previous page)

```
CREATE  ROLE  jbarker;
GRANT  LOGIN  TO  jbarker;
GRANT  PASSWORD  'action_jacC%k'  TO jbarker;
GRANT  CONNECT  ON  DATABASE  my_database  TO  jbarker;
GRANT my_schema_readers TO jbarker;

CREATE  ROLE  lbream;
GRANT  LOGIN  TO  lbream;
GRANT  PASSWORD  'artichoke1230$'  TO lbream;
GRANT  CONNECT  ON  DATABASE  my_database  TO  lbream;
GRANT my_schema_readers TO lbream;

CREATE  ROLE  pgregory;
GRANT  LOGIN  TO  pgregory;
GRANT  PASSWORD  'c1ca6aG$'  TO pgregory;
GRANT  CONNECT  ON  DATABASE  my_database  TO  pgregory;
GRANT my_schema_readers TO pgregory;

-- Create users in the security officers group

CREATE  ROLE  hoover;
GRANT  LOGIN  TO  hoover;
GRANT  PASSWORD  'mint*Rchip'  TO hoover;
GRANT  CONNECT  ON  DATABASE  my_database  TO  hoover;
GRANT my_schema_security_officers TO hoover;
```

After this setup:

- Database designers will be able to run any ddl on objects in the schema and create new objects, including ones created by other database designers
- Updaters will be able to insert and delete to existing and new tables
- Readers will be able to read from existing and new tables

All this will happen without having to run any more `GRANT` statements.

Any security officer will be able to add and remove users from these groups. Creating and dropping login users themselves must be done by a superuser.

4.2 Creating or Cloning Storage Clusters

When SQream DB is installed, it comes with a default storage cluster. This guide will help if you need a fresh storage cluster or a separate copy of an existing storage cluster.

4.2.1 Creating a new storage cluster

SQream DB comes with a CLI tool, *SqreamStorage*. This tool can be used to create a new empty storage cluster.

In this example, we will create a new cluster at `/home/rhendricks/raviga_database`:

```
$ SqreamStorage --create-cluster --cluster-root /home/rhendricks/raviga_database
Setting cluster version to: 26
```

This can also be written shorthand as `SqreamStorage -C -r /home/rhendricks/raviga_database`.

This `Setting cluster version...` message confirms the creation of the cluster successfully.

4.2.2 Tell SQream DB to use this storage cluster

4.2.2.1 Permanently setting the storage cluster setting

To permanently set the new cluster location, change the `"cluster"` path listed in the configuration file.

For example:

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/sqream/my_old_cluster",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

should be changed to

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/rhendricks/raviga_database",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

Now, the cluster should be restarted for the changes to take effect.

4.2.2.2 Start a temporary SQream DB worker with a storage cluster

Starting a SQream DB worker with a custom cluster path can be done in two ways:

4.2.2.2.1 Using a configuration file (recommended)

Similar to the technique above, create a configuration file with the correct cluster path. Then, start `sqreamd` using the `-config` flag:

```
$ sqreamd -config config_file.json
```

4.2.2.2.2 Using the command line parameters

Use sqreamd's command line parameters to override the default storage cluster path:

```
$ sqreamd /home/rhendricks/raviga_database 0 5000 /home/sqream/.sqream/license.enc
```

Note: sqreamd's command line parameters' order is `sqreamd <cluster path> <GPU ordinal> <TCP listen port (unsecured)> <License path>`

4.2.3 Copying an existing storage cluster

Copying an existing storage cluster to another path may be useful for testing or troubleshooting purposes.

1. Identify the location of the active storage cluster. This path can be found in the configuration file, under the `"cluster"` parameter.
2. Shut down the SQream DB cluster. This prevents very large storage directories from being modified during the copy process.
3. (optional) Create a tarball of the storage cluster, with `tar -zcvf sqream_cluster_`date +"%Y-%m-%d-%H-%M"`.tgz <cluster path>`. This will create a tarball with the current date and time as part of the filename.
4. Copy the storage cluster directory (or tarball) with `cp` to another location on the local filesystem, or use `rsync` to copy to a remote server.
5. After the copy is completed, start the SQream DB cluster to continue using SQream DB.

4.3 Working with External Data

SQream supports the following external data sources:

4.3.1 Inserting Data Using Amazon S3

SQream uses a native S3 connector for directly inserting data from a number of external sources directly into SQream. This is done using the `s3://` URI to specify an external file path on an S3 bucket. Your files can be saved in CSV or columnar format, such as Parquet and ORC, and your file names can include wildcard characters.

The **Amazon S3** page describes the following topics:

- *Configuring Amazon S3*
- *Setting the S3 URI Format*
- *Authenticating Users*
- *Examples*

4.3.1.1 Configuring Amazon S3

Any database host with access to S3 endpoints can access S3 without any configuration. To read files from an S3 bucket, the database must have listable files.

4.3.1.2 Setting the S3 URI Format

With S3, specify a location for a file (or files) when using copy_from or external_tables.

The following is an example of the general S3 syntax:

```
s3://bucket_name/path
```

4.3.1.3 Authenticating Users

SQream supports `AWS ID` and `AWS SECRET` authentication. These should be specified when executing a statement.

4.3.1.4 Examples

You can use a foreign table to stage data from S3 before loading from CSV, Parquet, or ORC files.

This section includes the following examples:

- *Planning for Data Staging*
- *Creating a Foreign Table*
- *Querying Foreign Tables*
- *Bulk Loading a File from a Public S3 Bucket*
- *Loading Files from an Authenticated S3 Bucket*

4.3.1.4.1 Planning for Data Staging

The examples in this section are based on the CSV file shown in the following table:

Table 2: nba-t10

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	
R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0		5000000.0
Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0		12000000.0
Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0

This CSV file is stored on Amazon S3, and this bucket is public and listable. To create a matching `CREATE FOREIGN TABLE` statement you can make a record of your source file's structure and use it to reproduce a corresponding foreign table, as shown in the following section.

4.3.1.4.2 Creating a Foreign Table

Based on the source file's structure above, you can create a foreign table with the structure you want and point it to your file, as shown in the following example:

```
CREATE FOREIGN TABLE nba
(
   Name text(40),
   Team text(40),
   Number tinyint,
   Position text(2),
   Age tinyint,
   Height text(4),
   Weight real,
   College text(40),
   Salary float
 )
 WRAPPER csv_fdw
 OPTIONS
   (
      LOCATION = 's3://sqream-demo-data/nba_players.csv',
      RECORD_DELIMITER = '\r\n' -- DOS delimited file
```

(continues on next page)

(continued from previous page)

```
    )
;
```

Note: In the example above the file format is CSV and is stored as an S3 object. If your file has an HDFS path, you must change the URI accordingly. Note that the record delimiter is a DOS newline (`\r\n`).

For more information, see the following:

- **Creating a foreign table** - see creating a foreign table.
- **Using SQream in an HDFS environment** - see *Using SQream in an HDFS Environment*.

4.3.1.4.3 Querying Foreign Tables

The following shows the data located in the foreign table:

```
t=> SELECT * FROM nba LIMIT 10;
name           | team           | number | position | age | height | weight | college
↪          | salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma
↪State     |  3431040
```

4.3.1.4.4 Bulk Loading a File from a Public S3 Bucket

Youc an use the `COPY FROM` command to load data without staging it first.

Note: The bucket must be publicly available and objects can be listed.

The following is an example of bulk loading a file from a public S3 bucket:

```
COPY nba FROM 's3://sqream-demo-data/nba.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n';
```

For more information on the `COPY FROM` command, see copy_from.

4.3.1.4.5 Loading Files from an Authenticated S3 Bucket

The following is an example of loading fles from an authenticated S3 bucket:

```
COPY nba FROM 's3://secret-bucket/*.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n'
AWS_ID '12345678'
AWS_SECRET 'super_secretive_secret';
```

4.3.2 Using SQream in an HDFS Environment

4.3.2.1 Configuring an HDFS Environment for the User sqream

This section describes how to configure an HDFS environment for the user **sqream** and is only relevant for users with an HDFS environment.

To configure an HDFS environment for the user sqream:

1. Open your **bash_profile** configuration file for editing:

```
$ vim /home/sqream/.bash_profile
```

2. Verify that the edits have been made:

```
source /home/sqream/.bash_profile
```

3. Check if you can access Hadoop from your machine:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

4. Verify that an HDFS environment exists for SQream services:

```
$ ls -l /etc/sqream/sqream_env.sh
```

5. If an HDFS environment does not exist for SQream services, create one (sqream_env.sh):

```
$ #!/bin/bash

$ SQREAM_HOME=/usr/local/sqream
$ export SQREAM_HOME

$ export JAVA_HOME=${SQREAM_HOME}/hdfs/jdk
$ export HADOOP_INSTALL=${SQREAM_HOME}/hdfs/hadoop
$ export CLASSPATH=`${HADOOP_INSTALL}/bin/hadoop classpath --glob`
$ export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_INSTALL}/lib/native
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${SQREAM_HOME}/lib:$HADOOP_COMMON_LIB_
↪NATIVE_DIR


$ PATH=$PATH:$HOME/.local/bin:$HOME/bin:${SQREAM_HOME}/bin/:${JAVA_HOME}/bin:
↪$HADOOP_INSTALL/bin
$ export PATH
```

Back to top

4.3.2.2 Authenticating Hadoop Servers that Require Kerberos

If your Hadoop server requires Kerberos authentication, do the following:

1. Create a principal for the user **sqream**.

```
$ kadmin -p root/admin@SQ.COM
$ addprinc sqream@SQ.COM
```

2. If you do not know yor Kerberos root credentials, connect to the Kerberos server as a root user with ssh and run **kadmin.local**:

```
$ kadmin.local
```

   Running **kadmin.local** does not require a password.

3. If a password is not required, change your password to **sqream@SQ.COM**.

```
$ change_password sqream@SQ.COM
```

4. Connect to the hadoop name node using ssh:

```
$ cd /var/run/cloudera-scm-agent/process
```

5. Check the most recently modified content of the directory above:

```
$ ls -lrt
```

6. Look for a recently updated folder containing the text **hdfs**.

   The following is an example of the correct folder name:

```
cd <number>-hdfs-<something>
```

   This folder should contain a file named **hdfs.keytab** or another similar .keytab file.

7. Copy the .keytab file to user **sqream's** Home directory on the remote machines that you are planning to use Hadoop on.

8. Copy the following files to the **sqream sqream@server:<sqream folder>/hdfs/hadoop/etc/hadoop:** directory:
   - core-site.xml
   - hdfs-site.xml

9. Connect to the sqream server and verify that the .keytab file's owner is a user sqream and is granted the correct permissions:

```
$ sudo chown sqream:sqream /home/sqream/hdfs.keytab
$ sudo chmod 600 /home/sqream/hdfs.keytab
```

10. Log into the sqream server.

11. Log in as the user **sqream**.

12. Navigate to the Home directory and check the name of a Kerberos principal represented by the following .keytab file:

```
$ klist -kt hdfs.keytab
```

The following is an example of the correct output:

```
$ sqream@Host-121 ~ $ klist -kt hdfs.keytab
$ Keytab name: FILE:hdfs.keytab
$ KVNO Timestamp           Principal
$ ---- ------------------- -----------------------------------------------------
↪----
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
```

13. Verify that the hdfs service named **hdfs/nn1@SQ.COM** is shown in the generated output above.

14. Run the following:

```
$ kinit -kt hdfs.keytab hdfs/nn1@SQ.COM
```

15. Verify that the output is correct:

```
$ klist
```

The following is an example of the correct output:

```
$ Ticket cache: FILE:/tmp/krb5cc_1000
$ Default principal: sqream@SQ.COM
$
$ Valid starting       Expires              Service principal
$ 09/16/2020 13:44:18  09/17/2020 13:44:18  krbtgt/SQ.COM@SQ.COM
```

16. List the files located at the defined server name or IP address:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

17. Do one of the following:

    - If the list below is output, continue with Step 18.

- If the list is not output, verify that your environment has been set up correctly.

If any of the following are empty, verify that you followed *Step 6* in the **Configuring an HDFS Environment for the User sqream** section above correctly:

```
$ echo $JAVA_HOME
$ echo $SQREAM_HOME
$ echo $CLASSPATH
$ echo $HADOOP_COMMON_LIB_NATIVE_DIR
$ echo $LD_LIBRARY_PATH
$ echo $PATH
```

18. Verify that you copied the correct keytab file.

19. Review this procedure to verify that you have followed each step.

Back to top

For more information, see the following:

- *Foreign Tables*
- copy_from
- copy_to

4.4 Foreign Tables

Foreign tables can be used to run queries directly on data without inserting it into SQream DB first. SQream DB supports read-only foreign tables so that you can query from foreign tables, but you cannot insert to them, or run deletes or updates on them.

Running queries directly on foreign data is most effectively used for one-off querying. If you are repeatedly querying data, the performance will usually be better if you insert the data into SQream DB first.

Although foreign tables can be used without inserting data into SQream DB, one of their main use cases is to help with the insertion process. An insert select statement on a foreign table can be used to insert data into SQream using the full power of the query engine to perform ETL.

In this topic:

- *Supported Data Formats*
- *Supported Data Staging*
- *Using Foreign Tables*
- *Error Handling and Limitations*

4.4.1 Supported Data Formats

SQream DB supports foreign tables over:

- Text - CSV, TSV, and PSV
- Parquet
- ORC
- Avro
- JSON

4.4.2 Supported Data Staging

SQream can stage data from:

- a local filesystem (e.g. `/mnt/storage/....`)
- *Inserting Data Using Amazon S3* buckets (e.g. `s3://pp-secret-bucket/users/*.parquet`)
- *Using SQream in an HDFS Environment* (e.g. `hdfs://hadoop-nn.piedpiper.com/rhendricks/*.csv`)

4.4.3 Using Foreign Tables

Use a foreign table to stage data before loading from CSV, Parquet or ORC files.

4.4.3.1 Planning for Data Staging

For the following examples, we will interact with a CSV file.

The file is stored on *Inserting Data Using Amazon S3*, at `s3://sqream-demo-data/nba_players.csv`. We will make note of the file structure, to create a matching `CREATE_EXTERNAL_TABLE` statement.

4.4.3.2 Creating a Foreign Table

Based on the source file structure, we create a foreign table with the appropriate structure, and point it to the file.

```
CREATE foreign table nba
(
   Name varchar,
   Team varchar,
   Number tinyint,
   Position varchar,
   Age tinyint,
   Height varchar,
   Weight real,
   College varchar,
   Salary float
 )
   WRAPPER csv_fdw
   OPTIONS
     ( LOCATION = 's3://sqream-demo-data/nba_players.csv',
```

(continues on next page)

(continued from previous page)

```
        DELIMITER = '\r\n' -- DOS delimited file
    );
```

The file format in this case is CSV, and it is stored as an *Inserting Data Using Amazon S3* object (if the path is on *Using SQream in an HDFS Environment*, change the URI accordingly).

We also took note that the record delimiter was a DOS newline (`\r\n`).

4.4.3.3 Querying Foreign Tables

Let's peek at the data from the foreign table:

```
t=> SELECT * FROM nba LIMIT 10;
name           | team           | number | position | age | height | weight | college
↪          | salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma
↪State     |  3431040
```

4.4.3.4 Modifying Data from Staging

One of the main reasons for staging data is to examine the content and modify it before loading. Assume we are unhappy with weight being in pounds because we want to use kilograms instead. We can apply the transformation as part of a query:

```
t=> SELECT name, team, number, position, age, height, (weight / 2.205) as weight,
↪college, salary
.          FROM nba
.          ORDER BY weight;

name                     | team                   | number | position | age | height
↪| weight    | college               | salary
-------------------------+------------------------+--------+----------+-----+--------
↪+----------+-----------------------+---------
Nikola Pekovic           | Minnesota Timberwolves |     14 | C        |  30 | 6-11
```

(continues on next page)

(continued from previous page)

```
↪|  139.229 |                        | 12100000
Boban Marjanovic          | San Antonio Spurs      |     40 | C        |  27 | 7-3   ␣
↪| 131.5193 |                        |  1200000
Al Jefferson              | Charlotte Hornets      |     25 | C        |  31 | 6-10  ␣
↪| 131.0658 |                        | 13500000
Jusuf Nurkic              | Denver Nuggets         |     23 | C        |  21 | 7-0   ␣
↪| 126.9841 |                        |  1842000
Andre Drummond            | Detroit Pistons        |      0 | C        |  22 | 6-11  ␣
↪| 126.5306 | Connecticut            |  3272091
Kevin Seraphin            | New York Knicks        |      1 | C        |  26 | 6-10  ␣
↪| 126.0771 |                        |  2814000
Brook Lopez               | Brooklyn Nets          |     11 | C        |  28 | 7-0   ␣
↪| 124.7166 | Stanford               | 19689000
Jahlil Okafor             | Philadelphia 76ers     |      8 | C        |  20 | 6-11  ␣
↪| 124.7166 | Duke                   |  4582680
Cristiano Felicio         | Chicago Bulls          |      6 | PF       |  23 | 6-10  ␣
↪| 124.7166 |                        |   525093
[...]
```

Now, if we're happy with the results, we can convert the staged foreign table to a standard table

4.4.3.5 Converting a Foreign Table to a Standard Database Table

create_table_as can be used to materialize a foreign table into a regular table.

Tip: If you intend to use the table multiple times, convert the foreign table to a standard table.

```
t=> CREATE TABLE real_nba AS
.    SELECT name, team, number, position, age, height, (weight / 2.205) as weight,␣
↪college, salary
.            FROM nba
.            ORDER BY weight;
executed
t=> SELECT * FROM real_nba LIMIT 5;

name              | team                   | number | position | age | height | weight␣
↪  | college      | salary
-----------------+------------------------+--------+----------+-----+--------+--------
↪--+-------------+---------
Nikola Pekovic    | Minnesota Timberwolves |     14 | C        |  30 | 6-11   |  139.
↪229 |              | 12100000
Boban Marjanovic | San Antonio Spurs      |     40 | C        |  27 | 7-3    | 131.
↪5193 |              |  1200000
Al Jefferson      | Charlotte Hornets      |     25 | C        |  31 | 6-10   | 131.
↪0658 |              | 13500000
Jusuf Nurkic      | Denver Nuggets         |     23 | C        |  21 | 7-0    | 126.
↪9841 |              |  1842000
Andre Drummond    | Detroit Pistons        |      0 | C        |  22 | 6-11   | 126.
↪5306 | Connecticut |  3272091
```

4.4.4 Error Handling and Limitations

- Error handling in foreign tables is limited. Any error that occurs during source data parsing will result in the statement aborting.
- Foreign tables are logical and do not contain any data, their structure is not verified or enforced until a query uses the table. For example, a CSV with the wrong delimiter may cause a query to fail, even though the table has been created successfully:

```
t=> SELECT * FROM nba;
master=> select * from nba;
Record delimiter mismatch during CSV parsing. User defined line delimiter \n does␣
↪not match the first delimiter \r\n found in s3://sqream-demo-data/nba.csv
```

- Since the data for a foreign table is not stored in SQream DB, it can be changed or removed at any time by an external process. As a result, the same query can return different results each time it runs against a foreign table. Similarly, a query might fail if the external data is moved, removed, or has changed structure.

4.5 Deleting Data

When working with a table in a database, deleting data typically involves removing rows, although it can also involve removing columns. The process for deleting data involves first deleting the desired content, followed by a cleanup operation that reclaims the space previously occupied by the deleted data. This process is further explained below.

The `DELETE` statement is used to remove rows that match a specified predicate, thereby preventing them from being included in subsequent queries. For example, the following statement deletes all rows in the `cool_animals` table where the weight of the animal is greater than 1000 weight units:

```
DELETE FROM cool_animals WHERE weight > 1000;
```

By using the WHERE clause in the DELETE statement, you can specify a condition or predicate that determines which rows should be deleted from the table. In this example, the predicate "weight > 1000" specifies that only rows with an animal weight greater than 1000 should be deleted.

- *The Deletion Process*
- *Usage Notes*
- *Examples*
- *Best Practice*

4.5.1 The Deletion Process

When you delete rows from a SQL database, the actual deletion process occurs in two steps:

- **Marking for Deletion**: When you issue a `DELETE` statement to remove one or more rows from a table, the database marks these rows for deletion. These rows are not actually removed from the database immediately, but are instead temporarily ignored when you run any query.

- **Clean-up**: Once the rows have been marked for deletion, you need to trigger a clean-up operation to permanently remove them from the database. During the clean-up process, the database frees up the disk space previously occupied by the deleted rows. To remove all files associated with the deleted rows, you can use the utility function

commands `CLEANUP_CHUNKS` and `CLEANUP_EXTENTS`. These commands should be run sequentially to ensure that these files removed from disk.

If you want to delete all rows from a table, you can use the TRUNCATE command, which deletes all rows in a table and frees up the associated disk space.

4.5.2 Usage Notes

4.5.2.1 General Notes

- The alter_table command and other DDL operations are locked on tables that require clean-up. If the estimated clean-up time exceeds the permitted threshold, an error message is displayed describing how to override the threshold limitation. For more information, see *Concurrency and Locks*.

- If the number of deleted records exceeds the threshold defined by the `mixedColumnChunksThreshold` parameter, the delete operation is aborted. This alerts users that the large number of deleted records may result in a large number of mixed chunks. To circumvent this alert, use the following syntax (replacing `XXX` with the desired number of records) before running the delete operation:

```
set mixedColumnChunksThreshold=XXX;
```

4.5.2.2 Clean-Up Operations Are I/O Intensive

The clean-up process reduces table size by removing all unused space from column chunks. While this reduces query time, it is a time-costly operation occupying disk space for the new copy of the table until the operation is complete.

Tip: Because clean-up operations can create significant I/O load on your database, consider using them sparingly during ideal times.

If this is an issue with your environment, consider using `CREATE TABLE AS` to create a new table and then rename and drop the old table.

4.5.3 Examples

To follow the examples section, create the following table:

```
CREATE OR REPLACE TABLE cool_animals (
     animal_id INT,
     animal_name TEXT,
     animal_weight FLOAT
);
```

Insert the following content:

```
INSERT INTO cool_animals (animal_id, animal_name, animal_weight)
VALUES
(1, 'Dog', 7),
(2, 'Possum', 3),
(3, 'Cat', 5),
(4, 'Elephant', 6500),
```

(continues on next page)

(continued from previous page)

```
(5, 'Rhinoceros', 2100),
(6, NULL, NULL);
```

View table content:

```
farm=> SELECT * FROM cool_animals;

Return:

   animal_id    | animal_name       | animal_weight
   ------------+------------------+--------------------
   1            | Dog               | 7
   2            | Possum            | 3
   3            | Cat               | 5
   4            | Elephant          | 6500
   5            | Rhinoceros        | 2100
   6            | NULL              | NULL
```

Now you may use the following examples for:

- *Deleting Rows from a Table*
- *Deleting Values Based on Complex Predicates*
- *Identifying and Cleaning Up Tables*

4.5.3.1 Deleting Rows from a Table

1. Delete rows from the table:

```
farm=> DELETE FROM cool_animals WHERE animal_weight > 1000;
```

2. Display the table:

```
farm=> SELECT * FROM cool_animals;

Return

animal_id    | animal_name       | animal_weight
------------+------------------+--------------
1            | Dog               | 7
2            | Possum            | 3
3            | Cat               | 5
6            | NULL              | NULL
```

4.5.3.2 Deleting Values Based on Complex Predicates

1. Delete rows from the table:

```
farm=>  DELETE FROM cool_animals
       WHERE animal_weight < 100 AND animal_name LIKE '%o%';
```

2. Display the table:

```
farm=> SELECT * FROM cool_animals;

Return

animal_id    | animal_name       | animal_weight
------------+------------------+--------------------
3            | Cat               | 5
4            | Elephant          | 6500
6            | NULL              | NULL
```

4.5.3.3 Identifying and Cleaning Up Tables

Listing tables that have not been cleaned up:

```
farm=> SELECT t.table_name FROM sqream_catalog.delete_predicates dp
   JOIN sqream_catalog.tables t
   ON dp.table_id = t.table_id
   GROUP BY 1;
cool_animals

1 row
```

Identifying predicates for Clean-Up:

```
farm=> SELECT delete_predicate FROM sqream_catalog.delete_predicates dp
   JOIN sqream_catalog.tables t
   ON dp.table_id = t.table_id
   WHERE t.table_name = 'cool_animals';
weight > 1000

1 row
```

4.5.3.3.1 Triggering a Clean-Up

When running the clean-up operation, you need to specify two parameters: `schema_name` and `table_name`. Note that both parameters are case-sensitive and cannot operate with upper-cased schema or table names.

Running a `CLEANUP_CHUNKS` command (also known as `SWEEP`) to reorganize the chunks:

```
farm=> SELECT CLEANUP_CHUNKS('<schema_name>','<table_name>');
```

Running a `CLEANUP_EXTENTS` command (also known as `VACUUM`) to delete the leftover files:

```
farm=> SELECT CLEANUP_EXTENTS('<schema_name>','<table_name>');
```

If you should want to run a clean-up operation without worrying about uppercase and lowercase letters, you can use the `false` flag to enable lowercase letters for both lowercase and uppercase table and schema names, such as in the following examples:

```
farm=> SELECT CLEANUP_CHUNKS('<schema_name>','<table_name>', true);
```

```
farm=> SELECT CLEANUP_EXTENTS('<schema_name>','<table_name>', true);
```

To display the table:

```
farm=> SELECT delete_predicate FROM sqream_catalog.delete_predicates dp
   JOIN sqream_catalog.tables t
   ON dp.table_id = t.table_id
   WHERE t.table_name = '<table_name>';
```

4.5.4 Best Practice

- After running large `DELETE` operations, run `CLEANUP_CHUNKS` and `CLEANUP_EXTENTS` to improve performance and free up space. These commands remove empty chunks and extents, respectively, and can help prevent fragmentation of the table.

- If you need to delete large segments of data from very large tables, consider using a `CREATE TABLE AS` operation instead. This involves creating a new table with the desired data and then renaming and dropping the original table. This approach can be faster and more efficient than running a large `DELETE` operation, especially if you don't need to preserve any data in the original table.

- Avoid interrupting or killing `CLEANUP_EXTENTS` operations that are in progress. These operations can take a while to complete, especially if the table is very large or has a lot of fragmentation, but interrupting them can cause data inconsistencies or other issues.

- SQream is optimized for time-based data, which means that data that is naturally ordered according to date or timestamp fields will generally perform better. If you need to delete rows from such tables, consider using the time-based columns in your `DELETE` predicates to improve performance.

4.6 Logging

4.6.1 Locating the Log Files

The *storage cluster* contains a `logs` directory. Each worker produces a log file in its own directory, which can be identified by the worker's hostname and port.

Note: Additional internal debug logs may reside in the main `logs` directory.

The worker logs contain information messages, warnings, and errors pertaining to SQream DB's operation, including:

- Server start-up and shutdown
- Configuration changes

- Exceptions and errors
- User login events
- Session events
- Statement execution success / failure
- Statement execution statistics

4.6.1.1 Log Structure and Contents

The log is a CSV, with several fields.

Table 3: Log fields

Field	Description
`#SQ#`	Start delimiter. When used with the end of line delimiter can be used to parse multi-line statements correctly
Row Id	Unique identifier for the row
Timestamp	Timestamp for the message (ISO 8601 date format)
Information Level	Information level of the message. See *information level table* below
Thread Id	System thread identifier (internal use)
Worker hostname	Hostname of the worker that generated the message
Worker port	Port of the worker that generated the message
Connection Id	Connection Id for the message. Defaults to `-1` if no connection
Database name	Database name that generated the message. Can be empty for no database
User Id	User role that was connected during the message. Can be empty if no user caused the message
Statement Id	Statement Id for the message. Defaults to `-1` if no statement
Service name	Service name for the connection. Can be empty.
Message type Id	Message type Id. See *message type table* below)
Message	Content for the message
`#EOM#`	End of line delimiter

Table 4: Information Level

Level	Description
`SYSTEM`	System information like start up, shutdown, configuration change
`FATAL`	Fatal errors that may cause outage
`ERROR`	Errors encountered during statement execution
`WARNING`	Warnings
`INFO`	Information and statistics

Table 5: Message Type

Type	Level	Description	Example message content
1	INFO	Statement start information	• "Query before parsing (statement handle opened) • "SELECT * FROM nba WHERE ""Team"" NOT LIKE ""Portland%%""" (statement preparing)
2	INFO	Statement passed to another worker for execution	• ""Reconstruct query before parsing" • "SELECT * FROM nba WHERE ""Team"" NOT LIKE ""Portland%%""" (statement preparing on node)
4	INFO	Statement has entered execution	"Statement execution"
10	INFO	Statement execution completed	"Success" / "Failed"
20	INFO	Compilation error, with accompanying error message	"Could not find function dateplart in catalog."
21	INFO	Execution error, with accompanying error message	Error text
30	INFO	Size of data read from disk in megabytes	18
31	INFO	Row count of result set	45
32	INFO	Processed Rows	450134749978
100	INFO	Session start - Client IP address	"192.168.5.5"
101	INFO	Login	"Login Success" / "Login Failed"
110	INFO	Session end	"Session ended"
200	INFO	show_node_info periodic output	
500	ERROR	Exception occured in a statement	"Cannot return the inverse cosine of a number not in [-1,1] range"
1000	SYSTEM	Worker startup message	"Server Start Time - 2019-12-30 21:18:31, SQream ver{v2020.2}"
1002	SYSTEM	Metadata	Metadata server

4.6.1.2 Log-Naming

Log file name syntax

`sqream_<date>_<sequence>.log`

- `date` is formatted `%y%m%d`, for example `20191231` for December 31st 2019.

  By default, each worker will create a new log file every time it is restarted.

- `sequence` is the log's sequence. When a log is rotated, the sequence number increases. This starts at `000`.

For example, `/home/rhendricks/sqream_storage/192.168.1.91_5000`.

See the *Changing Log Rotation* below for information about controlling this setting.

4.6.2 Log Control and Maintenance

4.6.2.1 Changing Log Verbosity

A few configuration settings alter the verbosity of the logs:

Table 6: Log verbosity configuration

Flag	Description	Default	Values
`logClientLevel`	Used to control which log level should appear in the logs	`4` (`INFO`)	`0` SYSTEM (lowest) - `4` INFO (highest). See *information level table* above.
`nodeInfoLoggingSec`	Sets an interval for automatically logging long-running statements' show_node_info output. Output is written as a message type `200`.	`60` (every minute)	Positive whole number >=1.

4.6.2.2 Changing Log Rotation

A few configuration settings alter the log rotation policy:

Table 7: Log rotation configuration

Flag	Description	Default	Values
`useLogMaxFileSize`	Rotate log files once they reach a certain file size. When `true`, set the `logMaxFileSizeMB` accordingly. When `false` set the `logFileRotateTimeFrequency` accordingly.	`false`	`false` or `true`.
`logMaxFileSizeMB`	Sets the size threshold in megabytes after which a new log file will be opened.	`20`	`1` to `1024` (1MB to 1GB)
`logFileRotateTimeFrequency`	Frequency of log rotation	`never`	`daily`, `weekly`, `monthly`, `never`

4.6.3 Collecting Logs from Your Cluster

Collecting logs from your cluster can be as simple as creating an archive from the `logs` subdirectory: `tar -czvf logs.tgz *.log`.

However, SQream DB comes bundled with a data collection utility and an SQL utility intended for collecting logs and additional information that can help SQream support drill down into possible issues.

4.6.3.1 SQL Syntax

```
SELECT REPORT_COLLECTION(output_path, mode)
;

output_path ::=
   filepath

mode ::=
   log | db | db_and_log
```

4.6.3.2 Command Line Utility

If you cannot access SQream DB for any reason, you can also use a command line tool to collect the same information:

```
$ ./bin/report_collection <path to storage> <path for output> <mode>
```

4.6.3.3 Parameters

Parameter	Description
`output_path`	Path for the output archive. The output file will be named `report_<date>_<time>.tar`.
`mode`	One of three modes: * `'log'` - Collects all log files * `'db'` - Collects the metadata database (includes DDL, but no data) * `'db_and_log'` - Collect both log files and metadata database

4.6.3.4 Example

Write an archive to `/home/rhendricks`, containing log files:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'log')
;
```

Write an archive to `/home/rhendricks`, containing log files and metadata database:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'db_and_log')
;
```

Using the command line utility:

```
$ ./bin/report_collection /home/rhendricks/sqream_storage /home/rhendricks db_and_log
```

4.6.4 Troubleshooting with Logs

4.6.4.1 Loading Logs with Foreign Tables

Assuming logs are stored at `/home/rhendricks/sqream_storage/logs/`, a database administrator can access the logs using the external_tables concept through SQream DB.

```
CREATE FOREIGN TABLE logs
(
  start_marker      TEXT(4),
  row_id            BIGINT,
  timestamp         DATETIME,
  message_level     TEXT,
  thread_id         TEXT,
  worker_hostname   TEXT,
  worker_port       INT,
  connection_id     INT,
  database_name     TEXT,
  user_name         TEXT,
  statement_id      INT,
  service_name      TEXT,
  message_type_id   INT,
  message           TEXT,
  end_message       TEXT(5)
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
     DELIMITER = '|',
     CONTINUE_ON_ERROR = true
  )
;
```

For more information, see Loading Logs with Foreign Tables.

4.6.4.2 Counting Message Types

```
t=> SELECT message_type_id, COUNT(*) FROM logs GROUP BY 1;
message_type_id | count
----------------+----------
              0 |         9
              1 |      5578
              4 |      2319
             10 |      2788
             20 |       549
             30 |       411
             31 |      1720
             32 |      1720
            100 |      2592
            101 |      2598
            110 |      2571
            200 |        11
            500 |       136
           1000 |        19
           1003 |        19
```

(continues on next page)

(continued from previous page)

```
           1004 |        19
           1010 |         5
```

4.6.4.3 Finding Fatal Errors

```
t=> SELECT message FROM logs WHERE message_type_id=1010;
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/rocksdb/LOCK: Resource temporarily unavailable
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/rocksdb/LOCK: Resource temporarily unavailable
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/LOCK: Resource temporarily unavailable
```

4.6.4.4 Countng Error Events Within a Certain Timeframe

```
t=> SELECT message_type_id,
.          COUNT(*)
.   FROM logs
.   WHERE message_type_id IN (1010,500)
.   AND timestamp BETWEEN '2019-12-20' AND '2020-01-01'
.   GROUP BY 1;
message_type_id | count
----------------+------
            500 |    18
           1010 |     3
```

4.6.4.5 Tracing Errors to Find Offending Statements

If we know an error occured, but don't know which statement caused it, we can find it using the connection ID and statement ID.

```
t=> SELECT connection_id, statement_id, message
.     FROM logs
.     WHERE message_level = 'ERROR'
.     AND timestamp BETWEEN '2020-01-01' AND '2020-01-06';
connection_id | statement_id | message
--------------+--------------+-------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪----------------------
           79 |           67 | Column type mismatch, expected UByte, got INT64 on␣
↪column Number, file name: /home/sqream/nba.parquet
```

Use the `connection_id` and `statement_id` to narrow down the results.

```
t=>   SELECT database_name, message FROM logs
.       WHERE connection_id=79 AND statement_id=67 AND message_type_id=1;
database_name | message
```

(continues on next page)

(continued from previous page)

```
--------------+-------------------------
master        | Query before parsing
master        | SELECT * FROM nba_parquet
```

4.7 Query Split

The split query operation optimizes long-running queries by executing them in parallel on different GPUs and/or Workers, reducing overall runtime. This involves breaking down a complex query into parallel executions on small data subsets. To ensure an ordered result set aligned with the original complex query, two prerequisites are essential. First, create an empty table mirroring the original result set's structure. Second, define the `@@SetResult` operator to split the query using an `INTEGER`, `DATE`, or `DATETIME` column, as these types are compatible with the operator's `min` and `max` variables.

Splitting is exclusive to the UI, utilizing Meta-scripting, a unique UI feature. Keep in mind that not all queries benefit, as this method introduces overhead runtime.

- *Syntax*
- *Example*
- *Best Practices*

4.7.1 Syntax

Creating an empty table mirroring the original query result set's structure using the same DDL:

```
CREATE TABLE <final_result_table>
AS
(
  SELECT
   -- Original query..
  WHERE
    <false_filter>
)
   -- A false_filter example: 1=2
```

Defining the `@@setresult` operator to split the original query using an `INTEGER`, `DATE`, or `DATETIME` column with `min` and `max` variables. If the column you're splitting by is used in a `WHERE` clause in the original query, use a `WHERE` clause when setting the `SetResult` operator as well.

```
@@SetResult minMax
SELECT
  MIN(<column>) AS min,
  MAX(<column>) AS max
FROM
  <my_table>
[WHERE
  <column> BETWEEN
   -- Integer Range:
   1 AND 100
   | -- Date Range:
   'yyyy-mm-dd' AND 'yyyy-mm-dd'
```

(continues on next page)

(continued from previous page)

```
  | -- DateTime Range:
  'yyyy-mm-dd hh:mm:ss:SSS' AND 'yyyy-mm-dd hh:mm:ss:SSS']
```

Defining the operator that determines the number of instances (splits) based on the data type of the column by which the query is split:

- **INTEGER column:** use the `@@SplitQueryByNumber` operator

```
@@SplitQueryByNumber instances = <number of instances>, from = minMax[0].min, to =␣
↪minMax[0].max
INSERT INTO <final_result_table>
(
  SELECT
   -- Original query..
  WHERE
    <column_to_split_by> BETWEEN '${from}' and '${to}'
)
```

- **DATE column:** use the `@@SplitQueryByDate` operator

```
@@SplitQueryByDate instances = <number of instances>, from = minMax[0].min, to =␣
↪minMax[0].max
INSERT INTO <final_result_table>
(
  SELECT
   -- Original query..
  WHERE
    <column_to_split_by> BETWEEN '${from}' and '${to}'
)
```

- **DATETIME column:** use the `@@SplitQueryByDateTime` operator

```
@@SplitQueryByDateTime instances = <number of instances>, from = minMax[0].min, to =␣
↪minMax[0].max
INSERT INTO <final_result_table>
(
  SELECT
   -- Original query..
  WHERE <column_to_split_by> BETWEEN '${from}' and '${to}'
)
```

Gathering results:

```
-- Gathering results for queries without aggregations:

SELECT *
FROM
  <final_result_table>
;

-- Gathering results for queries with aggregations and/or AVERAGE statement:

- AVERAGE:

SELECT
  <column1>, [,...],
  [SUM([DISTINCT] expr) AS <sum_column>],
```

(continues on next page)

(continued from previous page)

```
  [SUM(count_column) AS <sum_count_column>],
  [SUM(avg_column1) / SUM(avg_column2) AS <avg_column>]
FROM
  <final_result_table>
GROUP BY
  <column1>, <column2> [,...]
ORDER BY
  <column4>

-- Do not use a WHERE clause
```

4.7.2 Example

- *Creating a Sample Table and Query*
- *Splitting the Query*

4.7.2.1 Creating a Sample Table and Query

To split your first query, create the following table and insert data into it:

```
CREATE TABLE MyTable (
  id INT,
  name TEXT NOT NULL,
  age INT,
  salary INT,
  quantity INT
  );

-- Inserting data into the table
INSERT INTO MyTable (id, name, age, salary, quantity)
VALUES
  (1, 'John', 25, 50000, 10),
  (2, 'Jane', 30, 60000, 20),
  (3, 'Bob', 28, 55000, 15),
  (4, 'Emily', 35, 70000, 18),
  (5, 'David', 32, 62000, 22),
  (6, 'Sarah', 27, 52000, 12),
  (7, 'Michael', 40, 75000, 17),
  (8, 'Olivia', 22, 48000, 25),
  (9, 'William', 31, 58000, 14),
  (10, 'Sophia', 29, 56000, 19),
  (11, 'Liam', 26, 51000, 13),
  (12, 'Emma', 33, 64000, 16),
  (13, 'Daniel', 24, 49000, 23),
  (14, 'Ava', 37, 69000, 21),
  (15, 'Matthew', 23, 47000, 28),
  (16, 'Ella', 34, 67000, 24),
  (17, 'James', 28, 55000, 11),
  (18, 'Grace', 39, 72000, 26),
  (19, 'Benjamin', 30, 60000, 18),
  (20, 'Chloe', 25, 50000, 14),
```

(continues on next page)

(continued from previous page)

```
  (21, 'Logan', 38, 71000, 20),
  (22, 'Mia', 27, 52000, 16),
  (23, 'Christopher', 32, 62000, 22),
  (24, 'Aiden', 29, 56000, 19),
  (25, 'Lily', 36, 68000, 15),
  (26, 'Jackson', 31, 58000, 23),
  (27, 'Harper', 24, 49000, 12),
  (28, 'Ethan', 35, 70000, 17),
  (29, 'Isabella', 22, 48000, 25),
  (30, 'Carter', 37, 69000, 14),
  (31, 'Amelia', 26, 51000, 21),
  (32, 'Lucas', 33, 64000, 19),
  (33, 'Abigail', 28, 55000, 16),
  (34, 'Mason', 39, 72000, 18),
  (35, 'Evelyn', 30, 60000, 25),
  (36, 'Alexander', 23, 47000, 13),
  (37, 'Addison', 34, 67000, 22),
  (38, 'Henry', 25, 50000, 20),
  (39, 'Avery', 36, 68000, 15),
  (40, 'Sebastian', 29, 56000, 24),
  (41, 'Layla', 31, 58000, 11),
  (42, 'Wyatt', 38, 71000, 26),
  (43, 'Nora', 27, 52000, 19),
  (44, 'Grayson', 32, 62000, 17),
  (45, 'Scarlett', 24, 49000, 14),
  (46, 'Gabriel', 35, 70000, 23),
  (47, 'Hannah', 22, 48000, 16),
  (48, 'Eli', 37, 69000, 25),
  (49, 'Paisley', 28, 55000, 18),
  (50, 'Owen', 33, 64000, 12);
```

Next, we'll split the following query:

```
SELECT
  age,
  COUNT(*) AS total_people,
  AVG(salary) AS avg_salary,
  SUM(quantity) AS total_quantity,
  SUM(CASE WHEN quantity > 20 THEN 1 ELSE 0 END) AS high_quantity_count,
  SUM(CASE WHEN age BETWEEN 25 AND 30 THEN salary ELSE 0 END) AS total_salary_age_25_
↪30
FROM
  MyTable
WHERE
  salary > 55000
GROUP BY
  age
ORDER BY
  age;
```

4.7.2.2 Splitting the Query

1. Prepare an empty table mirroring the original query result set's structure with the same DDL, using a false filter under the `WHERE` clause.

   An **empty** table named `FinalResult` is created.

```
CREATE OR REPLACE TABLE FinalResult
AS
(
  SELECT
  age,
  COUNT(*) AS total_people,
  SUM(salary) AS avg_salary,
  COUNT(salary) AS avg_salary2,
  SUM(quantity) AS total_quantity,
  SUM(CASE WHEN quantity > 20 THEN 1 ELSE 0 END) AS high_quantity_count,
  SUM(CASE WHEN age BETWEEN 25 AND 30 THEN salary ELSE 0 END) AS total_salary_age_25_
↪30
FROM
  MyTable
WHERE
  1=0
  AND salary > 55000
GROUP BY
  age
ORDER BY
  age
  );
```

2. Set the `@@setresult` operator to split the original query using `min` and `max` variables.

```
@@ SetResult minMax
SELECT min(age) as min, max(age) as max
FROM mytable
;
```

3. Set the `@@SplitQueryByNumber` operator with the number of instances (splits) of your query (here based on an `INTEGER` column), and set the `between ${from} and ${to}` clause with the name of the column by which you wish to split your query (here the query is split by the `age` column.

```
@@SplitQueryByNumber instances = 4, from = minMax[0].min, to = minMax[0].max
INSERT INTO FinalResult
(
SELECT
  age,
  COUNT(*) AS total_people,
  SUM(salary) AS avg_salary,
  COUNT(salary) AS avg_salary2,
  SUM(quantity) AS total_quantity,
  SUM(CASE WHEN quantity > 20 THEN 1 ELSE 0 END) AS high_quantity_count,
  SUM(CASE WHEN age BETWEEN 25 AND 30 THEN salary ELSE 0 END) AS total_salary_age_25_
↪30
FROM
  MyTable
WHERE
  age between ${from} and ${to}
  AND salary > 55000
```

(continues on next page)

(continued from previous page)

```
GROUP BY
  age
ORDER BY
  age
  );
```

4. Gather the results of your query.

```
SELECT * FROM FinalResult ;
```

If we were to split the query Create a query that gathers the results of all instances (splits) into the empty table you created in step 1.

```
SELECT
  age,
  SUM(total_people) AS total_people,
  SUM(avg_salary) / SUM(avg_salary2) AS avg_salary,
  SUM(total_quantity) AS total_quantity,
  SUM(high_quantity_count) AS high_quantity_count,
  SUM(total_salary_age_25_30) AS total_salary_age_25_30
FROM
  FinalResult
GROUP BY
  age
ORDER BY
  age
  ;
```

5. Arrange ALL sequential scripts on one Editor tab.
6. Ensure that EACH script ends with a `;`.
7. Ensure that the **Execute** button is set to **All** so that all queries are consecutively executed.
8. Select the **Execute** button.

   All scripts are executed, resulting in the splitting of the initial query and a table containing the final result set.

4.7.3 Best Practices

4.7.3.1 General

- When incorporating the `LIMIT` clause or any aggregate function in your query, split the query based only on a `GROUP BY` column. If no relevant columns are present in the `GROUP BY` clause, the query might not be suitable for splitting.
- If you are not using aggregations, it's best to split the query using a column that appears in the a `WHERE` or `JOIN` clause.
- When using the `JOIN` key, it is usually better to use the key of the smaller table.

4.7.3.2 Choosing a Column to Split by

The column you split by must be sorted or mostly sorted. Meaning, that even if the column values may not be perfectly ordered, they still follow a general sequence or trend.

4.7.3.3 Aggregation Best Practices

Aggregation functions, or special functions need to have adjustments in the query that gathers the results of all instances (splits) into the empty table:

- `COUNT` becomes `SUM`
- The following statement and functions are split into two columns in the query split and then merged to be executed as one statement or function in the final query:
- `AVERAGE`
- User defined functions
- Variance functions
- Standard deviation functions

4.7.3.4 Date as Number best practices

When date is stored as number, using the number of workers as the instances number may not result in the expected way. e.g. if date run from 20210101 to 20210630 splitting to 8 will result in 6 relevant splits, as SQream only checks min and max and splits accordingly (20210630-20210101)/8. we get an instance of empty data with dates ranging from 20210432 to 20210499 (not really dates, but real numbers). In this case, we need to adjust the number of instance to get the right size splits. In the above example we need to split to 64, and each worker will run 3 splits with actual data.

4.8 Monitoring Query Performance

When analyzing options for query tuning, the first step is to analyze the query plan and execution. The query plan and execution details explain how SQream DB processes a query and where time is spent. This document details how to analyze query performance with execution plans. This guide focuses specifically on identifying bottlenecks and possible optimization techniques to improve query performance. Performance tuning options for each query are different. You should adapt the recommendations and tips for your own workloads. See also our *Optimization and Best Practices* guide for more information about data loading considerations and other best practices.

- *Setting Up the System for Monitoring*
- *Using the `SHOW_NODE_INFO` Command*
- *Understanding the Query Execution Plan Output*
- *Examples*
- *Further Reading*

4.8.1 Setting Up the System for Monitoring

By default, SQream DB logs execution details for every statement that runs for more than 60 seconds. If you want to see the execution details for a currently running statement, see *Using the SHOW_NODE_INFO Command* below.

4.8.1.1 Adjusting the Logging Frequency

To adjust the frequency of logging for statements, you may want to reduce the interval from 60 seconds down to, say, 5 or 10 seconds. Modify the configuration files and set the `nodeInfoLoggingSec` parameter as you see fit:

```
:emphasize-lines: 7

{
   "compileFlags":{
   },
   "runtimeFlags":{
   },
   "runtimeGlobalFlags":{
      "nodeInfoLoggingSec" : 5,
   },
   "server":{
   }
}
```

After restarting the SQream DB cluster, the execution plan details will be logged to the *standard SQream DB logs directory*, as a message of type `200`. You can see these messages with a text viewer or with queries on the log *Foreign Tables*.

4.8.1.2 Reading Execution Plans with a Foreign Table

First, create a foreign table for the logs

```
CREATE FOREIGN TABLE logs
(
  start_marker      TEXT(4),
  row_id            BIGINT,
  timestamp         DATETIME,
  message_level     TEXT,
  thread_id         TEXT,
  worker_hostname   TEXT,
  worker_port       INT,
  connection_id     INT,
  database_name     TEXT,
  user_name         TEXT,
  statement_id      INT,
  service_name      TEXT,
  message_type_id   INT,
  message           TEXT,
  end_message       TEXT(5)
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
     DELIMITER = '|'
  )
;
```

Once you've defined the foreign table, you can run queries to observe the previously logged execution plans. This is recommended over looking at the raw logs.

```
t=> SELECT message
.     FROM logs
.     WHERE message_type_id = 200
.     AND timestamp BETWEEN '2020-06-11' AND '2020-06-13';
message
-------------------------------------------------------------------------------------
↪-------------------------------------------
SELECT *,coalesce((depdelay > 15),false) AS isdepdelayed FROM ontime WHERE year IN␣
↪(2005, 2006, 2007, 2008, 2009, 2010)
 :
 : 1,PushToNetworkQueue  ,10354468,10,1035446,2020-06-12 20:41:42,-1,,,,13.55
 : 2,Rechunk             ,10354468,10,1035446,2020-06-12 20:41:42,1,,,,0.10
 : 3,ReorderInput        ,10354468,10,1035446,2020-06-12 20:41:42,2,,,,0.00
 : 4,DeferredGather      ,10354468,10,1035446,2020-06-12 20:41:42,3,,,,1.23
 : 5,ReorderInput        ,10354468,10,1035446,2020-06-12 20:41:41,4,,,,0.01
 : 6,GpuToCpu            ,10354468,10,1035446,2020-06-12 20:41:41,5,,,,0.07
 : 7,GpuTransform        ,10354468,10,1035446,2020-06-12 20:41:41,6,,,,0.02
 : 8,ReorderInput        ,10354468,10,1035446,2020-06-12 20:41:41,7,,,,0.00
 : 9,Filter              ,10354468,10,1035446,2020-06-12 20:41:41,8,,,,0.07
 : 10,GpuTransform        ,10485760,10,1048576,2020-06-12 20:41:41,9,,,,0.07
 : 11,GpuDecompress       ,10485760,10,1048576,2020-06-12 20:41:41,10,,,,0.03
 : 12,GpuTransform        ,10485760,10,1048576,2020-06-12 20:41:41,11,,,,0.22
 : 13,CpuToGpu            ,10485760,10,1048576,2020-06-12 20:41:41,12,,,,0.76
 : 14,ReorderInput        ,10485760,10,1048576,2020-06-12 20:41:40,13,,,,0.11
 : 15,Rechunk             ,10485760,10,1048576,2020-06-12 20:41:40,14,,,,5.58
 : 16,CpuDecompress       ,10485760,10,1048576,2020-06-12 20:41:34,15,,,,0.04
 : 17,ReadTable           ,10485760,10,1048576,2020-06-12 20:41:34,16,832MB,,public.
↪ontime,0.55
```

4.8.2 Using the SHOW_NODE_INFO Command

The show_node_info command returns a snapshot of the current query plan, similar to `EXPLAIN ANALYZE` from other databases. The show_node_info result, just like the periodically-logged execution plans described above, are an at-the-moment view of the compiler's execution plan and runtime statistics for the specified statement. To inspect a currently running statement, execute the `show_node_info` utility function in a SQL client like *sqream sql*, the SQream Studio Editor, or any other third party SQL terminal.

In this example, we inspect a statement with statement ID of 176. The command looks like this:

```
t=> SELECT SHOW_NODE_INFO(176);
stmt_id | node_id | node_type          | rows | chunks | avg_rows_in_chunk | time    ␣
↪           | parent_node_id | read | write | comment    | timeSum
--------+---------+--------------------+------+--------+-------------------+----------
↪-----------+----------------+------+-------+------------+--------
    176 |       1 | PushToNetworkQueue |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |             -1 |      |       |            |  0.0025
    176 |       2 | Rechunk            |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              1 |      |       |            |       0
    176 |       3 | GpuToCpu           |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              2 |      |       |            |       0
    176 |       4 | ReorderInput       |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              3 |      |       |            |       0
    176 |       5 | Filter             |    1 |      1 |                 1 | 2019-12-
```

(continues on next page)

(continued from previous page)

```
↪25 23:53:13 |              4 |       |        |             |  0.0002
   176 |       6 | GpuTransform        |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              5 |       |        |             |  0.0002
   176 |       7 | GpuDecompress       |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              6 |       |        |             |       0
   176 |       8 | CpuToGpu            |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              7 |       |        |             |  0.0003
   176 |       9 | Rechunk             |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              8 |       |        |             |       0
   176 |      10 | CpuDecompress       |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              9 |       |        |             |       0
   176 |      11 | ReadTable           |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |             10 | 4MB   |        | public.nba |  0.0004
```

Alternatively, you may also *retrieve the query execution plan output* using SQreamDB Studio, and contact SQream Support.

4.8.3 Understanding the Query Execution Plan Output

Both show_node_info and the logged execution plans represents the query plan as a graph hierarchy, with data separated into different columns. Each row represents a single logical database operation, which is also called a **node** or **chunk producer**. A node reports several metrics during query execution, such as how much data it has read and written, how many chunks and rows, and how much time has elapsed. Consider the example show_node_info presented above. The source node with ID #11 (`ReadTable`), has a parent node ID #10 (`CpuDecompress`). If we were to draw this out in a graph, it'd look like this:

The last node, also called the sink, has a parent node ID of -1, meaning it has no parent. This is typically a node that sends data over the network or into a table.

When using show_node_info, a tabular representation of the currently running statement execution is presented. See the examples below to understand how the query execution plan is instrumental in identifying bottlenecks and optimizing long-running statements.

4.8.3.1 Information Presented in the Execution Plan

4.8.3.2 Commonly Seen Nodes

Table 8: Node types

Column name	Execution location	Description
`CpuDecompress`	CPU	Decompression operation, common for longer `TEXT` types
`CpuLoopJoin`	CPU	A non-indexed nested loop join, performed on the CPU
`CpuReduce`	CPU	A reduce process performed on the CPU, primarily with `DISTINCT` aggregates (e
`CpuToGpu`, `GpuToCpu`		An operation that moves data to or from the GPU for processing
`CpuTransform`	CPU	A transform operation performed on the CPU, usually a *scalar function*
`DeferredGather`	CPU	Merges the results of GPU operations with a result set
`Distinct`	GPU	Removes duplicate rows (usually as part of the `DISTINCT` operation)
`Distinct_Merge`	CPU	The merge operation of the `Distinct` operation
`Filter`	GPU	A filtering operation, such as a `WHERE` or `JOIN` clause
`GpuDecompress`	GPU	Decompression operation
`GpuReduceMerge`	GPU	An operation to optimize part of the merger phases in the GPU
`GpuTransform`	GPU	A transformation operation such as a type cast or *scalar function*

Table 8 – continued from previous page

Column name	Execution location	Description
`LocateFiles`	CPU	Validates external file paths for foreign data wrappers, expanding directories and G
`LoopJoin`	GPU	A non-indexed nested loop join, performed on the GPU
`ParseCsv`	CPU	A CSV parser, used after `ReadFiles` to convert the CSV into columnar data
`PushToNetworkQueue`	CPU	Sends result sets to a client connected over the network
`ReadFiles`	CPU	Reads external flat-files
`ReadTable`	CPU	Reads data from a standard table stored on disk
`Rechunk`		Reorganize multiple small chunks into a full chunk. Commonly found after joins a
`Reduce`	GPU	A reduction operation, such as a `GROUP BY`
`ReduceMerge`	GPU	A merge operation of a reduction operation, helps operate on larger-than-RAM dat
`ReorderInput`		Change the order of arguments in preparation for the next operation
`SeparatedGather`	GPU	Gathers additional columns for the result
`Sort`	GPU	Sort operation
`TakeRowsFromChunk`		Take the first N rows from each chunk, to optimize `LIMIT` when used alongside `O`
`Top`		Limits the input size, when used with `LIMIT` (or its alias `TOP`)
`UdfTransform`	CPU	Executes a *user defined function*
`UnionAll`		Combines two sources of data when `UNION ALL` is used
`Window`	GPU	Executes a non-ranking window function
`WindowRanking`	GPU	Executes a ranking window function
`WriteTable`	CPU	Writes the result set to a standard table stored on disk

Tip: The full list of nodes appears in the Node types table, as part of the show_node_info reference.

4.8.4 Examples

In general, looking at the top three longest running nodes (as is detailed in the `timeSum` column) can indicate the biggest bottlenecks. In the following examples you will learn how to identify and solve some common issues.

4.8.4.1 Spooling to Disk

When there is not enough RAM to process a statement, SQream DB will spill over data to the `temp` folder in the storage disk. While this ensures that a statement can always finish processing, it can slow down the processing significantly. It's worth identifying these statements, to figure out if the cluster is configured correctly, as well as potentially reduce the statement size. You can identify a statement that spools to disk by looking at the `write` column in the execution details. A node that spools will have a value, shown in megabytes in the `write` column. Common nodes that write spools include `Join` or `LoopJoin`.

4.8.4.1.1 Identifying the Offending Nodes

1. Run a query.

   For example, a query from the TPC-H benchmark:

```
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS␣
↪mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
```

(continues on next page)

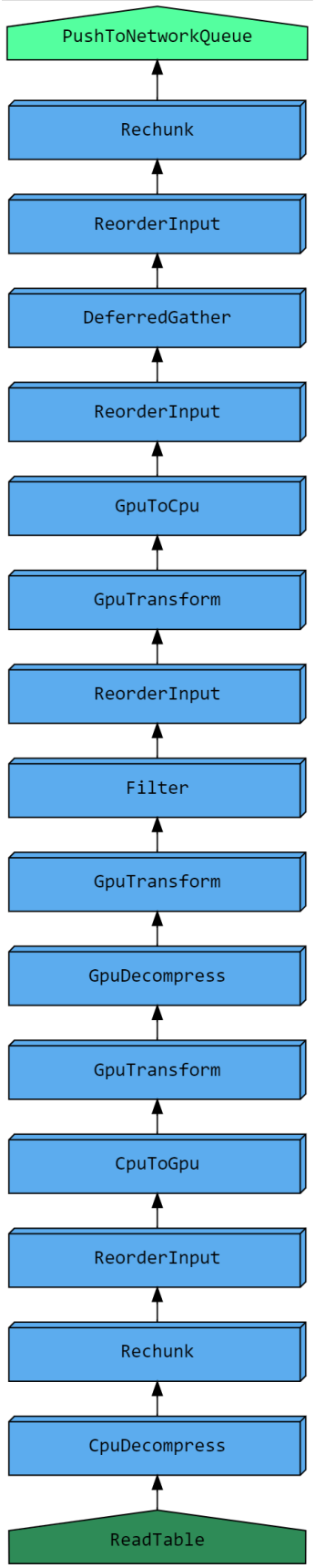


Fig. 1: This graph explains how the query execution details are arranged in a logical order, from the bottom up.

(continued from previous page)

```
            n2.n_name AS nation
      FROM lineitem
        JOIN part ON p_partkey = CAST (l_partkey AS INT)
        JOIN orders ON l_orderkey = o_orderkey
        JOIN customer ON o_custkey = c_custkey
        JOIN nation n1 ON c_nationkey = n1.n_nationkey
        JOIN region ON n1.n_regionkey = r_regionkey
        JOIN supplier ON s_suppkey = l_suppkey
        JOIN nation n2 ON s_nationkey = n2.n_nationkey
      WHERE o_orderdate BETWEEN '1995-01-01' AND '1996-12-31') AS all_nations
GROUP BY o_year
ORDER BY o_year;
```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   This statement is made up of 199 nodes, starting from a `ReadTable`, and finishes by returning only 2 results to the client.

   The execution below has been shortened, but note the highlighted rows for `LoopJoin`:

```
:emphasize-lines: 33,35,37,39

t=> SELECT message FROM logs WHERE message_type_id = 200 LIMIT 1;
message
--------------------------------------------------------------------------------
↪-------
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS␣
↪mkt_share
 : FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
 :              l_extendedprice*(1 - l_discount / 100.0) AS volume,
 :              n2.n_name AS nation
 :       FROM lineitem
 :         JOIN part ON p_partkey = CAST (l_partkey AS INT)
 :         JOIN orders ON l_orderkey = o_orderkey
 :         JOIN customer ON o_custkey = c_custkey
 :         JOIN nation n1 ON c_nationkey = n1.n_nationkey
 :         JOIN region ON n1.n_regionkey = r_regionkey
 :         JOIN supplier ON s_suppkey = l_suppkey
 :         JOIN nation n2 ON s_nationkey = n2.n_nationkey
 :       WHERE o_orderdate BETWEEN '1995-01-01' AND '1996-12-31') AS all_nations
 : GROUP BY o_year
 : ORDER BY o_year
 : 1,PushToNetworkQueue  ,2,1,2,2020-09-04 18:32:50,-1,,,,0.27
 : 2,Rechunk             ,2,1,2,2020-09-04 18:32:50,1,,,,0.00
 : 3,SortMerge           ,2,1,2,2020-09-04 18:32:49,2,,,,0.00
 : 4,GpuToCpu            ,2,1,2,2020-09-04 18:32:49,3,,,,0.00
 : 5,Sort                ,2,1,2,2020-09-04 18:32:49,4,,,,0.00
 : 6,ReorderInput        ,2,1,2,2020-09-04 18:32:49,5,,,,0.00
 : 7,GpuTransform        ,2,1,2,2020-09-04 18:32:49,6,,,,0.00
 : 8,CpuToGpu            ,2,1,2,2020-09-04 18:32:49,7,,,,0.00
 : 9,Rechunk             ,2,1,2,2020-09-04 18:32:49,8,,,,0.00
 : 10,ReduceMerge         ,2,1,2,2020-09-04 18:32:49,9,,,,0.03
 : 11,GpuToCpu            ,6,3,2,2020-09-04 18:32:49,10,,,,0.00
 : 12,Reduce              ,6,3,2,2020-09-04 18:32:49,11,,,,0.64
 [...]
 : 49,LoopJoin            ,182369485,7,26052783,2020-09-04 18:32:36,48,1915MB,
```

(continues on next page)

(continued from previous page)

```
↪1915MB,inner,4.94
 [...]
 : 98,LoopJoin            ,182369485,12,15197457,2020-09-04 18:32:16,97,2191MB,
↪2191MB,inner,5.01
 [...]
 : 124,LoopJoin            ,182369485,8,22796185,2020-09-04 18:32:03,123,3064MB,
↪3064MB,inner,6.73
 [...]
 : 150,LoopJoin            ,182369485,10,18236948,2020-09-04 18:31:47,149,12860MB,
↪12860MB,inner,23.62
 [...]
 : 199,ReadTable           ,20000000,1,20000000,2020-09-04 18:30:33,198,0MB,,
↪public.part,0.83
```

Because of the relatively low amount of RAM in the machine and because the data set is rather large at around 10TB, SQream DB needs to spool.

The total spool used by this query is around 20GB (1915MB + 2191MB + 3064MB + 12860MB).

4.8.4.1.2 Common Solutions for Reducing Spool

- Increase the amount of spool memory available for the workers, as a proportion of the maximum statement memory. When the amount of spool memory is increased, SQream DB may not need to write to disk.

  This setting is called `spoolMemoryGB`. Refer to the configuration guide.

- Reduce the amount of **workers** per host, and increase the amount of spool available to the (now reduced amount of) active workers. This may reduce the amount of concurrent statements, but will improve performance for heavy statements.

4.8.4.2 Queries with Large Result Sets

When queries have large result sets, you may see a node called `DeferredGather`. This gathering occurs when the result set is assembled, in preparation for sending it to the client.

4.8.4.2.1 Identifying the Offending Nodes

1. Run a query.

   For example, a modified query from the TPC-H benchmark:

```
SELECT s.*,
       l.*,
       r.*,
       n1.*,
       n2.*,
       p.*,
       o.*,
       c.*
FROM lineitem l
  JOIN part p ON p_partkey = CAST (l_partkey AS INT)
  JOIN orders o ON l_orderkey = o_orderkey
  JOIN customer c ON o_custkey = c_custkey
```

(continues on next page)

(continued from previous page)

```
  JOIN nation n1 ON c_nationkey = n1.n_nationkey
  JOIN region r ON n1.n_regionkey = r_regionkey
  JOIN supplier s ON s_suppkey = l_suppkey
  JOIN nation n2 ON s_nationkey = n2.n_nationkey
WHERE r_name = 'AMERICA'
AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL');
```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   This statement is made up of 221 nodes, containing 8 `ReadTable` nodes, and finishes by returning billions of results to the client.

   The execution below has been shortened, but note the highlighted rows for `DeferredGather`:

```
:emphasize-lines: 7,9,11

t=> SELECT show_node_info(494);
stmt_id | node_id | node_type            | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read    | write | comment         |␣
↪timeSum
--------+---------+----------------------+-----------+--------+------------------
↪+---------------------+----------------+---------+-------+-----------------+----
↪----
    494 |       1 | PushToNetworkQueue   |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |             -1 |         |       |                 |  ␣
↪0.36
    494 |       2 | Rechunk              |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              1 |         |       |                 |  ␣
↪   0
    494 |       3 | ReorderInput         |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              2 |         |       |                 |  ␣
↪   0
    494 |       4 | DeferredGather       |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              3 |         |       |                 |  ␣
↪0.16
    [...]
    494 |     166 | DeferredGather       |   3998730 |     39 |            102531␣
↪| 2020-09-04 19:07:47 |            165 |         |       |                 |  ␣
↪21.75
    [...]
    494 |     194 | DeferredGather       |    133241 |     20 |              6662␣
↪| 2020-09-04 19:07:03 |            193 |         |       |                 |  ␣
↪0.41
    [...]
    494 |     221 | ReadTable            |  20000000 |     20 |           1000000␣
↪| 2020-09-04 19:07:01 |            220 | 20MB    |       | public.part     |  ␣
↪ 0.1
```

   When you see `DeferredGather` operations taking more than a few seconds, that's a sign that you're selecting too much data. In this case, the DeferredGather with node ID 166 took over 21 seconds.

3. Modify the statement to see the difference Altering the select clause to be more restrictive will reduce the deferred gather time back to a few milliseconds.

```
SELECT DATEPART(year, o_orderdate) AS o_year,
       l_extendedprice * (1 - l_discount / 100.0) as volume,
```

(continues on next page)

(continued from previous page)

```
       n2.n_name as nation
FROM ...
```

4.8.4.2.2 Common Solutions for Reducing Gather Time

- Reduce the effect of the preparation time. Avoid selecting unnecessary columns (`SELECT * FROM...`), or reduce the result set size by using more filters.

4.8.4.3 Inefficient Filtering

When running statements, SQream DB tries to avoid reading data that is not needed for the statement by skipping chunks. If statements do not include efficient filtering, SQream DB will read a lot of data off disk. In some cases, you need the data and there's nothing to do about it. However, if most of it gets pruned further down the line, it may be efficient to skip reading the data altogether by using the metadata.

4.8.4.3.1 Identifying the Situation

We consider the filtering to be inefficient when the `Filter` node shows that the number of rows processed is less than a third of the rows passed into it by the `ReadTable` node. For example: #.

Run a query.

In this example, we execute a modified query from the TPC-H benchmark. Our `lineitem` table contains 600,037,902 rows.

```
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume)␣
↪AS mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
             n2.n_name AS nation
      FROM lineitem
        JOIN part ON p_partkey = CAST (l_partkey AS INT)
        JOIN orders ON l_orderkey = o_orderkey
        JOIN customer ON o_custkey = c_custkey
        JOIN nation n1 ON c_nationkey = n1.n_nationkey
        JOIN region ON n1.n_regionkey = r_regionkey
        JOIN supplier ON s_suppkey = l_suppkey
        JOIN nation n2 ON s_nationkey = n2.n_nationkey
      WHERE r_name = 'AMERICA'
      AND   lineitem.l_quantity = 3
      AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
      AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL')) AS all_
↪nations
GROUP BY o_year
ORDER BY o_year;
```

1. Observe the execution information by using the foreign table, or use `show_node_info`

   The execution below has been shortened, but note the highlighted rows for `ReadTable` and `Filter`:

```
:linenos:
:emphasize-lines: 9,17,19,27

t=> SELECT show_node_info(559);
stmt_id | node_id | node_type            | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read   | write | comment         |␣
↪timeSum
--------+---------+----------------------+-----------+--------+-------------------
↪+---------------------+----------------+--------+-------+-----------------+-----
↪---
    559 |       1 | PushToNetworkQueue   |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |             -1 |        |       |                 |   ␣
↪0.28
    559 |       2 | Rechunk              |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              1 |        |       |                 |    ␣
↪  0
    559 |       3 | SortMerge            |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              2 |        |       |                 |    ␣
↪  0
    559 |       4 | GpuToCpu             |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              3 |        |       |                 |    ␣
↪  0
[...]
    559 |     189 | Filter               |  12007447 |     12 |           1000620␣
↪| 2020-09-07 11:12:00 |            188 |        |       |                 |    ␣
↪0.3
    559 |     190 | GpuTransform         | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            189 |        |       |                 |   ␣
↪0.02
    559 |     191 | GpuDecompress        | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            190 |        |       |                 |   ␣
↪0.16
    559 |     192 | GpuTransform         | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            191 |        |       |                 |   ␣
↪0.02
    559 |     193 | CpuToGpu             | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            192 |        |       |                 |   ␣
↪1.47
    559 |     194 | ReorderInput         | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            193 |        |       |                 |    ␣
↪  0
    559 |     195 | Rechunk              | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            194 |        |       |                 |    ␣
↪  0
    559 |     196 | CpuDecompress        | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            195 |        |       |                 |    ␣
↪  0
    559 |     197 | ReadTable            | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            196 | 7587MB |       | public.lineitem |    ␣
↪0.1
[...]
    559 |     208 | Filter               |    133241 |     20 |              6662␣
↪| 2020-09-07 11:11:57 |            207 |        |       |                 |   ␣
↪0.01
    559 |     209 | GpuTransform         |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            208 |        |       |                 |   ␣
↪0.02
```

(continues on next page)

(continued from previous page)

```
    559 |     210 | GpuDecompress        |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            209 |        |       |                 |   ␣
↪0.03
    559 |     211 | GpuTransform         |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            210 |        |       |                 |   ␣
↪  0
    559 |     212 | CpuToGpu             |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            211 |        |       |                 |   ␣
↪0.01
    559 |     213 | ReorderInput         |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            212 |        |       |                 |   ␣
↪  0
    559 |     214 | Rechunk              |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            213 |        |       |                 |   ␣
↪  0
    559 |     215 | CpuDecompress        |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            214 |        |       |                 |   ␣
↪  0
    559 |     216 | ReadTable            |  20000000 |     20 |           1000000␣
↪| 2020-09-07 11:11:57 |            215 | 20MB   |       | public.part     |   ␣
↪  0
```

- The `Filter` on line 9 has processed 12,007,447 rows, but the output of `ReadTable` on `public.lineitem` on line 17 was 600,037,902 rows. This means that it has filtered out 98% ($1 - \frac{600037902}{12007447} = 98\%$) of the data, but the entire table was read.
- The `Filter` on line 19 has processed 133,000 rows, but the output of `ReadTable` on `public.part` on line 27 was 20,000,000 rows. This means that it has filtered out >99% ($1 - \frac{133241}{20000000} = 99.4\%$) of the data, but the entire table was read. However, this table is small enough that we can ignore it.

2. Modify the statement to see the difference Altering the statement to have a `WHERE` condition on the clustered `l_orderkey` column of the `lineitem` table will help SQream DB skip reading the data.

```
:emphasize-lines: 15

SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS␣
↪mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
             n2.n_name AS nation
      FROM lineitem
        JOIN part ON p_partkey = CAST (l_partkey AS INT)
        JOIN orders ON l_orderkey = o_orderkey
        JOIN customer ON o_custkey = c_custkey
        JOIN nation n1 ON c_nationkey = n1.n_nationkey
        JOIN region ON n1.n_regionkey = r_regionkey
        JOIN supplier ON s_suppkey = l_suppkey
        JOIN nation n2 ON s_nationkey = n2.n_nationkey
      WHERE r_name = 'AMERICA'
      AND   lineitem.l_orderkey > 4500000
      AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
      AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL')) AS all_nations
GROUP BY o_year
ORDER BY o_year;
```

```
:linenos:
:emphasize-lines: 5,13

t=> SELECT show_node_info(586);
stmt_id | node_id | node_type            | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read   | write | comment         |␣
↪timeSum
--------+---------+----------------------+-----------+--------+-------------------
↪+---------------------+----------------+--------+-------+-----------------+-----
↪---
[...]
    586 |     190 | Filter               | 494621593 |      8 |          61827699␣
↪| 2020-09-07 13:20:45 |            189 |        |       |                 |   ␣
↪0.39
    586 |     191 | GpuTransform         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            190 |        |       |                 |   ␣
↪0.03
    586 |     192 | GpuDecompress        | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            191 |        |       |                 |   ␣
↪0.26
    586 |     193 | GpuTransform         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            192 |        |       |                 |   ␣
↪0.01
    586 |     194 | CpuToGpu             | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            193 |        |       |                 |   ␣
↪1.86
    586 |     195 | ReorderInput         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            194 |        |       |                 |    ␣
↪  0
    586 |     196 | Rechunk              | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            195 |        |       |                 |    ␣
↪  0
    586 |     197 | CpuDecompress        | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            196 |        |       |                 |    ␣
↪  0
    586 |     198 | ReadTable            | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            197 | 6595MB |       | public.lineitem |   ␣
↪0.09
[...]
```

In this example, the filter processed 494,621,593 rows, while the output of `ReadTable` on `public.lineitem` was 494,927,872 rows. This means that it has filtered out all but 0.01% ($1 - \frac{494621593}{494927872} = 0.01\%$) of the data that was read.

The metadata skipping has performed very well, and has pre-filtered the data for us by pruning unnecessary chunks.

4.8.4.3.2 Common Solutions for Improving Filtering

- Use clustering keys and naturally ordered data in your filters.
- Avoid full table scans when possible

4.8.4.4 Joins with `text` Keys

Joins on long text keys do not perform as well as numeric data types or very short text keys.

4.8.4.4.1 Identifying the Situation

When a join is inefficient, you may note that a query spends a lot of time on the `Join` node. For example, consider these two table structures:

```
CREATE TABLE t_a
(
  amt             FLOAT NOT NULL,
  i               INT NOT NULL,
  ts              DATETIME NOT NULL,
  country_code    TEXT(3) NOT NULL,
  flag            TEXT(10) NOT NULL,
  fk              TEXT(50) NOT NULL
);
CREATE TABLE t_b
(
  id           TEXT(50) NOT NULL
  prob         FLOAT NOT NULL,
  j            INT NOT NULL,
);
```

1. Run a query.

   In this example, we will join `t_a.fk` with `t_b.id`, both of which are `TEXT(50)`.

   ```
   SELECT AVG(t_b.j :: BIGINT),
          t_a.country_code
   FROM t_a
     JOIN t_b ON (t_a.fk = t_b.id)
   GROUP BY t_a.country_code
   ```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   The execution below has been shortened, but note the highlighted rows for `Join`. The `Join` node is by far the most time-consuming part of this statement - clocking in at 69.7 seconds joining 1.5 billion records.

   ```
   :linenos:
   :emphasize-lines: 8

   t=> SELECT show_node_info(5);
   stmt_id | node_id | node_type             | rows        | chunks | avg_rows_in_
   ↪chunk | time                  | parent_node_id | read   | write | comment     |␣
   ↪timeSum
   --------+---------+----------------------+------------+--------+------------------
   ↪-+---------------------+----------------+-------+-------+------------+--------
   [...]
   ```

(continues on next page)

(continued from previous page)

```
      5 |      19 | GpuTransform         | 1497366528 |    204 |          ␣
↪7340032 | 2020-09-08 18:29:03 |             18 |        |        |            |  ␣
↪ 1.46
      5 |      20 | ReorderInput         | 1497366528 |    204 |          ␣
↪7340032 | 2020-09-08 18:29:03 |             19 |        |        |            |  ␣
↪    0
      5 |      21 | ReorderInput         | 1497366528 |    204 |          ␣
↪7340032 | 2020-09-08 18:29:03 |             20 |        |        |            |  ␣
↪    0
      5 |      22 | Join                 | 1497366528 |    204 |          ␣
↪7340032 | 2020-09-08 18:29:03 |             21 |        |        | inner      |  ␣
↪ 69.7
      5 |      24 | AddSortedMinMaxMet.. |    6291456 |      1 |          ␣
↪6291456 | 2020-09-08 18:26:05 |             22 |        |        |            |  ␣
↪    0
      5 |      25 | Sort                 |    6291456 |      1 |          ␣
↪6291456 | 2020-09-08 18:26:05 |             24 |        |        |            |  ␣
↪ 2.06
[...]
      5 |      31 | ReadTable            |    6291456 |      1 |          ␣
↪6291456 | 2020-09-08 18:26:03 |             30 | 235MB  |        | public.t_b |  ␣
↪ 0.02
[...]
      5 |      41 | CpuDecompress        |   10000000 |      2 |          ␣
↪5000000 | 2020-09-08 18:26:09 |             40 |        |        |            |  ␣
↪    0
      5 |      42 | ReadTable            |   10000000 |      2 |          ␣
↪5000000 | 2020-09-08 18:26:09 |             41 | 14MB   |        | public.t_a |  ␣
↪    0
```

4.8.4.4.2 Improving Query Performance

- In general, try to avoid `TEXT` as a join key. As a rule of thumb, `BIGINT` works best as a join key.
- Convert text values on-the-fly before running the query. For example, the crc64 function takes a text input and returns a `BIGINT` hash.

  For example:

```
SELECT AVG(t_b.j::BIGINT), t_a.country_code
FROM "public"."t_a"
JOIN "public"."t_b" ON (CRC64(t_a.fk::TEXT) = CRC64(t_b.id::TEXT))
GROUP BY t_a.country_code;
```

The execution below has been shortened, but note the highlighted rows for `Join`. The `Join` node went from taking nearly 70 seconds, to just 6.67 seconds for joining 1.5 billion records.

```
:linenos:
:emphasize-lines: 8

t=> SELECT show_node_info(6);
   stmt_id | node_id | node_type            | rows       | chunks | avg_rows_in_
↪chunk | time                | parent_node_id | read  | write | comment    |␣
↪timeSum
   --------+---------+----------------------+------------+--------+---------------
```

(continues on next page)

(continued from previous page)

```
↪----+--------------------+----------------+-------+-------+------------+-------
↪-
   [...]
         6 |      19 | GpuTransform         | 1497366528 |     85 |         ␣
↪17825792 | 2020-09-08 18:57:04 |             18 |       |       |            | ␣
↪  1.48
         6 |      20 | ReorderInput         | 1497366528 |     85 |         ␣
↪17825792 | 2020-09-08 18:57:04 |             19 |       |       |            | ␣
↪     0
         6 |      21 | ReorderInput         | 1497366528 |     85 |         ␣
↪17825792 | 2020-09-08 18:57:04 |             20 |       |       |            | ␣
↪     0
         6 |      22 | Join                 | 1497366528 |     85 |         ␣
↪17825792 | 2020-09-08 18:57:04 |             21 |       |       | inner      | ␣
↪  6.67
         6 |      24 | AddSortedMinMaxMet.. |    6291456 |      1 |          ␣
↪6291456 | 2020-09-08 18:55:12 |             22 |       |       |            |  ␣
↪    0
   [...]
         6 |      32 | ReadTable            |    6291456 |      1 |          ␣
↪6291456 | 2020-09-08 18:55:12 |             31 | 235MB |       | public.t_b |  ␣
↪ 0.02
   [...]
         6 |      43 | CpuDecompress        |   10000000 |      2 |          ␣
↪5000000 | 2020-09-08 18:55:13 |             42 |       |       |            |  ␣
↪    0
         6 |      44 | ReadTable            |   10000000 |      2 |          ␣
↪5000000 | 2020-09-08 18:55:13 |             43 | 14MB  |       | public.t_a |  ␣
↪    0
```

- You can map some text values to numeric types by using a dimension table. Then, reconcile the values when you need them by joining the dimension table.

4.8.4.5 Sorting on big `TEXT` fields

In general, SQream DB automatically inserts a `Sort` node which arranges the data prior to reductions and aggregations. When running a `GROUP BY` on large `TEXT` fields, you may see nodes for `Sort` and `Reduce` taking a long time.

4.8.4.5.1 Identifying the Situation

When running a statement, inspect it with show_node_info. If you see `Sort` and `Reduce` among your top five longest running nodes, there is a potential issue. For example: #.

Run a query to test it out.

Our `t_inefficient` table contains 60,000,000 rows, and the structure is simple, but with an oversized `country_code` column:

```
:emphasize-lines: 5

CREATE TABLE t_inefficient (
   i INT NOT NULL,
   amt DOUBLE NOT NULL,
   ts DATETIME NOT NULL,
```

(continues on next page)

(continued from previous page)

```
   country_code TEXT(100) NOT NULL,
   flag TEXT(10) NOT NULL,
   string_fk TEXT(50) NOT NULL
);
```

We will run a query, and inspect it's execution details:

```
t=> SELECT country_code,
.          SUM(amt)
.   FROM t_inefficient
.   GROUP BY country_code;
executed
time: 47.55s

country_code | sum
-------------+-----------
VUT          | 1195416012
GIB          | 1195710372
TUR          | 1195946178
[...]
```

```
:emphasize-lines: 8,9

t=> select show_node_info(30);
stmt_id | node_id | node_type          | rows     | chunks | avg_rows_in_
↪chunk | time                | parent_node_id | read  | write | comment    ␣
↪         | timeSum
--------+---------+--------------------+----------+--------+----------------
↪--+---------------------+----------------+-------+-------+----------------
↪-----+--------
     30 |       1 | PushToNetworkQueue |      249 |      1 |              ␣
↪249 | 2020-09-10 16:17:10 |             -1 |       |       |              ␣
↪       |    0.25
     30 |       2 | Rechunk            |      249 |      1 |              ␣
↪249 | 2020-09-10 16:17:10 |              1 |       |       |              ␣
↪       |       0
     30 |       3 | ReduceMerge        |      249 |      1 |              ␣
↪249 | 2020-09-10 16:17:10 |              2 |       |       |              ␣
↪       |    0.01
     30 |       4 | GpuToCpu           |     1508 |     15 |              ␣
↪100 | 2020-09-10 16:17:10 |              3 |       |       |              ␣
↪       |       0
     30 |       5 | Reduce             |     1508 |     15 |              ␣
↪100 | 2020-09-10 16:17:10 |              4 |       |       |              ␣
↪       |    7.23
     30 |       6 | Sort               | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |              5 |       |       |          ␣
↪           |    36.8
     30 |       7 | GpuTransform       | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |              6 |       |       |          ␣
↪           |    0.08
     30 |       8 | GpuDecompress      | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |              7 |       |       |          ␣
↪           |    2.01
     30 |       9 | CpuToGpu           | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |              8 |       |       |          ␣
```

(continues on next page)

(continued from previous page)

```
↪            |    0.16
     30 |      10 | Rechunk            | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |              9 |       |        |          ␣
↪            |       0
     30 |      11 | CpuDecompress      | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |             10 |       |        |          ␣
↪            |       0
     30 |      12 | ReadTable          | 60000000 |     15 |          ␣
↪4000000 | 2020-09-10 16:17:10 |             11 | 520MB |        | public.t_
↪inefficient |    0.05
```

1. We can look to see if there's any shrinking we can do on the `GROUP BY` key

```
t=> SELECT MAX(LEN(country_code)) FROM t_inefficient;
max
---
3
```

With a maximum string length of just 3 characters, our `TEXT(100)` is way oversized.

2. We can recreate the table with a more restrictive `TEXT(3)`, and can examine the difference in performance:

```
t=> CREATE TABLE t_efficient
.     AS SELECT i,
.               amt,
.               ts,
.               country_code::TEXT(3) AS country_code,
.               flag
.          FROM t_inefficient;
executed
time: 16.03s

t=> SELECT country_code,
.      SUM(amt::bigint)
.   FROM t_efficient
.   GROUP BY country_code;
executed
time: 4.75s
country_code | sum
-------------+-----------
VUT          | 1195416012
GIB          | 1195710372
TUR          | 1195946178
[...]
```

This time, the entire query took just 4.75 seconds, or just about 91% faster.

4.8.4.5.2 Improving Sort Performance on Text Keys

When using TEXT, ensure that the maximum length defined in the table structure is as small as necessary. For example, if you're storing phone numbers, don't define the field as `TEXT(255)`, as that affects sort performance.

You can run a query to get the maximum column length (e.g. `MAX(LEN(a_column))`), and potentially modify the table structure.

4.8.4.6 High Selectivity Data

Selectivity is the ratio of cardinality to the number of records of a chunk. We define selectivity as $\frac{\text{Distinct values}}{\text{Total number of records in a chunk}}$ SQream DB has a hint called `HIGH_SELECTIVITY`, which is a function you can wrap a condition in. The hint signals to SQream DB that the result of the condition will be very sparse, and that it should attempt to rechunk the results into fewer, fuller chunks. .. note:

```
SQream DB doesn't do this automatically because it adds a significant overhead on␣
↪naturally ordered and
well-clustered data, which is the more common scenario.
```

4.8.4.6.1 Identifying the Situation

This is easily identifiable - when the amount of average of rows in a chunk is small, following a `Filter` operation. Consider this execution plan:

```
t=> select show_node_info(30);
stmt_id | node_id | node_type         | rows      | chunks | avg_rows_in_chunk | time␣
↪               | parent_node_id | read  | write | comment    | timeSum
--------+---------+-------------------+-----------+--------+-------------------+------
↪---------------+----------------+-------+-------+------------+--------
[...]
     30 |      38 | Filter            |     18160 |     74 |               245 | 2020-
↪09-10 12:17:09 |             37 |       |       |            |   0.012
[...]
     30 |      44 | ReadTable         |  77000000 |     74 |           1040540 | 2020-
↪09-10 12:17:09 |             43 | 277MB |       | public.dim |   0.058
```

The table was read entirely - 77 million rows into 74 chunks. The filter node reduced the output to just 18,160 relevant rows, but they're distributed across the original 74 chunks. All of these rows could fit in one single chunk, instead of spanning 74 rather sparse chunks.

4.8.4.6.2 Improving Performance with High Selectivity Hints

- Use when there's a `WHERE` condition on an unclustered column, and when you expect the filter to cut out more than 60% of the result set.
- Use when the data is uniformly distributed or random

4.8.4.7 Performance of unsorted data in joins

When data is not well-clustered or naturally ordered, a join operation can take a long time.

4.8.4.7.1 Identifying the Situation

When running a statement, inspect it with show_node_info. If you see `Join` and `DeferredGather` among your top five longest running nodes, there is a potential issue. In this case, we're also interested in the number of chunks produced by these nodes.

Consider this execution plan:

```
:emphasize-lines: 6,11

t=> select show_node_info(30);
stmt_id | node_id | node_type          | rows      | chunks | avg_rows_in_chunk | time␣
↪                | parent_node_id | read  | write | comment    | timeSum
--------+---------+--------------------+-----------+--------+-------------------+------
↪---------------+----------------+-------+-------+------------+--------
[...]
     30 |      13 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             12 |       |       |            |   4.681
     30 |      14 | DeferredGather     | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             13 |       |       |            |  29.901
     30 |      15 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             14 |       |       |            |   3.053
     30 |      16 | GpuToCpu           | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             15 |       |       |            |   5.798
     30 |      17 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             16 |       |       |            |   2.899
     30 |      18 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             17 |       |       |            |   3.695
     30 |      19 | Join               | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             18 |       |       | inner      |  22.745
[...]
     30 |      38 | Filter             |     18160 |     74 |               245 | 2020-
↪09-10 12:17:09 |             37 |       |       |            |   0.012
[...]
     30 |      44 | ReadTable          |  77000000 |     74 |           1040540 | 2020-
↪09-10 12:17:09 |             43 | 277MB |       | public.dim |   0.058
```

- `Join` is the node that matches rows from both table relations.
- `DeferredGather` gathers the required column chunks to decompress

Pay special attention to the volume of data removed by the `Filter` node. The table was read entirely - 77 million rows into 74 chunks. The filter node reduced the output to just 18,160 relevant rows, but they're distributed across the original 74 chunks. All of these rows could fit in one single chunk, instead of spanning 74 rather sparse chunks.

4.8.4.7.2 Improving Join Performance when Data is Sparse

You can tell SQream DB to reduce the amount of chunks involved, if you know that the filter is going to be quite agressive by using the *HIGH_SELECTIVITY* hint described *above*. This forces the compiler to rechunk the data into fewer chunks. To tell SQream DB to rechunk the data, wrap a condition (or several) in the `HIGH_SELECTIVITY` hint:

```
:emphasize-lines: 13

-- Without the hint
SELECT *
FROM cdrs
WHERE
      RequestReceiveTime BETWEEN '2018-01-01 00:00:00.000' AND '2018-08-31 23:59:59.
↪999'
      AND EnterpriseID=1150
      AND MSISDN='9724871140341';

-- With the hint
SELECT *
FROM cdrs
WHERE
      HIGH_SELECTIVITY(RequestReceiveTime BETWEEN '2018-01-01 00:00:00.000' AND '2018-
↪08-31 23:59:59.999')
      AND EnterpriseID=1150
      AND MSISDN='9724871140341';
```

4.8.4.8 Manual Join Reordering

When joining multiple tables, you may wish to change the join order to join the smallest tables first.

4.8.4.8.1 Identifying the situation

When joining more than two tables, the `Join` nodes will be the most time-consuming nodes.

4.8.4.8.2 Changing the Join Order

Always prefer to join the smallest tables first. .. note:

```
We consider small tables to be tables that only retain a small amount of rows after␣
↪conditions
are applied. This bears no direct relation to the amount of total rows in the table.
```

Changing the join order can reduce the query runtime significantly. In the examples below, we reduce the time from 27.3 seconds to just 6.4 seconds.

```
:caption: Original query

-- This variant runs in 27.3 seconds
SELECT SUM(l_extendedprice / 100.0*(1 - l_discount / 100.0)) AS revenue,
       c_nationkey
FROM lineitem --6B Rows, ~183GB
  JOIN orders --1.5B Rows, ~55GB
```

(continues on next page)

(continued from previous page)

```
  ON   l_orderkey = o_orderkey
  JOIN customer --150M Rows, ~12GB
  ON   c_custkey = o_custkey

WHERE c_nationkey = 1
      AND   o_orderdate >= DATE '1993-01-01'
      AND   o_orderdate < '1994-01-01'
      AND   l_shipdate >= '1993-01-01'
      AND   l_shipdate <= dateadd(DAY,122,'1994-01-01')
GROUP BY c_nationkey
```

```
:caption: Modified query with improved join order

-- This variant runs in 6.4 seconds
SELECT SUM(l_extendedprice / 100.0*(1 - l_discount / 100.0)) AS revenue,
       c_nationkey
FROM orders --1.5B Rows, ~55GB
  JOIN customer --150M Rows, ~12GB
  ON   c_custkey = o_custkey
  JOIN lineitem --6B Rows, ~183GB
  ON   l_orderkey = o_orderkey

WHERE c_nationkey = 1
      AND   o_orderdate >= DATE '1993-01-01'
      AND   o_orderdate < '1994-01-01'
      AND   l_shipdate >= '1993-01-01'
      AND   l_shipdate <= dateadd(DAY,122,'1994-01-01')
GROUP BY c_nationkey
```

4.8.5 Further Reading

See our *Optimization and Best Practices* guide for more information about query optimization and data loading considerations.

4.9 Security

SQream DB has some security features that you should be aware of to increase the security of your data.

In this topic:

- *Overview*
- *Security best practices for SQream DB*
  - *Secure OS access*
  - *Change the default* `SUPERUSER`
  - *Create distinct user roles*
  - *Limit* `SUPERUSER` *access*
  - *Password strength guidelines*

- *Use TLS/SSL when possible*

4.9.1 Overview

An **initial, unsecured** installation of SQream DB can carry some risks:

- Your data open to any client that can access an open node through an IP and port combination.
- The initial administrator username and password, when unchanged, can let anyone log in.
- Network connections to SQream DB aren't encrypted.

To avoid these security risks, SQream DB provides authentication, authorizaiton, logging, and network encryption.

Read through the best practices guide to understand more.

4.9.2 Security best practices for SQream DB

4.9.2.1 Secure OS access

SQream DB often runs as a dedicated user on the host OS. This user is the file system owner of SQream DB data files.

Any user who logs in to the OS with this user can read or delete data from outside of SQream DB.

This user can also read any logs which may contain user login attempts.

Therefore, it is very important to secure the host OS and prevent unauthorized access.

System administrators should only log in to the host OS to perform maintenance tasks like upgrades. A database user should not log in using the same username in production environments.

4.9.2.2 Change the default `SUPERUSER`

To bootstrap SQream DB, a new install will always have one `SUPERUSER` role, typically named `sqream`. After creating a second `SUPERUSER` role, remove or change the default credentials to the default `sqream` user.

No database user should ever use the default `SUPERUSER` role in a production environment.

4.9.2.3 Create distinct user roles

Each user that signs in to a SQream DB cluster should have a distinct user role for several reasons:

- For logging and auditing purposes. Each user that logs in to SQream DB can be identified.
- For limiting permissions. Use groups and permissions to manage access. See our *Access Control* guide for more information.

4.9.2.4 Limit SUPERUSER access

Limit users who have the SUPERUSER role.

A superuser role bypasses all permissions checks. Only system administrators should have SUPERUSER roles. See our *Access Control* guide for more information.

4.9.2.5 Password strength guidelines

System administrators should verify the passwords used are strong ones.

SQream DB stores passwords as salted SHA1 hashes in the system catalog so they are obscured and can't be recovered. However, passwords may appear in server logs. Prevent access to server logs by securing OS access as described above.

Follow these recommendations to strengthen passwords:

- Pick a password that's easy to remember
- At least 8 characters
- Mix upper and lower case letters
- Mix letters and numbers
- Include non-alphanumeric characters (except " and ')

4.9.2.6 Use TLS/SSL when possible

SQream DB's protocol implements client/server TLS security (even though it is called SSL).

All SQream DB connectors and drivers support transport encryption. Ensure that each connection uses SSL and the correct access port for the SQream DB cluster:

- The load balancer (server_picker) is often started with the secure port at an offset of 1 from the original port (e.g. port 3108 for the unsecured connection and port 3109 for the secured connection).
- A SQream DB worker is often started with the secure port enabled at an offset of 100 from the original port (e.g. port 5000 for the unsecured connection and port 5100 for the secured connection).

Refer to each *client driver* for instructions on enabling TLS/SSL.

4.10 Saved Queries

The save_query command serves to both generate and store an execution plan, offering time savings for the execution of frequently used complex queries. It's important to note that the saved execution plan is closely tied to the structure of its underlying tables. Consequently, if any of the objects mentioned in the query undergo modification, the saved query must be recreated.

Saved queries undergo compilation during their creation. When executed, these queries utilize the precompiled query plan instead of compiling a new plan at query runtime.

4.10.1 Syntax

Saved queries related syntax:

```
-- Saving a query
SELECT SAVE_QUERY(saved_query_name, parameterized_query_string)

-- Showing a saved query
SELECT SHOW_SAVED_QUERY(saved_query_name)

-- Listing saved queries
SELECT LIST_SAVED_QUERIES()

-- Executing a saved query
SELECT EXECUTE_SAVED_QUERY(saved_query_name, [ , argument [ , ... ] ] )

-- Dropping a saved query
SELECT DROP_SAVED_QUERY(saved_query_name)

saved_query_name ::= string_literal
parameterized_query_string ::= string_literal
argument ::= string_literal | number_literal
```

4.10.1.1 Parameter Support

Query parameters can be used as substitutes for constants expressions in queries.

- Parameters cannot be used to substitute identifiers like column names and table names.
- Query parameters of a string datatype must be of a fixed length and may be used in equality checks but not with patterns such as like and rlike.

4.10.2 Permissions

Statement / Function	Permission
save_query	Saving queries requires no special permissions per se, however, it does require from the user to have permissions to access the tables referenced in the query and other query element permissions. The user who saved the query is granted all permissions on the saved query.
show_saved_query	Showing a saved query requires `SELECT` permissions on the saved query.
list_saved_queries	Listing saved queries requires no special permissions.
execute_saved_query	Executing a saved query requires `USAGE` permissions on the saved query and `SELECT` permissions to access the tables referenced in the query.
drop_saved_query	Dropping a saved query requires `DDL` permissions on the saved query and `SELECT` permissions to access the tables referenced in the query.

4.10.3 Parameterized Query

Parameterized queries, also known as prepared statements, enable the usage of parameters which may be replaced by actual values when executing the query. They are created and managed in application code, primarily to optimize query execution, enhance security, and allow for the reuse of query templates with different parameter values.

```
SELECT SAVE_QUERY('select_by_weight_and_team','SELECT * FROM nba WHERE Weight > ? AND␣
↪Team = ?');
```

4.11 Optimization and Best Practices

This topic explains some best practices of working with SQreamDB.

See also our *Monitoring Query Performance* guide for more information.

- *Table design*
- *Sorting*
- *Query Best Practices*
- *Data Loading Considerations*

4.11.1 Table design

4.11.1.1 Using DATE and DATETIME Data Types

When creating tables with dates or timestamps, using the purpose-built `DATE` and `DATETIME` types over integer types or `TEXT` will bring performance and storage footprint improvements, and in many cases huge performance improvements (as well as data integrity benefits). SQreamDB stores dates and datetimes very efficiently and can strongly optimize queries using these specific types.

4.11.1.2 Avoiding Data flattening and Denormalization

SQreamDB executes `JOIN` operations very effectively. It is almost always better to `JOIN` tables at query-time rather than flatten/denormalize your tables.

This will also reduce storage size and reduce row-lengths.

We highly suggest using `INT` or `BIGINT` as join keys, rather than a `TEXT` or `STRING` type.

4.11.1.3 Converting Foreign Tables to Native Tables

SQreamDB's native storage is heavily optimized for analytic workloads. It is always faster for querying than other formats, even columnar ones such as Parquet. It also enables the use of additional metadata to help speed up queries, in some cases by many orders of magnitude.

You can improve the performance of all operations by converting *Foreign Tables* into native tables by using the create_table_as syntax.

For example,

```
CREATE TABLE native_table AS SELECT * FROM foreign_table;
```

The one situation when this wouldn't be as useful is when data will be only queried once.

4.11.1.4 Leveraging Column Data Information

Knowing the data types and their ranges can help design a better table.

4.11.1.4.1 Appropriately Using `NULL` and `NOT NULL`

For example, if a value cannot be missing (or `NULL`), specify a `NOT NULL` constraint on the columns.

Not only does specifying `NOT NULL` save on data storage, it lets the query compiler know that a column cannot have a `NULL` value, which can improve query performance.

4.11.2 Sorting

Data sorting is an important factor in minimizing storage size and improving query performance.

- Minimizing storage saves on physical resources and increases performance by reducing overall disk I/O. Prioritize the sorting of low-cardinality columns. This reduces the number of chunks and extents that SQreamDB reads during query execution.
- Where possible, sort columns with the lowest cardinality first. Avoid sorting `TEXT` columns with lengths exceeding 50 characters.
- For longer-running queries that run on a regular basis, performance can be improved by sorting data based on the `WHERE` and `GROUP BY` parameters. Data can be sorted during insert by using *Foreign Tables* or by using create_table_as.

4.11.3 Query Best Practices

This section describes best practices for writing SQL queries.

4.11.3.1 Reducing Datasets Before Joining Tables

Reducing the input to a `JOIN` clause can increase performance. Some queries benefit from retrieving a reduced dataset as a subquery prior to a join.

For example,

```
SELECT store_name, SUM(amount)
FROM store_dim AS dim INNER JOIN store_fact AS fact ON dim.store_id=fact.store_id
WHERE p_date BETWEEN '2018-07-01' AND '2018-07-31'
GROUP BY 1;
```

Can be rewritten as

```
SELECT store_name, sum_amount
FROM store_dim AS dim INNER JOIN
   (SELECT SUM(amount) AS sum_amount, store_id
   FROM store_fact
```

(continues on next page)

(continued from previous page)

```
   WHERE p_date BETWEEN '2018-07-01' AND '2018-07-31'
   group by 2) AS fact
ON dim.store_id=fact.store_id;
```

4.11.3.2 Using ANSI JOIN

SQreamDB prefers the ANSI `JOIN` syntax. In some cases, the ANSI `JOIN` performs better than the non-ANSI variety.

For example, this ANSI JOIN example will perform better:

Listing 1: ANSI JOIN will perform better

```
SELECT p.name, s.name, c.name
FROM  "Products" AS p
JOIN  "Sales" AS s
  ON  p.product_id = s.sale_id
JOIN  "Customers" as c
  ON  s.c_id = c.id AND c.id = 20301125;
```

This non-ANSI `JOIN` is supported, but not recommended:

Listing 2: Non-ANSI JOIN may not perform well

```
SELECT p.name, s.name, c.name
FROM "Products" AS p, "Sales" AS s, "Customers" as c
WHERE p.product_id = s.sale_id
  AND s.c_id = c.id
  AND c.id = 20301125;
```

4.11.3.3 Using High-Selectivity hint

Selectivity is the ratio of cardinality to the number of records of a chunk. We define selectivity as $\frac{\text{Distinct values}}{\text{Total number of records in a chunk}}$

SQreamDB has a hint function called `HIGH_SELECTIVITY`, which is a function you can wrap a condition in.

The hint signals to SQreamDB that the result of the condition will be very sparse, and that it should attempt to rechunk the results into fewer, fuller chunks.

Use the high selectivity hint when you expect a predicate to filter out most values. For example, when the data is dispersed over lots of chunks (meaning that the data is not well-clustered).

For example,

```
SELECT store_name, SUM(amount) FROM store_dim
WHERE HIGH_SELECTIVITY(p_date = '2018-07-01')
GROUP BY 1;
```

This hint tells the query compiler that the `WHERE` condition is expected to filter out more than 60% of values. It never affects the query results, but when used correctly can improve query performance.

Tip: The `HIGH_SELECTIVITY()` hint function can only be used as part of the `WHERE` clause. It can't be used in equijoin conditions, cases, or in the select list.

Read more about identifying the scenarios for the high selectivity hint in our *Monitoring query performance guide*.

4.11.3.4 Avoiding Aggregation Overflow

When using an `INT` or smaller type, the `SUM` and `COUNT` operations return a value of the same type. To avoid overflow on large results, cast the column up to a larger type.

For example

```
SELECT store_name, SUM(amount :: BIGINT) FROM store_dim
GROUP BY 1;
```

4.11.3.5 Prefer `COUNT(*)` and `COUNT` to Non-nullable Columns

SQreamDB optimizes `COUNT(*)` queries very strongly. This also applies to `COUNT(column_name)` on non-nullable columns. Using `COUNT(column_name)` on a nullable column will operate quickly, but much slower than the previous variations.

4.11.3.6 Returning Only Required Columns

Returning only the columns you need to client programs can improve overall query performance. This also reduces the overall result set, which can improve performance in third-party tools.

SQreamDB is able to optimize out unneeded columns very strongly due to its columnar storage.

4.11.3.7 Reducing Recurring Compilation Time

Saved Queries are compiled when they are created. The query plan is saved in SQreamDB's metadata for later re-use.

Saved query plans enable reduced compilation overhead, especially with very complex queries, such as queries with lots of values in an IN predicate.

When executed, the saved query plan is recalled and executed on the up-to-date data stored on disk.

4.11.3.8 Reducing JOIN Complexity

Filter and reduce table sizes prior to joining on them

```
SELECT store_name,
       SUM(amount)
FROM dimention dim
  JOIN fact ON dim.store_id = fact.store_id
WHERE p_date BETWEEN '2019-07-01' AND '2019-07-31'
GROUP BY store_name;
```

Can be rewritten as:

```
SELECT store_name,
       sum_amount
FROM dimention AS dim
  INNER JOIN (SELECT SUM(amount) AS sum_amount,
                     store_id
              FROM fact
              WHERE p_date BETWEEN '2019-07-01' AND '2019-07-31'
              GROUP BY store_id) AS fact ON dim.store_id = fact.store_id;
```

4.11.4 Data Loading Considerations

4.11.4.1 Using Natural Data Sorting

Very often, tabular data is already naturally ordered along a dimension such as a timestamp or area.

This natural order is a major factor for query performance later on, as data that is naturally sorted can be more easily compressed and analyzed with SQreamDB's metadata collection.

For example, when data is sorted by timestamp, filtering on this timestamp is more effective than filtering on an unordered column.

Natural ordering can also be used for effective delete operations.

Use the *Monitoring Query Performance* guide to learn about built-in monitoring utilities. The guide also gives concrete examples for improving query performance.

CHAPTER

FIVE

CONFIGURATION GUIDES

The **Configuration Guides** page describes the following configuration information:

5.1 Configuring SQream

The **Configuring SQream** page describes the following configuration topics:

5.1.1 Cluster and Session

When configuring your SQreamDB environment, you have the option to use flags that apply to either the entire cluster or a specific session. Cluster configuration involve metadata and is persistent. Persistent modifications refer to changes made to a system or component that are saved and retained even after the system is restarted or shut down, allowing the modifications to persist over time. Session flags only apply to a specific session and are not persistent. Changes made using session flags are not visible to other users, and once the session ends, the flags return to their default values.

5.1.1.1 Setting the flags

5.1.1.1.1 Syntax

You may set both cluster and session flags using the following syntax on SQreamDB Acceleration Studio and Console:

Cluster flag syntax:

```
ALTER SYSTEM SET <flagName>
```

Session flag syntax:

```
SET <flagName>
```

5.1.1.1.2 Configuration file

You may set session flags within your *Legacy Configuration File*.

5.1.1.2 Flag List

Flag Name	Who May Configure	Cluster / Session	Description
binSizes	SUPERUSER	Session	Sets the custom bin size in the
cacheEvictionMilliseconds	Anyone	Session	Sets how long the cache stores
cacheDiskDir	Anyone	Session	Sets the ondisk directory locat
cacheDiskGB	Anyone	Session	Sets the amount of memory (
cachePartitions	Anyone	Session	Sets the number of partitions
cachePersistentDir	Anyone	Session	Sets the persistent directory lo
cachePersistentGB	Anyone	Session	Sets the amount of data (GB)
cacheRamGB	Anyone	Session	Sets the amount of memory (
checkCudaMemory	SUPERUSER	Session	Sets the pad device memory a
clientReconnectionTimeout	Anyone	Cluster	Reconnection time out for the
compilerGetsOnlyUFs	SUPERUSER	Session	Sets the runtime to pass only
copyToRestrictUtf8	SUPERUSER	Session	Sets the custom bin size in the
cpuReduceHashtableSize	SUPERUSER	Session	Sets the hash table size of the
csvLimitRowLength	SUPERUSER	Cluster	Sets the maximum supported
cudaMemcpyMaxSizeBytes	SUPERUSER	Session	Sets the chunk size for copyin
CudaMemcpySynchronous	SUPERUSER	Session	Indicates if copying from/to G
developerMode	SUPERUSER	Session	Enables modifying R&D flags
enableDeviceDebugMessages	SUPERUSER	Session	Checks for CUDA errors afte
enableLogDebug	SUPERUSER	Session	Enables creating and logging i
enableNvprofMarkers	SUPERUSER	Session	Activates the Nvidia profiler (
endLogMessage	SUPERUSER	Session	Appends a string at the end of
extentStorageFileSizeMB	SUPERUSER	Cluster	Sets the minimum size in meb
externalTableBlobEstimate	?	Session	?
flipJoinOrder	Anyone	Session	Reorders join to force equijoi
gatherMemStat	SUPERUSER	Session	Monitors all pinned allocation
increaseChunkSizeBeforeReduce	SUPERUSER	Session	Increases the chunk size to re
increaseMemFactors	SUPERUSER	Session	Adds rechunker before expens
leveldbWriteBufferSize	SUPERUSER	Session	Sets the buffer size.
logSysLevel	Anyone	Session	Determines the client log level
maxAvgBlobSizeToCompressOnGpu	Anyone	Session	Sets the CPU to compress col
maxPinnedPercentageOfTotalRAM	SUPERUSER	Session	Sets the maximum percentage
memMergeBlobOffsetsCount	SUPERUSER	Session	Sets the size of memory used
memoryResetTriggerMB	SUPERUSER	Session	Sets the size of memory used
mtRead	SUPERUSER	Session	Splits large reads to multiple s
mtReadWorkers	SUPERUSER	Session	Sets the number of workers to
orcImplicitCasts	SUPERUSER	Session	Sets the implicit cast in orc fil
sessionTag	Anyone	Session	Sets the name of the session ta
spoolMemoryGB	Anyone	Session	Sets the amount of memory (
statementLockTimeout	SUPERUSER	Session	Sets the timeout (seconds) for
useLegacyDecimalLiterals	SUPERUSER	Session	Interprets decimal literals as D
blockNewVarcharObjects	SUPERUSER	Session	Disables the creation of new t
defaultGracefulShutdownTimeoutMinutes	SUPERUSER	Cluster	Used for setting the amount o
limitQueryMemoryGB	SUPERUSER	Cluster	Prevents a query from process

Flag Name	Who May Configure	Cluster / Session	Description
queueTimeoutMinutes	Anyone	Session	Terminates queries that have e
QueryTimeoutMinutes	Anyone	Session	The flag is intended to identify

5.1.2 Workers

Workers can be individually configured using the *worker configuration file*, which allows for persistent modifications to be made. Persistent modification refers to changes made to a system or component that are saved and retained even after the system is restarted or shut down, allowing the modifications to persist over time.

It is worth noting that the worker configuration file is not subject to frequent changes on a daily basis, providing stability to the system's configuration.

Flag Name	Who May Configure	Description	Data Type	Default Value
cudaMemQuota	SUPERUSER	Sets the percentage of total device memory used by your instance of SQream.	uint	90
healerMaxInactivityHours	SUPERUSER	Used for defining the threshold for creating a log recording a slow statement. The log includes information about the log memory, CPU and GPU.	bigint	5
isHealerOn	SUPERUSER	Enables the Query Healer, which periodically examines the progress of running statements and logs statements exceeding the healerMaxInactivityHours flag setting.	boolean	TRUE
limitQueryMemoryGB	Anyone	Prevents a query from processing more memory than the defined value.	uint	100000
loginMaxRetries	SUPERUSER	Sets the permitted log-in attempts.	bigint	5
machineIP	SUPERUSER	Enables you to manually set the reported IP.	string	127.0.0.1
maxConnections	SUPERUSER	Defines the maximum allowed connections per Worker.	bigint	1000
metadataServerPort	SUPERUSER	Sets the port used to connect to the metadata server. SQream recommends using port ranges above 1024 because ports below 1024 are usually reserved, although there are no strict limitations. You can use any positive number (1 - 65535) while setting this flag.	uint	3105
useConfigIP	SUPERUSER	Activates the machineIP (TRUE). Setting this flag to FALSE ignores the machineIP and automatically assigns a local network IP. This cannot be activated in a cloud scenario (on-premises only).	boolean	FALSE

5.1.3 Modification Methods

5.1.3.1 Worker Configuration File

You can modify your configuration using the **worker configuration file (config.json)**. Changes that you make to worker configuration files are persistent. Note that you can only set the attributes in your worker configuration file **before** initializing your SQream worker, and while your worker is active these attributes are read-only.

The following is an example of the default worker configuration file:

```
{
 "cluster": "/home/sqream/sqream_storage",
 "cudaMemQuota": 96,
 "gpu": 0,
 "cpu": -1,
 "legacyConfigFilePath": "sqream_config_legacy.json",
 "licensePath": "/etc/sqream/license.enc",
 "limitQueryMemoryGB": 30,
 "parquetReaderThreads" : 8,
 "machineIP": "127.0.0.1",
 "metadataServerIp": "127.0.0.1",
 "metadataServerPort": 3105,
 "port": 5000,
 "instanceId": "sqream_0_1",
 "portSsl": 5100,
 "initialSubscribedServices": "sqream",
 "useConfigIP": true
}
```

You can access the legacy configuration file from the `legacyConfigFilePath` parameter shown above. If all (or most) of your workers require the same flag settings, you can set the `legacyConfigFilePath` attribute to the same legacy file.

5.1.3.2 Cluster and Session Configuration File

You can modify your configuration using a legacy configuration file.

The Legacy configuration file provides access to the read/write flags. A link to this file is provided in the **legacyConfigFilePath** parameter in the worker configuration file.

The following is an example of the default cluster and session configuration file:

```
{
 "diskSpaceMinFreePercent": 1,
 "DefaultPathToLogs": "/home/sqream/sqream_storage/tmp_logs/",
 "enableLogDebug": false,
 "insertCompressors": 8,
 "insertParsers": 8,
 "isUnavailableNode": false,
 "logBlackList": "webui",
 "logDebugLevel": 6,
 "nodeInfoLoggingSec": 60,
 "useClientLog": true,
 "useMetadataServer": true,
 "spoolMemoryGB": 28,
 "useLogMaxFileSize": false,
 "logMaxFileSizeMB": 20,
```

(continues on next page)

(continued from previous page)

```
 "logFileRotateTimeFrequency": "daily",
 "waitForClientSeconds": 18000
}
```

5.1.4 Parameter Values

Command	Description	Example
SET<flag_name>	Used for modifying flag attributes.	SET enableLogDebug=false
SHOW <flag-name> / ALL	Used to preset either a specific flag value or all flag values.	SHOW <heartbeatInterval>
SHOW ALL LIKE	Used as a wildcard character for flag names.	SHOW <heartbeat*>
SELECT show_conf() ;	Used to print all flags with the following attributes: • Flag name • Default value • Is Developer Mode (Boolean) • Flag category • Flag type	rechunkThreshold,90, true,RND,regular
SELECT show_conf_extended();	Used to print all information output by the show_conf UF command, in addition to description, usage, data type, default value and range.	rechunkThreshold,90, true,RND,regular
show_md_flag UF	Used to show a specific flag/all flags stored in the metadata.	• Example 1: * master=> ALTER SYSTEM SET heartbeatTimeout=111; • Example 2: * master=> select show_md_flag('all'); heartbeatTimeout, 111 • Example 3: * master=> select show_md_flag('heartbeatTimeout'); heartbeatTimeout, 111
ALTER SYSTEM SET <flag-name>	Used for storing or modifying flag attributes in the metadata.	ALTER SYSTEM SET <heartbeatInterval=12;>
ALTER SYSTEM RESET <flag-name / ALL>	Used to remove a flag or all flag attributes from the metadata.	ALTER SYSTEM RESET <heartbeatInterval ALTER SYSTEM RESET ALL>

5.1.5 Showing All Flags in the Catalog Table

SQream uses the **sqream_catalog.parameters** catalog table for showing all flags, providing the scope (default, cluster and session), description, default value and actual value.

The following is the correct syntax for a catalog table query:

```
SELECT * FROM sqream_catalog.parameters;
```

The following is an example of a catalog table query:

```
externalTableBlobEstimate, 100, 100, default,
ascii, ascii, default, Changes the expected encoding for Varchar columns
useCrcForTextJoinKeys, true, true, default,
hiveStyleImplicitStringCasts, false, false, default,
```

5.2 LDAP

Lightweight Directory Access Protocol (LDAP) is an authentication management service used with Microsoft Active Directory and other directory services. Once LDAP authentication has been configured for SQream, authorization for all existing and newly added roles must be handled by the LDAP server, except for the initial system deployment `sqream` role, which was immediately given full control permissions when SQream was initially deployed.

Before integrating SQream with LDAP consider the following:

- If SQream DB is being installed within an environment where LDAP is already configured, it is best practice to ensure that the newly created SQream role names are consistent with existing LDAP user names.
- If SQream DB has been installed and LDAP has not yet been integrated with SQream, it is best practice to ensure that the newly created LDAP user names are consistent with existing SQream role names. Previously existing SQream roles that were mistakenly not configured in LDAP or that have names which are different than in LDAP, will be recreated in SQream as roles that cannot log in, have no permissions, and have no default schema.

- *Configuring SQream roles*
- *Configuring LDAP Authentication*

5.2.1 Configuring SQream roles

Follow this procedure if you already have LDAP configured for your environment.

1. Create a new role:

```
CREATE ROLE <new_role>;
```

2. Grant the new role login permission:

```
GRANT LOGIN TO <new_role>;
```

3. Grant the new role `CONNECT` permission:

```
GRANT CONNECT ON DATABASE <my_database> TO <new_role>;
```

You may also wish to rename SQream roles so that they are consistent with existing LDAP user names.

5.2.2 Configuring LDAP Authentication

- *Configuration Methods*
- *Basic Method*
- *Advanced Method*
- *Disabling LDAP Authentication*

5.2.2.1 Configuration Methods

To configure LDAP authentication for SQream, you may choose one of the following configuration methods:

Method	Description
Basic method	A traditional approach to authentication in which the user provides a username and password combination to authenticate with the LDAP server. In this approach, all users are given access to SQream.
Advanced method	This approach allows for compartmentalization, which means that users can be grouped into categories, and each category can be assigned or denied access to SQream. This allows administrators to control access to SQream.

5.2.2.2 Basic Method

5.2.2.2.1 Flag Attributes

To enable basic LDAP authentication, configure the following cluster flag attributes using the `ALTER SYSTEM SET` command:

Attribute	Description
`authenticationMethod`	Configure an authentication method: `sqream` or `ldap`. To configure LDAP authentication, choose `ldap`
`ldapIpAddress`	Configure the IP address or the Fully Qualified Domain Name (FQDN) of your LDAP server and select a protocol: `ldap` or `ldaps`. Sqream recommends using the encrypted `ldaps` protocol
`ldapConnTimeoutSec`	Configure the LDAP connection timeout threshold (seconds). Default = 30 seconds
`ldapPort`	LDAP server port number.
`ldapAdvancedMode`	Configure either basic or advanced authentication method. Default = `false`
`ldapPrefix`	String to prefix to the user name when forming the DN to bind as, when doing simple bind authentication
`ldapSuffix`	String to append to the user name when forming the DN to bind as, when doing simple bind authentication

5.2.2.2.2 Basic Method Configuration

Only roles with admin privileges or higher may enable LDAP Authentication.

Procedure

1. Set the `authenticationMethod` attribute:

```
ALTER SYSTEM SET authenticationMethod = 'ldap';
```

2. Set the `ldapIpAddress` attribute:

```
ALTER SYSTEM SET ldapIpAddress = '<ldaps://...>';
```

3. Set the `ldapPrefix` attribute:

```
ALTER SYSTEM SET ldapPrefix = '<DN_binding_string_prefix>=';
```

4. Set the `ldapSuffix` attribute:

```
ALTER SYSTEM SET ldapSuffix  = '<DN_binding_string_suffix>';
```

5. To set the `ldapPort` attribute (Optional), run:

```
ALTER SYSTEM SET ldapPort = <port_number>
```

6. To set the `ldapConnTimeoutSec` attribute (Optional), run:

```
ALTER SYSTEM SET ldapConnTimeoutSec = <15>;
```

7. Restart all sqreamd servers.

5.2.2.2.3 Example

After completing the setup above, we can try to bind to a user by a distinguished name. For example, if the DN of the user is:

```
CN=ElonMusk,OU=Sqream Users,DC=sqream,DC=loc
```

We could set the ldapPrefix and ldapSuffix to

```
ALTER SYSTEM SET ldapPrefix = 'CN=';

ALTER SYSTEM SET ldapSuffix  = ',OU=Sqream Users,DC=sqream,DC=loc';
```

Logging in will be possible using the username ElonMusk using sqream client

```
./sqream sql --username=ElonMusk --password=sqream123 --databasename=master --
↪port=5000
```

5.2.2.3 Advanced Method

5.2.2.3.1 Flag Attributes

To enable advanced LDAP authentication, configure the following cluster flag attributes using the `ALTER SYSTEM SET` command:

Attribute	Description
`authentication-Method`	Configure an authentication method: `sqream` or `ldap`. To configure LDAP authentication, choose `ldap`
`ldapIpAddress`	Configure the IP address or the Fully Qualified Domain Name (FQDN) of your LDAP server and select a protocol: `ldap` or `ldaps`. Sqream recommends using the encrypted `ldaps` protocol
`ldapConnTimeout-Sec`	Configure the LDAP connection timeout threshold (seconds). Default = 30 seconds
`ldapPort`	LDAP server port number
`ldapAdvancedMode`	Set `ldapAdvancedMode` = `true`
`ldapBaseDn`	Root DN to begin the search for the user in, when doing advanced authentication
`ldapBindDn`	DN of user with which to bind to the directory to perform the search when doing search + bind authentication
`ldap-BindDnPass-word`	Password for user with which to bind to the directory to perform the search when doing search + bind authentication
`ldapSearchAttribute`	Attribute to match against the user name in the search when doing search + bind authentication. If no attribute is specified, `the uid` attribute will be used
`ldapSearch-Filter`	Filters `ldapAdvancedMode` authentication. `ALTER SYSTEM SET ldapSearchFilter = '(<attribute>=<value>)(<attribute2>=<value2>)(…)';`

5.2.2.3.2 Advanced Method Configuration

Only roles with admin privileges and higher may enable LDAP Authentication.

Procedure

1. Set the `authenticationMethod` attribute:

```
ALTER SYSTEM SET authenticationMethod = 'ldap';
```

2. Set the `ldapAdvancedMode` attribute:

```
ALTER SYSTEM SET ldapAdvancedMode = true;
```

3. Set the `ldapIpAddress` attribute:

```
ALTER SYSTEM SET ldapIpAddress = '<ldaps://<IpAddress>';
```

4. Set the `ldapBindDn` attribute:

```
ALTER SYSTEM SET ldapBindDn = <binding_user_DN>;
```

5. Set the `ldapBindDnPassword` attribute:

```
ALTER SYSTEM SET ldapBindDnPassword = '<binding_user_password>';
```

6. Set the `ldapBaseDn` attribute:

```
ALTER SYSTEM SET ldapBaseDn = '<search_root_DN>';
```

7. Set the `ldapSearchAttribute` attribute:

```
ALTER SYSTEM SET ldapSearchAttribute = '<search_attribute>';
```

8. To set the `ldapSearchFilter` attribute (Optional), run:

```
ALTER SYSTEM SET ldapSearchFilter = '(<attribute>=<value>)(<attribute2>=<value2>)(…)';
```

9. To set the `ldapPort` attribute (Optional), run:

```
ALTER SYSTEM SET ldapPort = <port_number>
```

10. To set the `ldapConnTimeoutSec` attribute (Optional), run:

```
ALTER SYSTEM SET ldapConnTimeoutSec = <15>;
```

11. Restart all sqreamd servers.

5.2.2.3.3 Example

After completing the setup above we can try to bind to a user by locating it by one of its unique attributes.

User DN =

```
CN=ElonMusk,OU=Sqream Users,DC=sqream,DC=loc
```

User has value of elonm for attribute `sAMAccountName`.

```
ALTER SYSTEM SET authenticationMethod = 'ldap';

ALTER SYSTEM SET ldapAdvancedMode = true;

ALTER SYSTEM SET ldapIpAddress = 'ldaps://192.168.10.20';

ALTER SYSTEM SET ldapPort = 5000

ALTER SYSTEM SET ldapBindDn = 'CN=LDAP admin,OU=network admin,DC=sqream,DC=loc';

ALTER SYSTEM SET ldapBindDnPassword = 'sqream123';

ALTER SYSTEM SET ldapBaseDn = 'OU=Sqream Users,DC=sqream,DC=loc';

ALTER SYSTEM SET ldapSearchAttribute = 'sAMAccountName';

ALTER SYSTEM SET ldapConnTimeoutSec = 30;

ALTER SYSTEM SET ldapSearchFilter =  "(memberOf=CN=SqreamGroup,CN=Builtin,DC=sqream,
↪DC=loc)(memberOf=CN=Admins,CN=Builtin,DC=sqream,DC=loc)";
```

Logging in will be possible using the username elonm using sqream client

```
./sqream sql --username=elonm --password=<elonm_password> --databasename=master --
↪port=5000
```

5.2.2.4 Disabling LDAP Authentication

To disable LDAP authentication and configure sqream authentication:

1. Execute the following syntax:

```
ALTER SYSTEM SET authenticationMethod = 'sqream';
```

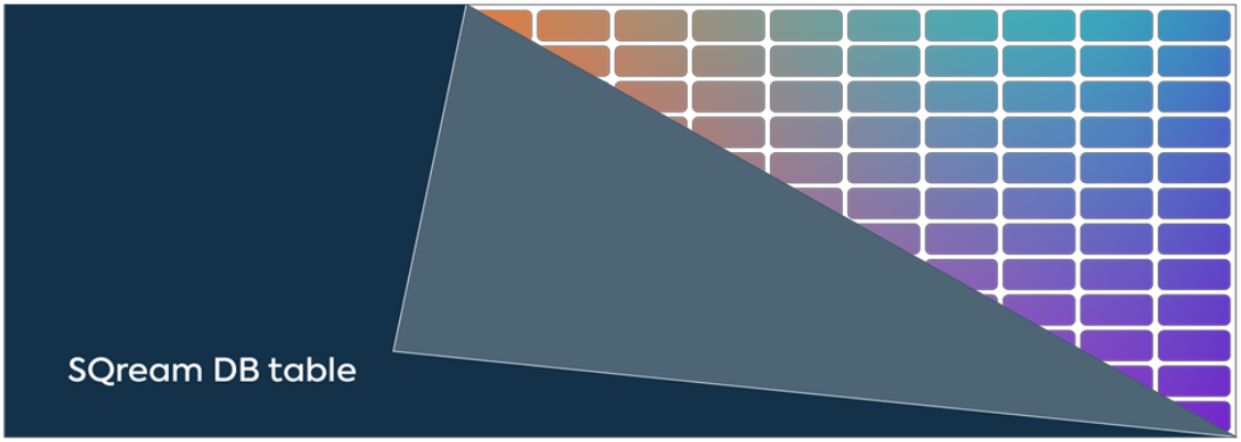
2. Restart all sqreamd servers.

CHAPTER

SIX

ARCHITECTURE

The *Internals and Architecture* and *Filesystem and Usage* guides are walk-throughs for end-users, database administrators, and system architects who wish to get familiarized with the SQreamDB system and its unique capabilities.



6.1 Internals and Architecture

Get to know the SQreamDB key functions and system architecture components, best practices, customization possibilities, and optimizations.

SQreamDB leverages GPU acceleration as an essential component of its core database operations, significantly enhancing columnar data processing. This integral GPU utilization isn't an optional feature but is fundamental to a wide range of data tasks such as `GROUP BY`, scalar functions, `JOIN`, `ORDER BY`, and more. This approach harnesses the inherent parallelism of GPUs, effectively employing a single instruction to process multiple values, akin to the Single-Instruction, Multiple Data (SIMD) concept, tailored for high-throughput operations.

6.1.1 Concurrency and Admission Control

The SQreamDB execution engine employs thread workers and message passing for its foundation. This threading approach enables the concurrent execution of diverse operations, seamlessly integrating IO and GPU tasks with CPU operations while boosting the performance of CPU-intensive tasks.

Learn more about *Sizing*.

6.1.2 Statement Compiler

The Statement Compiler, developed using Haskell, accepts SQL text and generates optimized statement execution plans.

6.1.3 Building Blocks (GPU Workers)

In SQreamDB, the main workload is carried out by specialized C++/CUDA building blocks, also known as Workers, which intentionally lack inherent intelligence and require precise instructions for operation. Effectively assembling these components relies largely on the capabilities of the statement compiler.

6.1.4 Storage Layer

The storage is split into the metadata layer and an append-only data layer.

6.1.4.1 Metadata Layer

Utilizing RocksDB key/value data store, the metadata layer incorporates features such as snapshots and atomic writes within the transaction system, while working in conjunction with the append-only bulk data layer to maintain overall data consistency.

6.1.4.2 Bulk Data Layer Optimization

SQreamDB harnesses the power of its columnar storage architecture within the bulk data layer for performance optimization. This layer employs IO-optimized extents containing compression-enabled CPU and GPU-efficient chunks. Even during small insert operations, SQreamDB maintains efficiency by generating less optimized chunks and extents as needed. This is achieved through background transactional reorganization, such as `DeferredGather`, that doesn't disrupt Data Manipulation Language (DML) operations. Deferred Gather optimizes GPU processing by selectively gathering only the necessary columns after GPU execution, effectively conserving memory and enhancing query performance.

The system initially writes small chunks via small inserts and subsequently reorganizes them, facilitating swift medium-sized insert transactions and rapid queries. This optimization strategy, coupled with SQreamDB's columnar storage, ensures peak performance across diverse data processing tasks.

6.1.5 Transactions

SQreamDB has serializable (auto commit) transactions, with these features:

- Serializable, with any kind of statement
- Run multiple SELECT queries concurrently with anything
- Run multiple inserts to the same table at the same time
- Cannot run multiple statements in a single transaction
- Other operations such as delete, truncate, and DDL use *coarse-grained exclusive locking*.

6.2 Filesystem and Usage

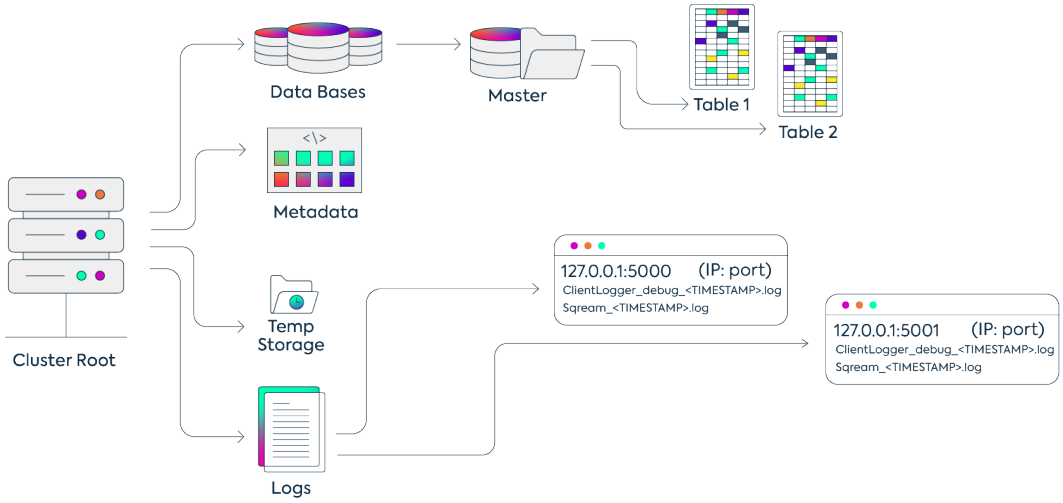
SQreamDB writes and reads data from disk.

The SQreamDB storage directory, sometimes referred to as a **storage cluster** is a collection of database objects, metadata database, and logs.

Each SQreamDB worker and the metadata server must have access to the storage cluster in order to function properly.

6.2.1 Directory organization

The **cluster root** is the directory in which all data for SQreamDB is stored.



SQreamDB storage cluster directories

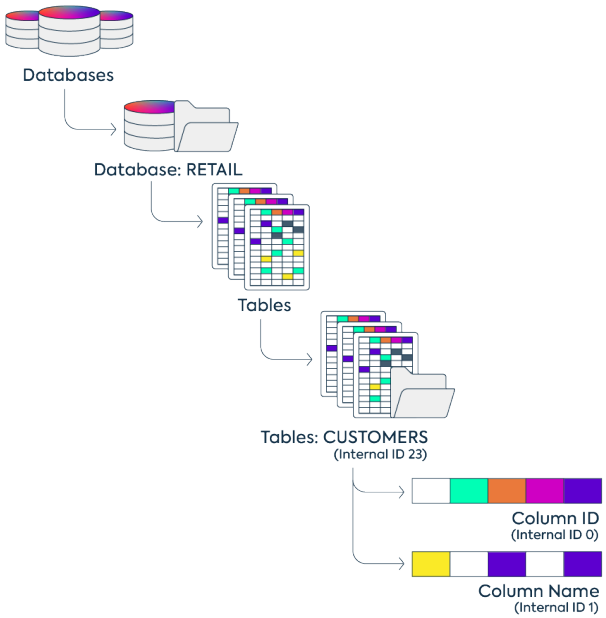
- `databases`
- `metadata` *or* `rocksdb`
- `temp`
- `logs`

6.2.1.1 `databases`

The databases directory houses all of the actual data in tables and columns.

Each database is stored as its own directory. Each table is stored under its respective database, and columns are stored in their respective table.

In the example above, the database named `retail` contains a table directory with a directory named `23`.



Tip: To find table IDs, use a catalog query:

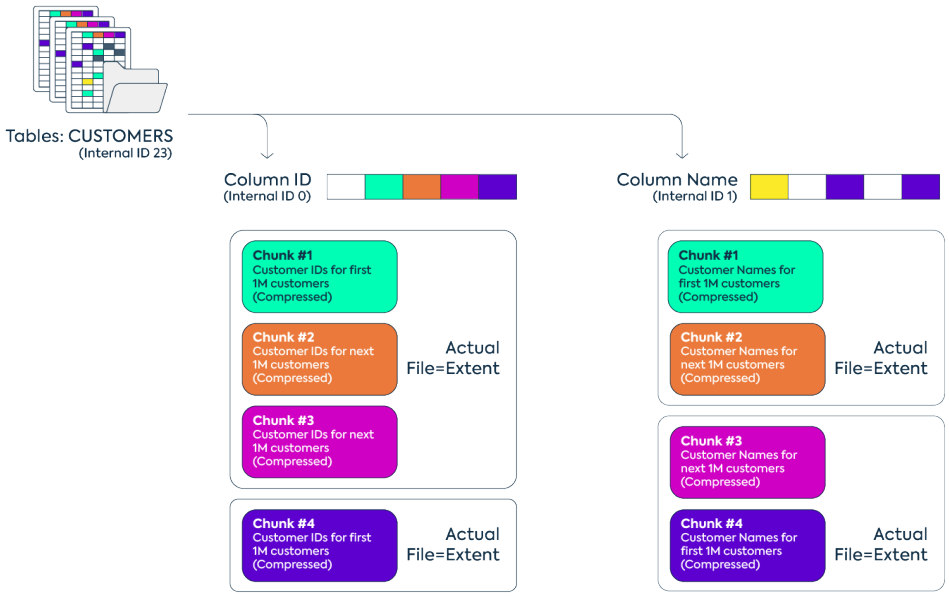
```
master=>
↪ SELECT table_
↪name, table_
↪id FROM sqream_
↪catalog.tables␣
↪WHERE table_name␣
↪= 'customers';
table_
↪name | table_id
----
↪-------+---------
 customers␣
↪|       23
```

Each table directory contains a directory for each physical column. An SQL column may be built up of several physical columns (e.g. if the data type is nullable).

Tip: To find column IDs, use a catalog query:

```
master=> SELECT␣
↪column_id,␣
↪column_name FROM␣
↪sqream_catalog.
↪columns WHERE␣
↪table_id=23;
column_
↪id | column_name
------
↪----+------------
  ␣
↪     0 | name@null
  ␣
↪     1 | name@val
  ␣
↪     2 | age@null
       3 | age@val
    ␣
↪    4 | email@null
   ␣
↪    5 | email@val
```

Each column directory will contain extents, which are collections of chunks.



6.2.1.2 `metadata` or `rocksdb`

SQreamDB's metadata is an embedded key-value store, based on RocksDB. RocksDB helps SQreamDB ensure efficient storage for keys, handle atomic writes, snapshots, durability, and automatic recovery.

The metadata is where all database objects are stored, including roles, permissions, database and table structures, chunk mappings, and more.

6.2.1.3 `temp`

The `temp` directory is where SQreamDB writes temporary data.

The directory to which SQreamDB writes temporary data can be changed to any other directory on the filesystem. SQreamDB recommends remapping this directory to a fast local storage to get better performance when executing intensive larger-than-RAM operations like sorting.

SQreamDB recommends an SSD or NVMe drive, in mirrored RAID 1 configuration.

If desired, the `temp` folder can be redirected to a local disk for improved performance, by setting the `tempPath` setting in the configuration file.

6.2.1.4 `logs`

The logs directory contains logs produced by SQreamDB.

See more about the logs in the *Logging* guide.

6.3 Sizing

6.3.1 Concurrency and Scaling in SQreamDB

A SQreamDB cluster can execute one statement per worker process while also supporting the concurrent operation of multiple workers. Utility functions with minimal resource requirements, such as show_server_status, show_locks, and show_node_info will be executed regardless of the workload.

Minimum Resource Required Per Worker:

Component	CPU Cores	RAM (GB)	Local Storage (GB)
Worker	8	128	10
Metadata Server	16 cores per 100 Workers	20 GB RAM for every 1 trillion rows	10
SqreamDB Acceleration Studio	16	16	50
Server Picker	1	2	

Lightweight queries, such as copy_to and *Clean-Up* require 64 RAM (GB).

Maximum Workers Per GPU:

GPU	Workers
NVIDIA Turing T4 (16GB)	1
NVIDIA Volta V100 (32GB)	2
NVIDIA Ampere A100 (40GB)	3
NVIDIA Ampere A100 (80GB)	6
NVIDIA Hopper H100 (80GB)	6
L40S Ada Lovelace (48GB)	4

Tip: Your GPU is not on the list? Visit SQreamDB Support for additional information.

6.3.1.1 Scaling When Data Sizes Grow

For many statements, SQreamDB scales linearly when adding more storage and querying on large data sets. It uses optimized 'brute force' algorithms and implementations, which don't suffer from sudden performance cliffs at larger data sizes.

6.3.1.2 Scaling When Queries Are Queuing

SQreamDB scales well by adding more workers, GPUs, and nodes to support more concurrent statements.

6.3.1.3 What To Do When Queries Are Slow

Adding more workers or GPUs does not boost the performance of a single statement or query.

To boost the performance of a single statement, start by examining the *best practices* and ensure the guidelines are followed.

Adding additional RAM to nodes, using more GPU

memory, and faster CPUs or storage can also sometimes help.

6.3.2 Spooling Configuration

$$limitQueryMemoryGB = \frac{\text{Total RAM - Internal Operation - metadata Server - Server picke}}{\text{Number of Workers}}$$

$$spoolMemoryGB = limitQueryMemoryGB - 50GB$$

The `limitQueryMemoryGB` flag is the total memory you've allocated for processing queries. In addition, the `limitQueryMemoryGB` defines how much total system memory is used by each worker. Note that `spoolMemoryGB` must bet set to less than the `limitQueryMemoryGB`.

6.3.2.1 Example

6.3.2.1.1 Setting Spool Memory

The provided examples assume a configuration with 2T of RAM, 8 workers running on 2 A100(80GB) GPUs, with 200 GB allocated for Internal Operations, Metadata Server, Server Picker, and UI.

Configuring the `limitQueryMemoryGB` using the Worker configuration file:

```
{
    "cluster":␣
↪"/home/test_user/
↪sqream_testing_
↪temp/sqreamdb",
    "gpu":  0,
    "licensePath":␣
```

(continues on next page)

(continued from previous page)

```
↪"home/test_user/
↪SQream/tests/
↪license.enc",
    "machineIP
↪": "127.0.0.1",
   ␣
↪"metadataServerIp
↪": 127.0.0.1,
␣
↪␣
↪␣
↪␣
↪
↪"metadataServerPort
↪": 3105,
    "port": 5000,
    "useConfigIP
↪": true,
␣
↪␣
↪␣
↪␣
↪
↪"limitQueryMemoryGB
↪" : 225,
}
```

Configuring the `spoolMemoryGB` using the legacy configuration file:

```
{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"diskSpaceMinFreePercent
↪": 10,
     ␣
↪   "enableLogDebug
↪": false,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
```

(continues on next page)

(continued from previous page)

```
↪"insertCompressors
↪": 8,
    ␣
↪   "insertParsers
↪": 8,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"isUnavailableNode
↪": false,
   ␣
↪    "logBlackList
↪": "webui",
    ␣
↪   "logDebugLevel
↪": 6,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"nodeInfoLoggingSec
↪": 60,
   ␣
↪    "useClientLog
↪": true,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"useMetadataServer
↪": true,
    ␣
↪   "spoolMemoryGB
↪": 175,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪
↪"waitForClientSeconds
↪": 18000,
       ␣
↪"enablePythonUdfs
↪": true
}
```

Need help?

Visit SQreamDB Support for additional information.

CHAPTER

SEVEN

ACCELERATION STUDIO

The SQreamDB Acceleration Studio 5.8.0 is a web-based client for use with SQreamDB. Studio provides users with all functionality available from the command line in an intuitive and easy-to-use format. This includes running statements, managing roles and permissions, and managing SQreamDB clusters.

7.1 Getting Started with SQream Acceleration Studio

7.1.1 Setting Up and Starting Studio

When starting Studio, it listens on the local machine on port 8080.

7.1.2 Logging In to Studio

To log in to SQream Studio:

1. Open a browser to the host on **port 8080**.

   For example, if your machine IP address is `192.168.0.100`, insert the IP address into the browser as shown below:

   ```
   $ http://192.168.0.100:8080
   ```

2. Fill in your SQream DB login credentials. These are the same credentials used for *sqream sql* or JDBC.

   When you sign in, the License Warning is displayed.

7.1.3 Navigating Studio's Main Features

When you log in, you are automatically taken to the **Editor** screen. The Studio's main functions are displayed in the **Navigation** pane on the left side of the screen.

From here you can navigate between the main areas of the Studio:

Element	Description
Editor	Lets you select databases, perform statement operations, and write and execute queries.
Logs	Lets you view usage logs.
Roles	Lets you create users and manage user permissions.
Configuration	Lets you configure your instance of SQream.

By clicking the user icon, you can also use it for logging out and viewing the following:

- User information
- Connection type
- SQream version
- SQream Studio version
- License expiration date
- License storage capacity
- Log out

7.2 Executing Statements and Running Queries from the Editor

The **Editor** is used for the following:

- Selecting an active database and executing queries.
- Performing statement-related operations and showing metadata.
- Executing pre-defined queries.
- Writing queries and statements and viewing query results.

The following is a brief description of the Editor panels:

No.	Element	Description
1	*Toolbar*	Used to select the active database you want to work on, limit the number of rows, save query, etc.
2	*Database Tree and System Queries panel*	Shows a hierarchy tree of databases, views, tables, and columns
3	*Statement panel*	Used for writing queries and statements
4	*Results panel*	Shows query results and execution information.

7.2.1 Executing Statements from the Toolbar

You can access the following from the Toolbar pane:

- **Database dropdown list** - select a database that you want to run statements on.

- **Service dropdown list** - select a service that you want to run statements on. The options in the service dropdown menu depend on the database you select from the **Database** dropdown list.

- **Execute** - lets you set which statements to execute. The **Execute** button toggles between **Execute** and **Stop**, and can be used to stop an active statement before it completes:
  - **Statements** - executes the statement at the location of the cursor.
  - **Selected** - executes only the highlighted text. This mode should be used when executing subqueries or sections of large queries (as long as they are valid SQLs).
  - **All** - executes all statements in a selected tab.

- **Format SQL** - Lets you reformat and reindent statements.

- **Download query** - Lets you download query text to your computer.

- **Open query** - Lets you upload query text from your computer.

- **Max Rows** - By default, the Editor fetches only the first 10,000 rows. You can modify this number by selecting an option from the **Max Rows** dropdown list. Note that setting a higher number may slow down your browser if the result is very large. This number is limited to 100,000 results. To see a higher number, you can save the results in a file or a table using the create_table_as command.

For more information on stopping active statements, see the STOP_STATEMENT command.

7.2.2 Performing Statement-Related Operations from the Database Tree

From the Database Tree you can perform statement-related operations and show metadata (such as a number indicating the amount of rows in the table).

The database object functions are used to perform the following:

- The **SELECT** statement - copies the selected table's **columns** into the Statement panel as `SELECT` parameters.

- The **copy** feature - copies the selected table's **name** into the Statement panel.

- The **additional operations** - displays the following additional options:

Function	Description
Insert statement	Generates an INSERT statement for the selected table in the editing area.
Delete statement	Generates a DELETE statement for the selected table in the editing area.
Create Table As statement	Generates a CREATE TABLE AS statement for the selected table in the editing area.
Rename statement	Generates an RENAME TABLE AS statement for renaming the selected table in the editing area.
Adding column statement	Generates an ADD COLUMN statement for adding columns to the selected table in the editing area.
Drop table statement	Generates a `DROP` statement for the selected object in the editing area.
Table DDL	Generates a DDL statement for the selected object in the editing area. To get the entire database DDL, click the icon next to the database name in the tree root.
DDL Optimizer	The *DDL Optimizer* lets you analyze database tables and recommends possible optimizations.

7.2.2.1 Optimizing Database Tables Using the DDL Optimizer

The **DDL Optimizer** tab analyzes database tables and recommends possible optimizations according to SQreamDB's best practices.

As described in the previous table, you can access the DDL Optimizer by clicking the **additional options icon** and selecting **DDL Optimizer**.

The following table describes the DDL Optimizer screen:

Element	Description
Column area	Shows the column **names** and **column types** from the selected table. You can scroll down or to the right/left for long column lists.
Optimization area	Shows the number of rows to sample as the basis for running an optimization, the default setting (1,000,000) when running an optimization (this is also the overhead threshold used when analyzing `TEXT` fields), and the default percent buffer to add to `TEXT` lengths (10%). Attempts to determine field nullability.
Run Optimizer	Starts the optimization process.

Clicking **Run Optimizer** adds a tab to the Statement panel showing the optimized results of the selected object.

For more information, see *Optimization and Best Practices*.

7.2.2.2 Executing Pre-Defined Queries from the System Queries Panel

The **System Queries** panel lets you execute predefined queries and includes the following system query types:

- **Catalog queries** - Used for analyzing table compression rates, users and permissions, etc.

- **Admin queries** - Queries useful for SQreamDB database management.

Clicking an item pastes the query into the Statement pane, and you can undo a previous operation by pressing **Ctrl + Z**.

7.2.3 Writing Statements and Queries from the Statement Panel

The multi-tabbed statement area is used for writing queries and statements, and is used in tandem with the toolbar. When writing and executing statements, you must first select a database from the **Database** dropdown menu in the toolbar. When you execute a statement, it passes through a series of statuses until completed. Knowing the status helps you with statement maintenance, and the statuses are shown in the **Results panel**.

The auto-complete feature assists you when writing statements by suggesting statement options.

The following table shows the statement statuses:

Status	Description
Pending	The statement is pending.
In queue	The statement is waiting for execution.
Initializing	The statement has entered execution checks.
Executing	The statement is executing.
Statement stopped	The statement has been stopped.

You can add and name new tabs for each statement that you need to execute, and Studio preserves your created tabs when you switch between databases. You can add new tabs by clicking , which creates a new tab to the right with a default name of SQL and an increasing number. This helps you keep track of your statements.

You can also rename the default tab name by double-clicking it and typing a new name and write multiple statements in tandem in the same tab by separating them with semicolons (`;`).If too many tabs to fit into the Statement Pane are open at the same time, the tab arrows are displayed. You can scroll through the tabs by clicking or , and close tabs by clicking . You can also close all tabs at once by clicking **Close all** located to the right of the tabs.

7.2.4 Viewing Statement and Query Results from the Results Panel

The results panel shows statement and query results. By default, only the first 10,000 results are returned, although you can modify this from the *Executing Statements from the Toolbar*, as described above. By default, executing several statements together opens a separate results tab for each statement. Executing statements together executes them serially, and any failed statement cancels all subsequent executions.

The following is a brief description of the Results panel views highlighted in the figure above:

Element	Description
Results view	Lets you view search query results.
Execution Details view	Lets you analyze your query for troubleshooting and optimization purposes.
SQL view	Lets you see the SQL view.

7.2.4.1 Searching Query Results in the Results View

The **Results view** lets you view search query results.

From this view you can also do the following:

- View the amount of time (in seconds) taken for a query to finish executing.
- Switch and scroll between tabs.
- Close all tabs at once.
- Enable keeping tabs by selecting **Keep tabs**.
- Sort column results.

7.2.4.1.1 Saving Results to the Clipboard

The **Save results to clipboard** function lets you save your results to the clipboard to paste into another text editor or into Excel for further analysis.

7.2.4.1.2 Saving Results to a Local File

The **Save results to local file** functions lets you save your search query results to a local file. Clicking **Save results to local file** downloads the contents of the Results panel to an Excel sheet. You can then use copy and paste this content into other editors as needed.

In the Results view you can also run parallel statements, as described in **Running Parallel Statements** below.

7.2.4.1.3 Running Parallel Statements

While Studio's default functionality is to open a new tab for each executed statement, Studio supports running parallel statements in one statement tab. Running parallel statements requires using macros and is useful for advanced users.

The following shows the syntax for running parallel statements:

```
$ @@ parallel
$ $$
$ select 1;
$ select 2;
$ select 3;
$ $$
```

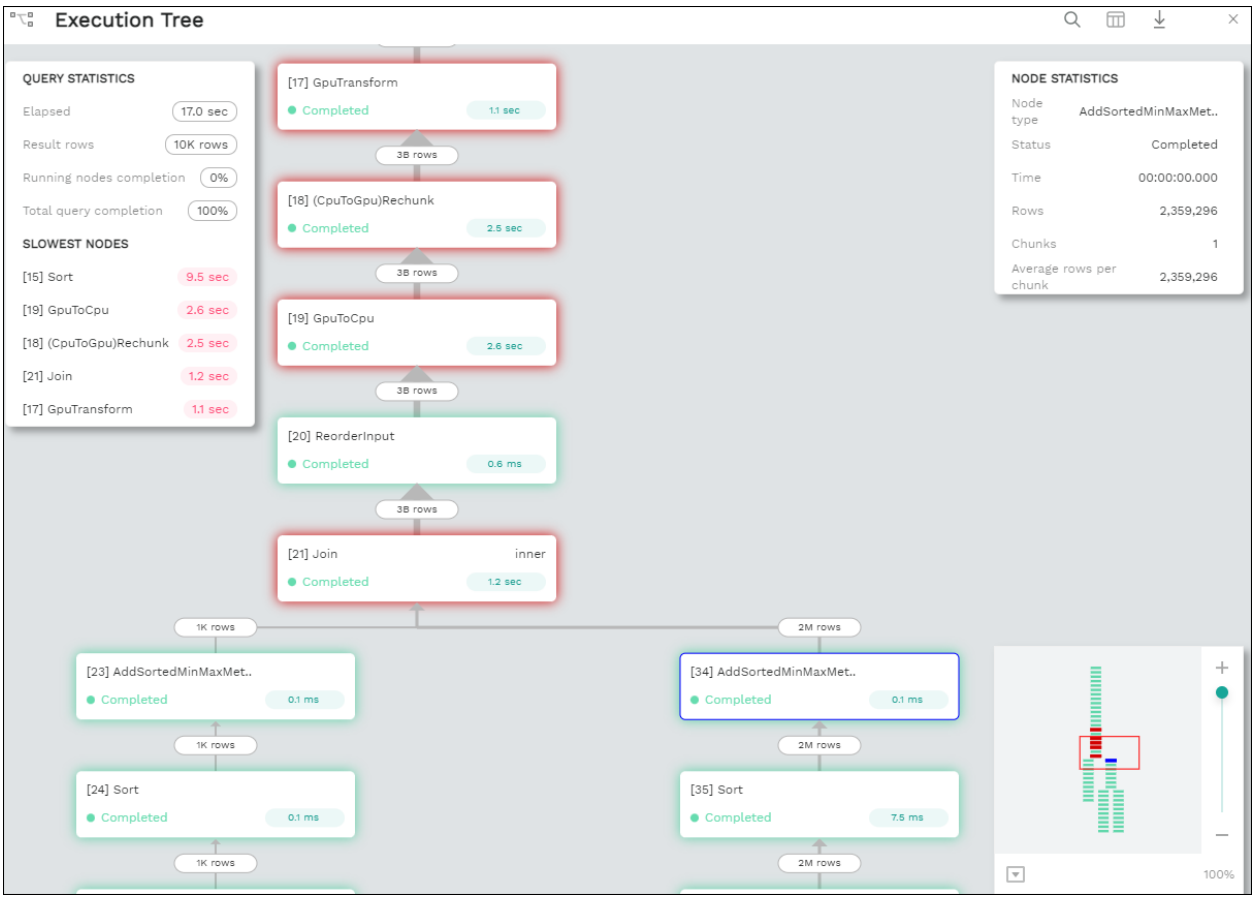
7.2.4.2 Execution Details View

Clicking **Execution Details View** displays the **Execution Tree**, which is a chronological tree of processes that occurred to execute your queries. The purpose of the Execution Tree is to analyze all aspects of your query for troubleshooting and optimization purposes, such as resolving queries with an exceptionally long runtime.

Note: The **Execution Details View** button is enabled only when a query takes longer than five seconds.

- *Viewing Query Statistics*
- *Using the Plain View*

From this screen you can scroll in, out, and around the execution tree with the mouse to analyze all aspects of your query. You can navigate around the execution tree by dragging or by using the mini-map in the bottom right corner.



You can also search for query data by pressing **Ctrl+F** or clicking the search icon in the search field in the top right corner and typing text.

Pressing **Enter** takes you directly to the next result matching your search criteria, and pressing **Shift + Enter** takes you directly to the previous result. You can also search next and previous results using the up and down arrows.

The nodes are color-coded based on the following:

- **Slow nodes** - red
- **In progress nodes** - yellow
- **Completed nodes** - green
- **Pending nodes** - white

- **Currently selected node** - blue
- **Search result node** - purple (in the mini-map)

The execution tree displays the same information as shown in the plain view in tree format.

The Execution Tree tracks each phase of your query in real time as a vertical tree of nodes. Each node refers to an operation that occurred on the GPU or CPU. When a phase is completed, the next branch begins to its right until the entire query is complete. Joins are displayed as two parallel branches merged together in a node called **Join**, as shown in the figure above. The nodes are connected by a line indicating the number of rows passed from one node to the next. The width of the line indicates the amount of rows on a logarithmic scale.

Each node displays a number displaying its **node ID**, its **type**, **table name** (if relevant), **status**, and **runtime**. The nodes are color-coded for easy identification. Green nodes indicate **completed nodes**, yellow indicates **nodes in progress**, and red indicates **slowest nodes**, typically joins, as shown below:

7.2.4.2.1 Viewing Query Statistics

The following statistical information is displayed in the top left corner, as shown in the figure above:

- **Query Statistics**:
  - **Elapsed** - the total time taken for the query to complete.
  - **Result rows** - the amount of rows fetched.
  - **Running nodes completion**
  - **Total query completion** - the amount of the total execution tree that was executed (nodes marked green).
- **Slowest Nodes** information is displayed in the top right corner in red text. Clicking the slowest node centers automatically on that node in the execution tree.

You can also view the following **Node Statistics** in the top right corner for each individual node by clicking a node:

Element	Description
Node type	Shows the node type.
Status	Shows the execution status.
Time	The total time taken to execute.
Rows	Shows the number of produced rows passed to the next node.
Chunks	Shows number of produced chunks.
Average rows per chunk	Shows the number of average rows per chunk.
Table (for **ReadTable** and joins only)	Shows the table name.
Write (for joins only)	Shows the total date size written to the disk.
Read (for **ReadTable** and joins only)	Shows the total data size read from the disk.

Note that you can scroll the Node Statistics table. You can also download the execution plan table in .csv format by clicking the download arrow in the upper-right corner.

7.2.4.2.2 Using the Plain View

You can use the **Plain View** instead of viewing the execution tree by clicking **Plain View** in the top right corner. The plain view displays the same information as shown in the execution tree in table format.

The plain view lets you view a query's execution plan for monitoring purposes and highlights rows based on how long they ran relative to the entire query.

This can be seen in the **timeSum** column as follows:

- **Rows highlighted red** - longest runtime
- **Rows highlighted orange** - medium runtime
- **Rows highlighted yellow** - shortest runtime

7.2.4.3 Viewing Wrapped Strings in the SQL View

The SQL View panel allows you to more easily view certain queries, such as a long string that appears on one line. The SQL View makes it easier to see by wrapping it so that you can see the entire string at once. It also reformats and organizes query syntax entered in the Statement panel for more easily locating particular segments of your queries. The SQL View is identical to the **Format SQL** feature in the Toolbar, allowing you to retain your originally constructed query while viewing a more intuitively structured snapshot of it.

7.3 Viewing Logs

The **Logs** screen is used for viewing logs and includes the following elements:

Element	Description
Filter area	Lets you filter the data shown in the table.
Query tab	Shows basic query information logs, such as query number and the time the query was run.
Session tab	Shows basic session information logs, such as session ID and user name.
System tab	Shows all system logs.
Log lines tab	Shows the total amount of log lines.

7.3.1 Filtering Table Data

From the Logs tab, from the **FILTERS** area you can also apply the **TIMESPAN**, **ONLY ERRORS**, and additional filters (**Add**). The **Timespan** filter lets you select a timespan. The **Only Errors** toggle button lets you show all queries, or only queries that generated errors. The **Add** button lets you add additional filters to the data shown in the table. The **Filter** button applies the selected filter(s).

Other filters require you to select an item from a dropdown menu:

- INFO
- WARNING
- ERROR
- FATAL
- SYSTEM

You can also export a record of all of your currently filtered logs in Excel format by clicking **Download** located above the Filter area.

Back to Viewing Logs

7.3.2 Viewing Query Logs

The **QUERIES** log area shows basic query information, such as query number and the time the query was run. The number next to the title indicates the amount of queries that have been run.

From the Queries area you can see and sort by the following:

- Query ID
- Start time
- Query
- Compilation duration
- Execution duration
- Total duration
- Details (execution details, error details, successful query details)

In the Queries table, you can click on the **Statement ID** and **Query** items to set them as your filters. In the **Details** column you can also access additional details by clicking one of the **Details** options for a more detailed explanation of the query.

Back to Viewing Logs

7.3.3 Viewing Session Logs

The **SESSIONS** tab shows the sessions log table and is used for viewing activity that has occurred during your sessions. The number at the top indicates the amount of sessions that have occurred.

From here you can see and sort by the following:

- Timestamp
- Connection ID
- Username

- Client IP
- Login (Success or Failed)
- Duration (of session)
- Configuration Changes

In the Sessions table, you can click on the **Timestamp**, **Connection ID**, and **Username** items to set them as your filters.

Back to Viewing Logs

7.3.4 Viewing System Logs

The **SYSTEM** tab shows the system log table and is used for viewing all system logs. The number at the top indicates the amount of sessions that have occurred. Because system logs occur less frequently than queries and sessions, you may need to increase the filter timespan for the table to display any system logs.

From here you can see and sort by the following:

- Timestamp
- Log type
- Message

In the Systems table, you can click on the **Timestamp** and **Log type** items to set them as your filters. In the **Message** column, you can also click on an item to show more information about the message.

Back to Viewing Logs

7.3.5 Viewing All Log Lines

The **LOG LINES** tab is used for viewing the total amount of log lines in a table. From here users can view a more granular breakdown of log information collected by Studio. The other tabs (QUERIES, SESSIONS, and SYSTEM) show a filtered form of the raw log lines. For example, the QUERIES tab shows an aggregation of several log lines.

From here you can see and sort by the following:

- Timestamp
- Message level
- Worker hostname
- Worker port
- Connection ID
- Database name
- User name
- Statement ID

In the **LOG LINES** table, you can click on any of the items to set them as your filters.

Back to Viewing Logs

7.4 Creating, Assigning, and Managing Roles and Permissions

In the **Roles** area you can create and assign roles and manage user permissions.

The **Type** column displays one of the following assigned role types:

Role Type	Description
Groups	Roles with no users.
Enabled users	Users with log-in permissions and a password.
Disabled users	Users with log-in permissions and with a disabled password. An admin may disable a user's password permissions to temporary disable access to the system.

Note: If you disable a password, when you enable it you have to create a new one.

Back to Creating, Assigning, and Managing Roles and Permissions

7.4.1 Viewing Information About a Role

Clicking a role in the roles table displays the following information:

- **Parent Roles** - displays the parent roles of the selected role. Roles inherit all roles assigned to the parent.
- **Members** - displays all members that the role has been assigned to. The arrow indicates the roles that the role has inherited. Hovering over a member displays the roles that the role is inherited from.
- **Permissions** - displays the role's permissions. The arrow indicates the permissions that the role has inherited. Hovering over a permission displays the roles that the permission is inherited from.

Back to Creating, Assigning, and Managing Roles and Permissions

7.4.2 Creating a New Role

You can create a new role by clicking **New Role**.

An admin creates a **user** by granting login permissions and a password to a role. Each role is defined by a set of permissions. An admin can also group several roles together to form a **group** to manage them simultaneously. For example, permissions can be granted to or revoked on a group level.

Clicking **New Role** lets you do the following:

- Add and assign a role name (required)
- Enable or disable log-in permissions for the role
- Set a password
- Assign or delete parent roles
- Add or delete permissions
- Grant the selected user with superuser permissions

From the New Role panel you view directly and indirectly (or inherited) granted permissions. Disabled permissions have no connect permissions for the referenced database and are displayed in gray text. You can add or remove permissions from the **Add permissions** field. From the New Role panel you can also search and scroll through the permissions. In the **Search** field you can use the **and** operator to search for strings that fulfill multiple criteria.

When adding a new role, you must select the **Enable login for this role** and **Has password** check boxes.

Back to Creating, Assigning, and Managing Roles and Permissions

7.4.3 Editing a Role

Once you've created a role, clicking the **Edit Role** button lets you do the following:

- Edit role name
- Enable or disable log-in permissions
- Set a password
- Assign or delete parent roles
- Assign a role **administrator** permissions
- Add or delete permissions
- Grant the selected user with superuser permissions

From the Edit Role panel you view directly and indirectly (or inherited) granted permissions. Disabled permissions have no connect permissions for the referenced database and are displayed in gray text. You can add or remove permissions from the **Add permissions** field. From the Edit Role panel you can also search and scroll through the permissions. In the **Search** field you can use the **and** operator to search for strings that fulfill multiple criteria.

Back to Creating, Assigning, and Managing Roles and Permissions

7.4.4 Deleting a Role

Clicking the **delete** icon displays a confirmation message with the amount of users and groups that will be impacted by deleting the role.

Back to Creating, Assigning, and Managing Roles and Permissions

7.5 Configuring Your Instance of SQreams

The **Configuration** section lets you edit parameters from one centralized location. While you can edit these parameters from the **worker configuration file (config.json)** or from your CLI, you can also modify them in Studio in an easy-to-use format.

Configuring your instance of SQream in Studio is session-based, which enables you to edit parameters per session on your own device. Because session-based configurations are not persistent and are deleted when your session ends, you can edit your required parameters while avoiding conflicts between parameters edited on different devices at different points in time.

7.5.1 Editing Your Parameters

When configuring your instance of SQream in Studio you can edit session and cluster parameters only.

Studio includes two types of parameters: toggle switches, such as **flipJoinOrder**, and text fields, such as **logSysLevel**. After editing a parameter, you can reset each one to its previous value or to its default value individually, or revert all parameters to their default setting simultaneously. Note that you must click **Save** to save your configurations.

You can hover over the **information** icon located on each parameter to read a short description of its behavior.

7.5.2 Exporting and Importing Configuration Files

You can also export and import your configuration settings into a .json file. This allows you to easily edit your parameters and to share this file with other users if required.

For more information about configuring your instance of SQream, see Configuration.

CHAPTER

EIGHT

CONNECTING TO SQREAMDB

SQream supports the most common database tools and interfaces, giving you direct access through a variety of drivers, connectors, and visualization tools and utilities.

8.1 Client Platforms

SQreamDB is designed to work with the most common database tools and interfaces, allowing you direct access through a variety of drivers, connectors, visualization tools, and utilities.

8.1.1 Data Integration Tools

Informatica Cloud Services

Pentaho Data Integration and Analytics

Talend

SQL Workbench

8.1.2 Business Intelligence (BI) Tools

Denodo

MicroStrategy

Power BI Desktop

SAP BusinessObjects

SAS Viya

Tableau

TIBCO Spotfire

8.1.3 Data Analysis and Programming Languages

PHP

R

8.1.3.1 Denodo Platform

Denodo Platform is a data virtualization solution that enables integration, access, and real-time data delivery from disparate on-premises and cloud-based sources.

8.1.3.1.1 Before You Begin

It is essential that you have the following

installed:

- Denodo 8.0
- Java 1.8

8.1.3.1.2 Setting Up a Connection to SQreamDB

1. Under `Denodo\DenodoPlatform8.0\lib\extensions\jdbc-drivers`, create a directory named `sqream`.
2. Download the SQreamDB JDBC Connector *.jar file* and save it un-

der the newly created `sqream` directory.

3. In the Denodo Platform menu, go to **File** > **New** > **Data Source** > **JDBC**.

   A connection dialog box is displayed.

4. Under the **Configuration** tab, select the **Con-**

nec-tion tab and fill in the data source in-for-ma-tion:

Field name	Description	Value	Example
Name	The name of the data source	`sqream`	
Database adapter	The database adapter allows Denodo Platform to communicate and interact with SQreamDB	`Generic`	
Driver class path	The path to the location of the JDBC driver required for the connection to the data source		`path/to/jdbcdriver/sqream-jdbc-x.x.x`
Driver class	The class name of the JDBC driver used to connect to the data source	`com.sqream.jdbc.SQDriver`	
Database URI	The URI that specifies the location and details of the database or data source to be connected	`jdbc:Sqream://<host_and_port>/<database_name>;[<optional_parameters>;...]`	
Trans-action isola-tion	The level of isolation used to manage concurrent transactions in the database connection, ensuring data consistency and integrity	`Database default`	
Au-then-tica-tion	Authentication method	`Use login and password`	
Login	The SQreamDB role		`SqreamRole`
Pass-word	The SQreamDB role password		`Sqream-RolePass-word2023`

5. To ver-ify your newly cre-ated con-nec-

tion, select the **Test connection** button.

Note: When adding the JDBC driver in Denodo, it's important to note that a restart of Denodo may be required. Additionally, in some cases, the SQream driver may not immediately appear in the list of available JDBC drivers. If you encounter this issue, a simple solution is to reboot the machine and attempt the process again.

8.1.3.1.3 Limitation

When working with table joins involving columns with identical names and exporting a view as a REST service, the query transformation process can introduce ambiguity due to the indistinguishable column identifiers. This ambiguity may result in unresolved column references during query execution, necessitating thoughtful aliasing or disambiguation strategies to ensure accurate results.

8.1.3.2 Informatica Cloud Services

8.1.3.2.1 Overview

The **Connecting to SQream Using Informatica Cloud Services** page is quick start guide for connecting to SQream using Informatica cloud services.

It describes the following:

- *Establishing a Connection between SQream*

and In-for-mat-ica

- *Establishing a Con-nec-tion In Your En-vi-ron-ment*

- *Establishing an ODBC DSN Con-nec-tion In Your En-vi-ron-ment*

- *Establishing a JDBC Con-nec-tion In Your En-vi-ron-ment*

- *Supported SQream Driver Ver-sions*

8.1.3.2.1.1 Establishing a Connection between SQream and Informatica

The **Establishing a Connection between SQream and Informatica** page describes how to establish a connection between SQream and the Informatica data integration Cloud.

To establish a connection between SQream and

the Informatica data integration Cloud:

1. Go to the Informatica Cloud homepage.

2. Do one of the following:

1. Log in using your credentials.

2. Log in using your SAML Iden-

tity Provider.

3. From the **Services** window, select **Administrator** or click **Show all services** to show all services.

   The SQream dashboard is displayed.

4. In the menu on the left, click **Runtime Environments**.

   The

Runtime Environments panel is displayed.

5. Click **Download Secure Agent**.

6. When the **Download the Secure Agent** panel is displayed, do the following:

1. Select a platform (Windows 64 or Linux 64).

2. Click **Copy** and save the token on your local hard drive.

   The token is used in combination with your user name to authorize the agent to access your account.

7. Click **Download**.

   The installation be-

gins.

8. When the **Informatica Cloud Secure Agent Setup** panel is displayed, click **Next**.

9. Provide your **User Name** and **Install Token** and click **Register**.

10. From the Runtime Environments panel,

click **New Runtime Environment**.

The **New Secure Agent Group** window is displayed.

11. On the New Secure Agent Group window, click **OK** to connect your Runtime Environment with the running agent.

Note: If you

do not download Secure Agent, you will not be able to connect your Runtime Environment with the running agent and continue establishing a connection between SQream and the Informatica data integration Cloud.

8.1.3.2.1.2 Establishing a Connection In Your Environment

The **Establishing a Connection In Your Environment**

describes the following:

- *Establishing an ODBC DSN Connection In Your Environment*
- *Establishing a JDBC Connection In Your Environment*

8.1.3.2.1.3 Establishing an ODBC DSN Connection In Your Environment

After establishing a connection between SQream and Informatica you can establish an ODBC DSN connection in your environment.

To establish an ODBC connection in your en-

vi-ron-ment:

1. Click **Add**.

2. Click **Con-fig-ure**.

   Note: Ver-ify that **Use Server Picker** is se-lected.

3. Click **Test**.

4. Ver-ify that the con-nec-tion has tested suc-cess-fully.

5. Click **Save**.

6. Click **Actions > Publish**.

8.1.3.2.1.4 Establishing a JDBC Connection In Your Environment

After establishing a connection between SQream and Informatica you can establish a JDBC connection in your environment.

To es-

tablish a JDBC connection in your environment:

1. Create a new DB connection by clicking **Connections > New Connection**.

   The **New Connection** window is displayed.

2. In the **JDBC_IC Connection**

Properties section, in the **JDBC Connection URL** field, establish a JDBC connection by providing the correct connection string.

For connection string examples, see Connection Strings.

3. Click **Test**.

4. Verify that the connection has tested suc-

cessfully.

5. Click **Save**.

6. Click **Actions** > **Publish**.

8.1.3.2.1.5 Supported SQream Driver Versions

SQream supports the following SQream driver versions:

- **JDBC** -

  Version 4.3.4 and above.

- **ODBC** -

Version 4.0.0 and above.

8.1.3.3 MicroStrategy

8.1.3.3.1 Overview

This document is a Quick Start Guide that describes how to install MicroStrategy and connect a datasource to the MicroStrategy dasbhoard for analysis.

The **Connecting to SQream Using MicroStrategy** page describes

the following:

- *What is MicroStrategy?*
- *Connecting a Data Source*
- *Supported SQream Drivers*

8.1.3.3.1.1 What is MicroStrategy?

MicroStrategy is a Business Intelligence software offering a wide vari-

ety of data analytics capabilities. SQream uses the MicroStrategy connector for reading and loading data into SQream.

MicroStrategy provides the following:

- Data discovery
- Advanced analytics
- Data visualization
- Embedded BI
- Banded reports and statements

For more in-

formation about MicroStrategy, see MicroStrategy.

Back to Overview

8.1.3.3.1.2 Connecting a Data Source

1. Activate the **MicroStrategy Desktop** app. The app displays the Dossiers panel to the right.

2. Download the most

current version of the SQream JDBC driver.

3. Click **Dossiers** and **New Dossier**. The **Untitled Dossier** panel is displayed.

4. Click **New Data**.

5. From the **Data Sources** panel, select **Databases** to access data from tables. The

Select Import Options panel is displayed.

6. Select one of the following:

- Build a Query
- Type a Query
- Select Tables

7. Click **Next**.

8. In the Data Source panel, do the

following:

1. From the **Database** drop-down menu, select **Generic**. The **Host Name**, **Port Number**, and **Database Name** fields are removed from the panel.

2. In the **Version** drop-down menu, verify that **Generic DBMS** is selected.

3. Click

Show Connection String.

4. Select the **Edit connection string** checkbox.

5. From the **Driver** drop-down menu, select a driver for one of the following connectors:

- **JDBC** -

  The SQream driver is not in-

te-
grated
with
Mi-
croS-
trat-
egy
and
does
not
ap-
pear
in
the
drop-
down
menu. However, to proceed, you must select an item, and in the next step you must specify the path to the SQream driver that you installed on your machine.

- **ODBC**
-

SQreamDB
ODBC

6. In
the
**Con-
nec-
tion
String**
text
box,
type
the
rel-
e-
vant
con-
nec-
tion
string
and
path
to
the
JDBC
jar
file
us-

ing the following syntax:

```
↪$␣
↪jdbc:Sqream:/
↪/
↪
↪<host␣
↪and␣
↪port>
↪/
↪
↪<database␣
↪name>
↪;
↪user=
↪<username>
↪;
↪password=
↪<password>
↪sqream;
↪[
↪<optional␣
↪parameters>
↪;
↪␣
↪.
↪.
↪.]
```

The following example shows the correct syntax for the JDBC connector:

```
jdbc;
↪MSTR_
↪JDBC_
↪JAR_
↪FOLDER=C:\
↪path\
```

(continues on next page)

(continued from previous page)

```
↪to\
↪jdbc\
↪folder;
↪DRIVER=
↪<driver>
↪;
↪URL=
↪{jdbc:Sqream:/
↪/
↪
↪<host␣
↪and␣
↪port>
↪/
↪
↪<database␣
↪name>
↪;
↪user=
↪<username>
↪;password=<password>;[<optional parameters>; ...];}
```

The following example shows the correct syntax for the ODBC connector:

```
odbc:Driver=
↪{SqreamODBCDriver}
↪;
↪DSN=
↪{SQreamDB␣
↪ODBC}
↪;
↪Server=
↪<Host>
↪;
↪Port=
↪<Port>
↪;
```

(continues on next page)

(continued from previous page)

```
↪Database=
↪<database␣
↪name>
↪;
↪User=
↪<username>
↪;
↪Password=
↪<password>
↪;
↪Cluster=
↪<boolean>
↪;
↪
```

For more information about the available **connection parameters** and other examples, see *Connection Parameters*.

7. In the **User** and **Password** fields, fill out your

user name and password.

8. In the **Data Source Name** field, type **SQreamDB**.

9. Click **Save**. The SQreamDB that you picked in the Data Source panel is displayed.

9. In the **Namespace** menu, select a namespace. The tables files are dis-

played.

10. Drag and drop the tables into the panel on the right in your required order.

11. **Recommended** - Click **Prepare Data** to customize your data for analysis.

12. Click **Finish**.

13. From the **Data Access Mode** dialog box, select one of the following:

- Connect Live
- Import as an In-memory Dataset

Your populated dashboard is displayed and is ready for data discovery and analy-

lytics.

Back to Overview

8.1.3.3.1.3 Supported SQream Drivers

The following list shows the supported SQream drivers and versions:

- **JDBC** - Version 4.3.3 and higher.
- **ODBC** - Version 4.0.0.

Back to Overview

8.1.3.4 Pentaho Data Integration

8.1.3.4.1 Overview

This document is a Quick Start Guide that describes how to install Pentaho, create a transformation, and define your output.

The Connecting to SQream Using Pentaho page describes the following:

- *Installing Pentaho*
- *Installing and setting up the JDBC driver*
- *Creating a transformation*
- *Defining your output*
- *Importing your data*

8.1.3.4.1.1 Installing Pentaho

To install PDI, see the Pentaho Commu-

nity Edition (CE) Installation Guide.

The **Pentaho Community Edition (CE) Installation Guide** describes how to do the following:

- Downloading the PDI software.
- Installing the **JRE (Java Runtime Environment)**

and **JDK (Java Development Kit)**.

- Setting up the JRE and JDK environment variables for PDI.

Back to Overview

8.1.3.4.1.2 Installing and Setting Up the JDBC Driver

After installing Pentaho you must install and set up the JDBC

driver. This section explains how to set up the JDBC driver using Pentaho. These instructions use Spoon, the graphical transformation and job designer associated with the PDI suite.

You can install the driver by copying and pasting the SQream JDBC .jar file into your **<directory>/design-tools/data-integration/lib** directory.

Back to Overview

8.1.3.4.1.3 Creating a Transformation

After in-

stalling Pentaho you can create a transformation.

To create a transformation:

1. Use the CLI to open the PDI client for your operating system (Windows):

```
↪$␣
↪spoon.
↪bat
```

2. Open the spoon.bat file from its folder

location.

3. In the **View** tab, right-click **Transformations** and click **New**.

   A new transformation tab is created.

4. In the **Design** tab, click **Input** to show its file contents.

5. Drag and drop the **CSV file input** item

to the new transformation tab that you created.

6. Double-click **CSV file input**. The **CSV file input** panel is displayed.

7. In the **Step name** field, type a name.

8. To the right of the **Filename** field, click **Browse**.

9. Select the

file that you want to read from and click **OK**.

10. In the CSV file input window, click **Get Fields**.

11. In the **Sample data** window, enter the number of lines you want to sample and click **OK**. The default setting is **100**.

    The tool reads

the file and suggests the field name and type.

12. In the CSV file input window, click **Preview**.

13. In the **Preview size** window, enter the number of rows you want to preview and click **OK**. The default setting is **1000**.

14. Verify

that the preview data is correct and click **Close**.

15. Click **OK** in the **CSV file input** window.

Back to Overview

8.1.3.4.1.4 Defining Your Output

After creating your transformation you must define your output.

To define

your output:

1. In the **Design** tab, click **Output**.

   The Output folder is opened.

2. Drag and drop **Table output** item to the Transformation window.

3. Double-click **Table output** to open the **Table output** dialog

box.

4. From the **Table output** dialog box, type a **Step name** and click **New** to create a new connection.
   Your **steps** are the building blocks of a transformation, such as file input or a table output.

   The **Database Connection** window is displayed with the **General** tab selected by default.

5. Enter or

select the following information in the Database Connection window and click **Test**.

The following table shows and describes the information that you need to fill out in the Database Connection window:

No.	Element Name	Description
1	Connection name	Enter a name that uniquely describes your connection, such as **sample-data**.
2	Connection type	Select **Generic database**.
3	Access	Select **Native (JDBC)**.
4	Custom connection URL	Insert **jdbc:Sqream://<host:port>/<database name>;user=<username>;password=<password>;[<optional parameters>; …];**. The IP is a node in your SQream cluster and is the name or schema of the database you want to connect to. Verify that you have not used any leading or trailing spaces.
5	Custom driver class name	Insert **com.sqream.jdbc.SQDriver**. Verify that you have not used any leading or trailing spaces.
6	Username	Your SQreamdb username. If you leave this blank, you will be prompted to provide it when you connect.
7	Password	Your password. If you leave this blank, you will be prompted to provide it when you connect.

6. Click **OK** in the window above, in the Database Connection window, and Table Output window.

Back to Overview

8.1.3.4.1.5 Importing Data

After defining your output you can begin importing your data.

For more information about backing up users, permissions, or schedules, see Backup and Restore Pentaho Repositories

To import data:

1. Double-

click the **Table output** connection that you just created.

2. To the right of the **Target schema** field, click **Browse** and select a schema name.

3. Click **OK**. The selected schema name is displayed in the **Target schema** field.

4. Create

a new hop connection between the **CSV file input** and **Table output** steps:

1. On the CSV file input step item, click the **new hop connection** icon.
2. Drag an arrow from the **CSV file input** step item to the **Table**

output step item.

3. Release the mouse button. The following options are displayed.

4. Select **Main output of step**.

5. Double-click **Table output** to open the **Table output** dialog box.

6. In the **Target ta-**

ble field, define a target table name.

7. Click **SQL** to open the **Simple SQL editor.**

8. In the **Simple SQL editor**, click **Execute**.

   The system processes and displays the results of the SQL statements.

9.

Close all open dialog boxes.

10. Click the play button to execute the transformation.

    The **Run Options** dialog box is displayed.

11. Click **Run**.

    The **Execution Results** are displayed.

    Back to Overview

8.1.3.5 BI Desktop

Power BI Desktop lets you connect to SQream and use underlying data as with other data sources in Power BI Desktop.

SQream integrates with Power BI Desktop to do the following:

- Extract and transform your

datasets into usable visual models in approximately one minute.

- Use **DAX** functions **(Data Analysis Expressions)** to analyze your datasets.

- Refresh datasets as needed or by using scheduled jobs.

SQream

uses Power BI for extracting data sets using the following methods:

- **Direct query** - Direct queries let you connect easily with no errors, and refresh Power BI artifacts, such as graphs and reports, in a considerable amount of time in relation to the time taken for queries to run using the *SQream SQL CLI Reference guide*.

- **Import** -

Lets you extract datasets from remote databases.

The **Connect to SQream Using Power BI** page describes the following:

- *Prerequisites*
- *Installing Power BI Desktop*
- *Best Practices for Power BI*

8.1.3.5.1 Prerequisites

To connect to SQream, the following must be installed:

- **ODBC data source administrator** - 32 or 64, depending on your operating system. For Windows users, the ODBC data source administrator is embedded within the operating system.
- **SQream driver** - The SQream application

required for interacting with the ODBC according to the configuration specified in the ODBC administrator tool.

8.1.3.5.2 Installing Power BI Desktop

To install Power BI Desktop:

1. Download Power BI Desktop 64x.

2. Download and configure your ODBC driver.

   For

information about downloading and configuring your ODBC driver, see *ODBC* or contact SQream Support.

3. Navigate to **Windows > Documents** and create a folder named **Power BI Desktop** with a subfolder named **Custom Connectors**.

4. From the Client Drivers page, *download* the **PowerQuery.mez** file.

5. Save the PowerQuery.mez file in the **Custom Connectors** folder you created in Step 3.

6. Open the Power BI application.

7. Nav-

i-
gate
to
File
>
Op-
tions
and
Set-
tings
>
Op-
tion
>
Se-
cu-
rity
>
Data
Ex-
ten-
sions,
and
se-
lect **(Not Recommended) Allow any extension to load without validation or warning**.

8. Restart the Power BI Desktop application.

9. From the **Get Data** menu, select **SQream**.

10. Click **Connect** and provide the information shown in the following table:

Element Name	Description
Server	Provide the network address to your database server. You can use a hostname or an IP address.
Port	Provide the port that the database is responding to at the network address.
Database	Provide the name of your database or the schema on your database server.
User	Provide a SQreamdb username.
Passwords	Provide a password for your user.

11. Under **Data Connectivity mode**, select **Direct-Query mode**.

12. Click **Connect**.

13. Provide your user name and password and click **Connect**.

8.1.3.5.3 Best Practices for Power BI

SQream recommends using Power BI in the following ways for acquiring the best performance metrics:

- Creating bar,

pie, line, or plot charts when illustrating one or more columns.

- Displaying trends and statuses using visual models.

- Creating a unified view using **PowerQuery** to connect different data

sources into a single dashboard.

-

8.1.3.6 R

You can use R to interact with a SQream DB cluster.

This tutorial is a guide that will show you how to connect R to SQream DB.

In this topic:

- *JDBC*

  A full example

- *ODBC*

  A full example

8.1.3.6.1 JDBC

1. Get the *SQream DB JDBC driver*.
2. In R, install RJDBC

```
>
↪␣
↪install.
↪packages(
↪"RJDBC
↪")
Installing␣
↪package␣
↪into␣
↪
↪'C:/
↪Users/
↪r/
↪.
↪.
↪.
↪
↪'
```

(continues on next page)

(continued from previous page)

```
(as␣
↪
↪'lib
↪'␣
↪is␣
↪unspecified)

package␣
↪
↪'RJDBC
↪'␣
↪successfully␣
↪unpacked␣
↪and␣
↪MD5␣
↪sums␣
↪checked
```

3. Import the RJDBC library

```
>
↪␣
↪library(RJDBC)
```

4. Set the classpath and initialize the JDBC driver which was previously installed. For example, on Win-

dows:

```
>
↪␣
↪cp␣
↪=␣
↪c(
↪"C:\
↪\
↪Program␣
↪Files\
↪\
↪SQream␣
↪Technologies\
↪\
↪JDBC␣
↪Driver\
↪\
↪2020.
↪1-
↪3.
↪2.
↪0\
↪\
↪sqream-
↪jdbc-
↪3.
↪2.
↪jar")
>
↪␣
↪.
↪jinit(classpath=cp)
>
↪␣
↪drv␣
↪
↪<-
↪␣
↪JDBC(
↪"com.
↪sqream.
↪jdbc.
↪SQDriver
↪",
↪
↪"C:\
↪\
↪Program␣
↪Files\
↪\
↪SQream␣
↪Technologies\
↪\
↪JDBC␣
↪Driver\
↪\
↪2020.
```

(continues on next page)

(continued from previous page)

```
↪1-
↪3.2.0\\sqream-jdbc-3.2.jar")
```

5. Open a connection with a JDBC connection string and run your first statement

```
>
↪␣
↪con␣
↪
↪<-
↪␣
↪dbConnect(drv,
↪
↪"jdbc:Sqream:/
↪/
↪127.
↪0.
↪0.
↪1:3108/
↪master;
↪user=rhendricks;
↪password=Tr0ub4dor&
↪3;
↪cluster=true
↪")

>
↪␣
↪dbGetQuery(con,
↪
↪"select␣
↪top␣
↪5␣
↪*␣
↪from␣
↪t
↪")
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪␣
↪xint␣
↪␣
↪xtinyint␣
↪xsmallint␣
↪xbigint
1␣
↪␣
↪␣
↪␣
↪1␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪82␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5067␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1
2␣
↪␣
↪␣
↪␣
↪2␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪14␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1756␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪2
3␣
↪␣
↪␣
↪␣
↪3␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪91␣
↪␣
↪␣
↪␣
↪␣
↪22356␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪3
4␣
↪␣
↪␣
↪␣
↪4␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪84␣
↪␣
↪␣
↪␣
↪␣
↪17232␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪4
5␣
↪␣
↪␣
↪␣
↪5␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪13␣
↪␣
↪␣
↪␣
↪␣
↪14315␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5
```

6. Close the connection

```
>
↪␣
↪close(con)
```

8.1.3.6.1.1 A full example

```
>
↪␣
↪library(RJDBC)
>
↪␣
↪cp␣
↪=␣
↪c(
↪"C:\
↪\
↪Program␣
↪Files\
↪\
↪SQream␣
↪Technologies\
↪\
↪JDBC␣
↪Driver\
↪\
↪2020.
```

(continues on next page)

(continued from previous page)

```
↪1-
↪3.
↪2.
↪0\
↪\
↪sqream-
↪jdbc-
↪3.
↪2.
↪jar")
>
↪␣
↪.
↪jinit(classpath=cp)
>
↪␣
↪drv␣
↪
↪<-
↪␣
↪JDBC(
↪"com.
↪sqream.
↪jdbc.
↪SQDriver
↪",
↪
↪"C:\
↪\
↪Program␣
↪Files\
↪\
↪SQream␣
↪Technologies\
↪\
↪JDBC␣
↪Driver\
↪\
↪2020.
↪1-
↪3.2.0\\sqream-jdbc-3.2.jar")
>
↪␣
↪con␣
↪
↪<-
↪␣
↪dbConnect(drv,
↪
↪"jdbc:Sqream:/
↪/
↪127.
↪0.
↪0.
↪1:3108/
↪master;
↪user=rhendricks;
```

(continues on next page)

(continued from previous page)

```
↪password=Tr0ub4dor&
↪3;
↪cluster=true
↪")
>
↪␣
↪dbGetQuery(con,
↪
↪"select␣
↪top␣
↪5␣
↪*␣
↪from␣
↪t
↪")
␣
↪␣
↪␣
↪xint␣
↪␣
↪xtinyint␣
↪xsmallint␣
↪xbigint
1␣
↪␣
↪␣
↪␣
↪1␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪82␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5067␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1
2␣
↪␣
↪␣
↪␣
↪2␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪14␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1756␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪2
3␣
↪␣
↪␣
↪␣
↪3␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪91␣
↪␣
↪␣
↪␣
↪␣
↪22356␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪3
4␣
↪␣
↪␣
↪␣
↪4␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪84␣
↪␣
↪␣
↪␣
↪␣
↪17232␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪4
5␣
↪␣
↪␣
↪␣
↪5␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪13␣
↪␣
↪␣
↪␣
↪␣
↪14315␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5
>
↪␣
↪close(con)
```

8.1.3.6.2 ODBC

1. Install the *SQreamDB ODBC driver* for your operating system, and

create a DSN.

2. In R, install RODBC

```
>
↪␣
↪install.
↪packages(
↪"RODBC
↪")
Installing␣
↪package␣
↪into␣
↪
↪'C:/
↪Users/
↪r/
↪.
↪.
↪.
↪
↪'
(as␣
↪
↪'lib
↪'␣
↪is␣
↪unspecified)

package␣
↪
↪'RODBC
↪'␣
↪successfully␣
↪unpacked␣
↪and␣
↪MD5␣
↪sums␣
↪checked
```

3. Import the RODBC library

```
>
```

(continues on next page)

(continued from previous page)

```
↪␣
↪library(RODBC)
```

4. Open a connection handle to an existing DSN (`my_cool_dsn` in this example)

```
>
↪␣
↪ch␣
↪
↪<-
↪␣
↪odbcConnect(
↪"my_
↪cool_
↪dsn
↪",
↪believeNRows=F)
```

5. Run your first statement

```
>
↪␣
↪sqlQuery(ch,
↪
↪"select␣
↪top␣
↪5␣
↪*␣
↪from␣
↪t
```

(continues on next page)

(continued from previous page)

```
↪")
␣
↪␣
↪␣
↪xint␣
↪␣
↪xtinyint␣
↪xsmallint␣
↪xbigint
1␣
↪␣
↪␣
↪␣
↪1␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪82␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5067␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1
2␣
↪␣
↪␣
↪␣
↪2␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪14␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1756␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪2
3␣
↪␣
↪␣
↪␣
↪3␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪91␣
↪␣
↪␣
↪␣
↪␣
↪22356␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪3
4␣
↪␣
↪␣
↪␣
↪4␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪84␣
↪␣
↪␣
↪␣
↪␣
↪17232␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪4
5␣
↪␣
↪␣
↪␣
↪5␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪13␣
↪␣
↪␣
↪␣
↪␣
↪14315␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5
```

6. Close the connection

```
>
↪␣
↪close(ch)
```

8.1.3.6.2.1 A full example

```
>
↪␣
↪library(RODBC)
>
↪␣
↪ch␣
↪
↪<-
↪␣
↪odbcConnect(
↪"my_
↪cool_
↪dsn
↪",
↪believeNRows=F)
>
↪␣
↪sqlQuery(ch,
↪
```

(continues on next page)

(continued from previous page)

```
↪"select␣
↪top␣
↪5␣
↪*␣
↪from␣
↪t
↪")
␣
↪␣
↪␣
↪xint␣
↪␣
↪xtinyint␣
↪xsmallint␣
↪xbigint
1␣
↪␣
↪␣
↪␣
↪1␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪82␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5067␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪1
2␣
↪␣
↪␣
↪␣
↪2␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪14␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪1756␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪2
3␣
↪␣
↪␣
↪␣
↪3␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪91␣
↪␣
↪␣
↪␣
↪␣
↪22356␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪3
4␣
↪␣
↪␣
↪␣
↪4␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪84␣
↪␣
↪␣
↪␣
↪␣
↪17232␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪4
5␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪5␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪13␣
↪␣
↪␣
↪␣
↪␣
↪14315␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪5
>
↪␣
↪close(ch)
```

8.1.3.7 SAP BusinessObjects

The **Connecting to SQream Using SAP BusinessObjects** guide includes the following

sec-
tions:

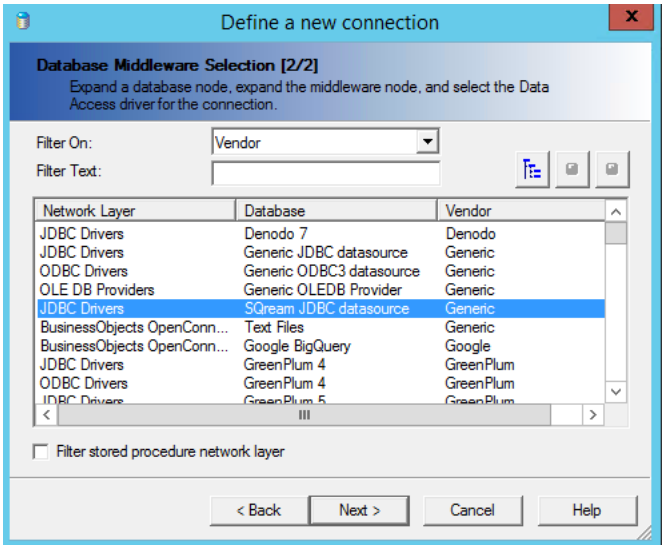
- *Overview*
- *Establishing a New Connection Using a Generic JDCB Connector*

8.1.3.7.1 Overview

The **Connecting to SQream Using SAP BusinessObjects** guide describes the best practices for configuring a connection between SQream and the SAP BusinessObjects BI platform. SAP BO's multi-tier architecture includes

both client and server components, and this guide describes integrating SQream with SAP BO's object client tools using a generic JDBC connector. The instructions in this guide are relevant to both the **Universe Design Tool (UDT)** and the **Information Design Tool (IDT)**. This document only covers how to establish a connection using the generic out-of-the-box JDBC connectors, and does not cover related business object products, such as the **Business Objects Data Integrator**.

The **Define a new connection** window below shows the generic JDBC driver, which you can use to establish a new connection to a database.



SAP BO also

lets you customize the interface to include a SQream data source.

8.1.3.7.2 Establishing a New Connection Using a Generic JDCB Connector

This section shows an example of using a generic JDBC connector to establish a

new connection.

To establish a new connection using a generic JDBC connector:

1. In the fields, provide a user name, password, database URL, and JDBC class.

   The following is the correct format for the database URL:

```
↪<pre>
↪jdbc:Sqream:/
↪/
↪
↪<ipaddress>
↪:3108/
↪
↪<nameofdatabase>
↪
```

SQream recommends quickly testing your connection to SQream by selecting the Generic JDBC data source in the **Define a new connection** window. When you connect using a generic JDBC data source you do not need to modify your configuration files, but are limited to the out-of-the-box settings defined in the default **jdbc.prm** file.

Note: Modifying the jdbc.prm file for the generic driver impacts

all other databases using the same driver.

For more information, see Connection String Examples.

2. (Optonal)If you are using the generic JDBC driver specific to SQream, modify the jdbc.sbo file to include the SQream JDBC driver location by adding the following lines under the Database section of the file:

```
Database␣
```

(continues on next page)

(continued from previous page)

```
↪Active=
↪"Yes
↪"␣
↪Name=
↪"SQream␣
↪JDBC␣
↪data␣
↪source
↪">
↪

↪<JDBCDriver>
↪

↪<ClassPath>
↪

↪<Path>
↪C:\
↪Program␣
↪Files\
↪SQream␣
↪Technologies\
↪JDBC␣
↪Driver\
↪2021.
↪2.
↪0-
↪4.
↪5.
↪3\
↪sqream-
↪jdbc-
↪4.
↪5.
↪3.
↪jar
↪</
↪Path>
↪

↪</
↪ClassPath>
↪

↪</
↪Parameter>
↪

↪<Parameter␣
↪Name=
↪"JDBC␣
↪Class
↪">
↪
com.
↪sqream.
```

(continues on next page)

(continued from previous page)

```
↪jdbc.
↪SQDriver


↪</
↪JDBCDriver>
↪

↪</
↪DataBase>
↪
```

3. Restart the BusinessObjects server.

   When the connection is established, **SQream** is listed as a driver selection.

8.1.3.8 SAS Viya

SAS Viya is a cloud-enabled analytics

engine used for producing useful insights.

- *Installing SAS Viya*
- *Configuring SAS Viya*
- *Operating SAS Viya*
- *Troubleshooting SAS Viya*

8.1.3.8.1 Installing SAS Viya

The **Installing SAS Viya** section describes the following:

8.1.3.8.1.1 Downloading SAS Viya

Integrating with SQreamDB has been tested with SAS Viya v.03.05 and newer.

To download SAS Viya, see SAS Viya.

8.1.3.8.1.2 Installing the JDBC Driver

The SQreamDB JDBC driver is required for establishing a connection between SAS

Viya and SQreamDB.

To install the JDBC driver:

1. Download the *JDBC driver*.

2. Unzip the JDBC driver into a location on the SAS Viya server.

   SQreamDB recommends creating the directory `/opt/sqream` on the SAS Viya

server.

8.1.3.8.2 Configuring SAS Viya

After installing the JDBC driver, you must configure the JDBC driver from the SAS Studio so that it can be used with SQreamDB BStudio.

To configure the JDBC driver from the SAS Studio:

1. Sign in to the SAS

Studio.

2. From the **New** menu, click **SAS Program**.

3. Configure the SQreamDB JDBC connector by adding the following rows:

```
options␣
↪sastrace=
↪'d,
↪d,
↪d,
↪d
↪'␣
↪
sastraceloc=saslog␣
↪
nostsuffix␣
↪
msglevel=i␣
↪
sql_
↪ip_
↪trace=(note,
↪source)␣
↪
DEBUG=DBMS_
```

(continues on next page)

(continued from previous page)

```
↪SELECT;
↪

options␣
↪validvarname=any;
↪

libname␣
↪sqlib␣
↪jdbc␣
↪driver=
↪"com.
↪sqream.
↪jdbc.
↪SQDriver
↪"
␣
↪␣
↪␣
↪classpath=
↪"/
↪opt/
↪sqream/
↪sqream-
↪jdbc-
↪4.
↪0.
↪0.
↪jar
↪"␣
↪
␣
↪␣
↪␣
↪URL=
↪"jdbc:Sqream:/
↪/
↪sqream-
↪cluster.
↪piedpiper.
↪com:3108/
↪raviga;
↪cluster=true
↪"␣
↪
␣
↪␣
↪␣
↪user=
↪"rhendricks
↪"
␣
↪␣
↪␣
↪password=
↪"Tr0ub4dor3
↪"
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪␣
↪schema=
↪"public
↪"␣
↪
␣
↪␣
↪␣
↪PRESERVE_
↪TAB_
↪NAMES=YES
␣
↪␣
↪␣
↪PRESERVE_
↪COL_
↪NAMES=YES;
↪
```

8.1.3.8.3 Operating SAS Viya

The **Operating SAS Viya** section describes the following:

8.1.3.8.3.1 Using SAS Viya Visual Analytics

This section describes how to use SAS Viya Visual Analytics.

To use SAS Viya Visual Analytics:

1. Log in to SAS Viya Visual Analytics using your creden-

tials:

2. Click **New Report**.

3. Click **Data**.

4. Click **Data Sources**.

5. Click the **Connect** icon.

6. From the **Type** menu, select **Database**.

7. Provide the required informa-

tion and select **Persist this connection beyond the current session**.

8. Click **Advanced** and provide the required information.

9. Add the following additional parameters by clicking **Add**

Parameters:

Name	Value
class	com.sqream.jdbc.SQDriver
classPath	<path_to_jar_file>
url	\jdbc:Sqream://*<IP>*:*<port>*/*<database>*;cluster=true
username	<username>
password	<password>

10. Click **Test Connection**.

11. If the connection is successful, click **Save**.

8.1.3.8.4 Troubleshooting SAS Viya

The **Best Practices and Troubleshooting** section describes

the following best practices and troubleshooting procedures when connecting to SQreamDB using SAS Viya:

8.1.3.8.4.1 Inserting Only Required Data

When using SAS Viya, SQreamDB recommends using only data that you need, as described below:

- Insert only the data sources you need

into SAS Viya, excluding tables that don't require analysis.

- To increase query performance, add filters before analyzing. Every modification you make while analyzing data queries the SQreamDB database, sometimes several times. Adding filters to the datasource before exploring limits the amount of data analyzed and increases query performance.

8.1.3.8.4.2 Creating a Separate Service for SAS Viya

SQreamDB recommends creating a separate service for SAS Viya with the DWLM. This reduces the impact that Tableau has on other applications and processes, such as ETL. In addition, this works in conjunction with the load balancer to ensure good performance.

8.1.3.8.4.3 Locating the SQreamDB JDBC Driver

In some cases, SAS Viya cannot

locate the SQreamDB JDBC driver, generating the following error message:

```
java.
↪lang.
↪ClassNotFoundException:␣
↪com.
↪sqream.
↪jdbc.
↪SQDriver
```

To locate the SQreamDB JDBC driver:

1. Verify that you have placed the JDBC driver in a directory that SAS Viya can ac-

cess.

2. Verify that the classpath in your SAS program is correct, and that SAS Viya can access the file that it references.

3. Restart SAS Viya.

   For more troubleshooting assistance, see the SQreamDB Support Portal.

8.1.3.8.4.4 Supporting TEXT

In SAS Viya versions lower than 4.0, casting `TEXT` to `CHAR` changes the size to 1,024, such as when creating a table including a `TEXT` column. This is resolved by casting `TEXT` into `CHAR` when using the JDBC driver.

8.1.3.9 SQL Workbench

You can use SQL Workbench to interact with a SQreamDB cluster. SQL Work-

bench/J is a free SQL query tool, and is designed to run on any JRE-enabled environment.

This tutorial is a guide that will show you how to connect SQL Workbench to SQream DB.

In this topic:

- *Installing SQL Workbench with the SQream Installer*
- *Installing SQL Workbench Man-*

u-
ally

– *Install Java Runtime*

– *Get the SQreamDB JDBC Driver*

– *Install SQL Workbench*

– *Setting up the SQreamDB JDBC Driver Profile*

- *Create a New Connection Profile for Your Cluster*

- *Suggested Optional Configu-ra-*

tion

8.1.3.9.1 Installing SQL Workbench with the SQream Installer

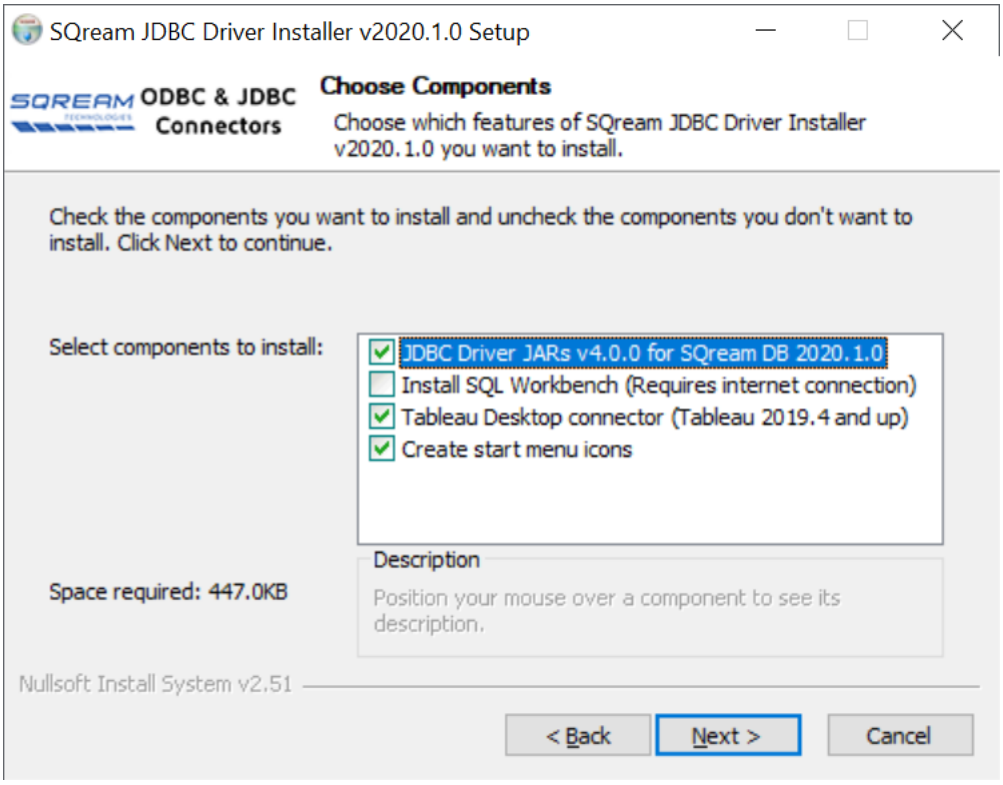
This section applies to Windows only.

SQream DB's driver installer for Windows can install the Java prerequisites and SQL Workbench for you.

1. Get the JDBC driver installer

available for download from the SQream Drivers page. The Windows installer takes care of the Java prerequisites and subsequent configuration.

2. Install the driver by following the on-screen instructions in the easy-to-follow installer. By default, the installer does not install SQL Workbench. Make sure to select the item!

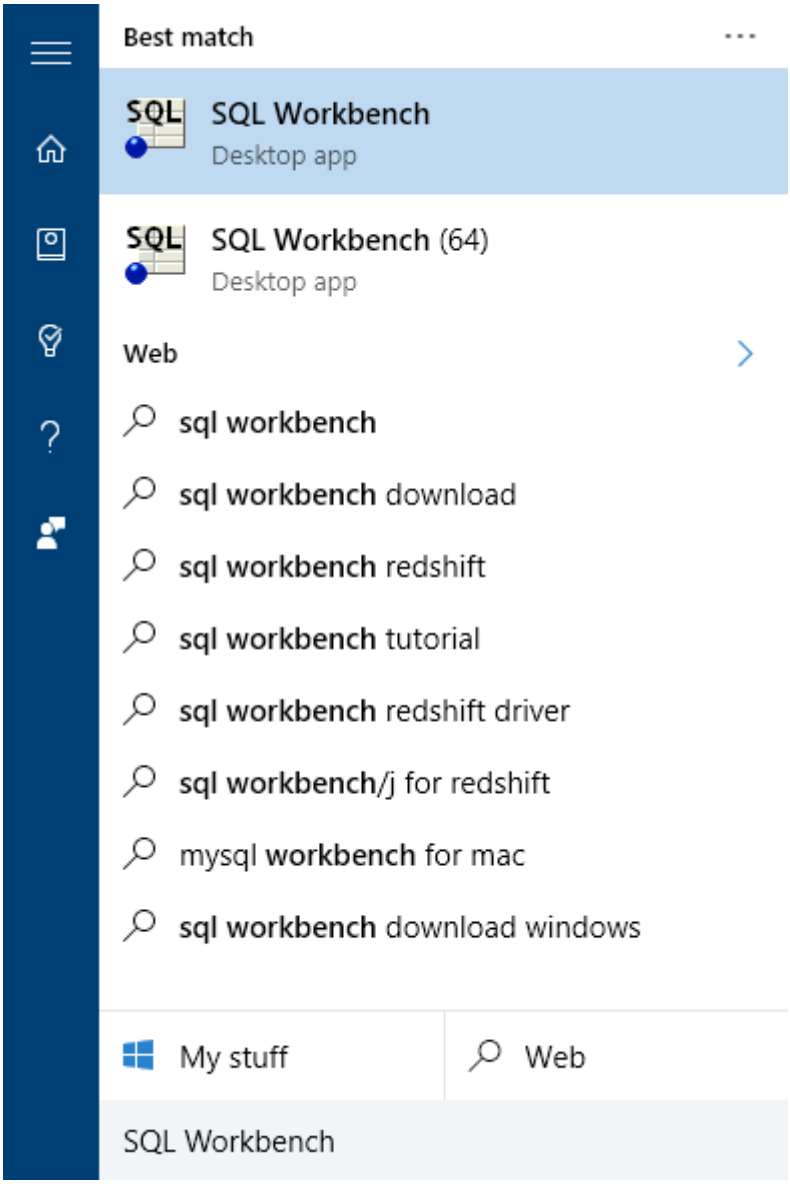


Note: The installer will install SQL Workbench in `C:\Program Files\SQream Technologies\SQLWorkbench` by default. You can change this path during the installation.

1. Once finished, SQL Workbench is installed and contains the necessary configuration for con-

nect-
ing
to SQream DB clusters.

2. Start SQL Work-bench from the Win-dows start menu. Be sure to se-lect **SQL Work-bench (64)** if you're on 64-bit Win-dows.



You are now ready to create a profile for your cluster. Continue

to *Creating a new connection profile*.

8.1.3.9.2 Installing SQL Workbench Manually

This section applies to Linux and MacOS only.

8.1.3.9.2.1 Install Java Runtime

Both SQL Workbench and the SQream DB JDBC driver require Java 1.8 or

newer. You can install either Oracle Java or OpenJDK.

Oracle Java

Download and install Java 8 from Oracle for your platform - https://www.java.com/en/download/manual.jsp

OpenJDK

For Linux and BSD, see https://openjdk.

java.net/install/

For Windows, SQream recommends Zulu 8 https://www.azul.com/downloads/zulu-community/?&version=java-8-lts&architecture=x86-64-bit&package=jdk

8.1.3.9.2.2 Get the SQream DB JDBC Driver

SQream DB's JDBC driver is provided as a zipped JAR file, available for download from

the SQream Drivers page.

Download and extract the JAR file from the zip archive.

8.1.3.9.2.3 Install SQL Workbench

1. Download the latest stable release from https://www.sql-workbench.eu/downloads.html. The **Generic package for all systems** is recommended.
2. Extract

the downloaded ZIP archive into a directory of your choice.

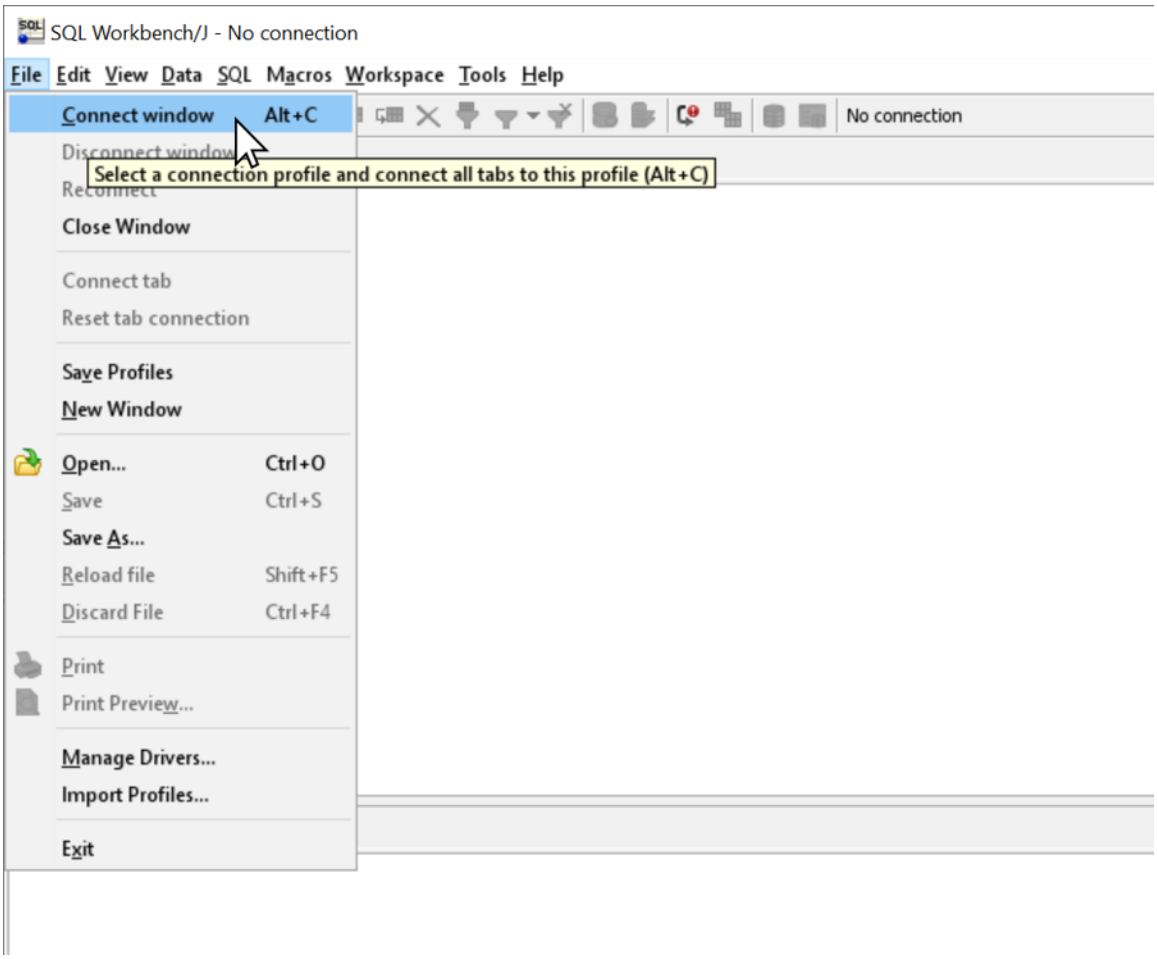
3. Start SQL workbench. If you are using 64 bit windows, run `SQLWorkbench64.exe` instead of `SQLWOrkbench.exe`.

8.1.3.9.2.4 Setting up the SQream DB JDBC Driver Profile

1. Define a connec-

tion pro-file -

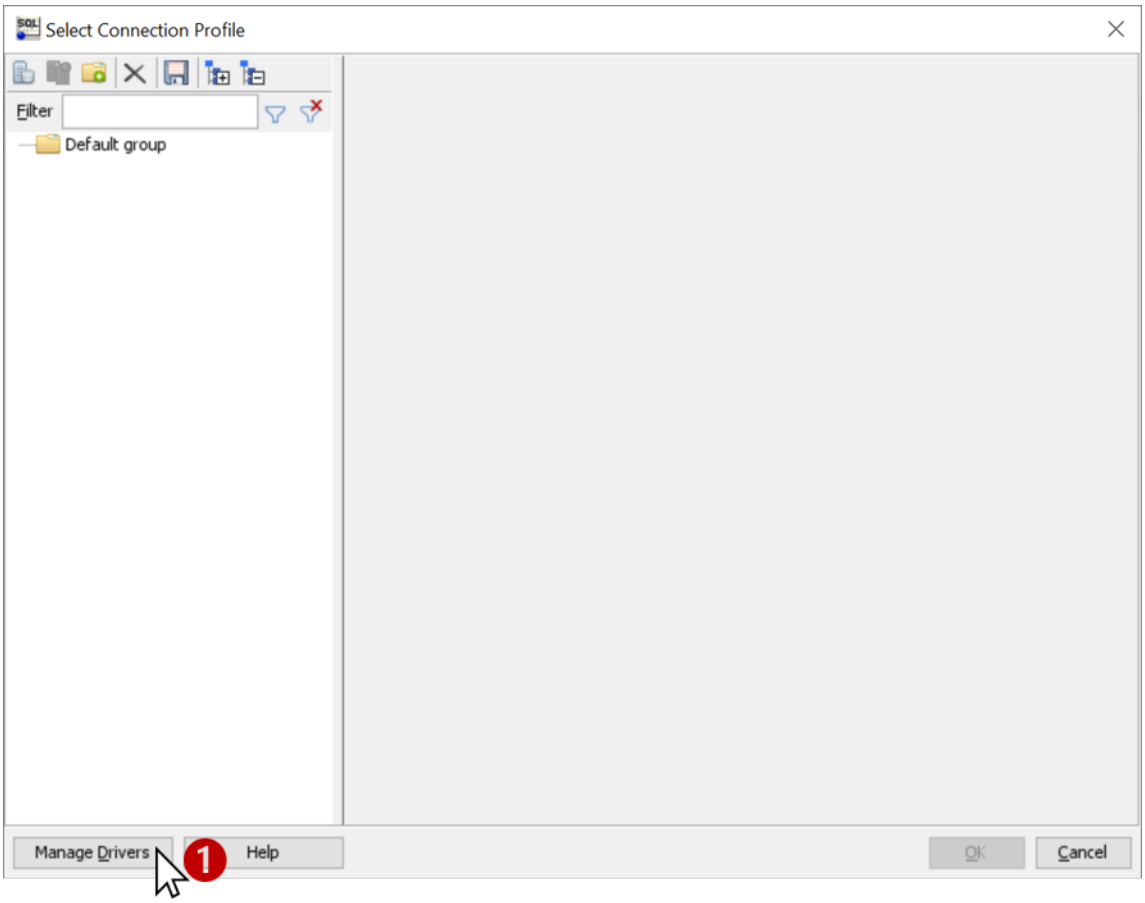
File ▸ *Connect window (Alt+C)*



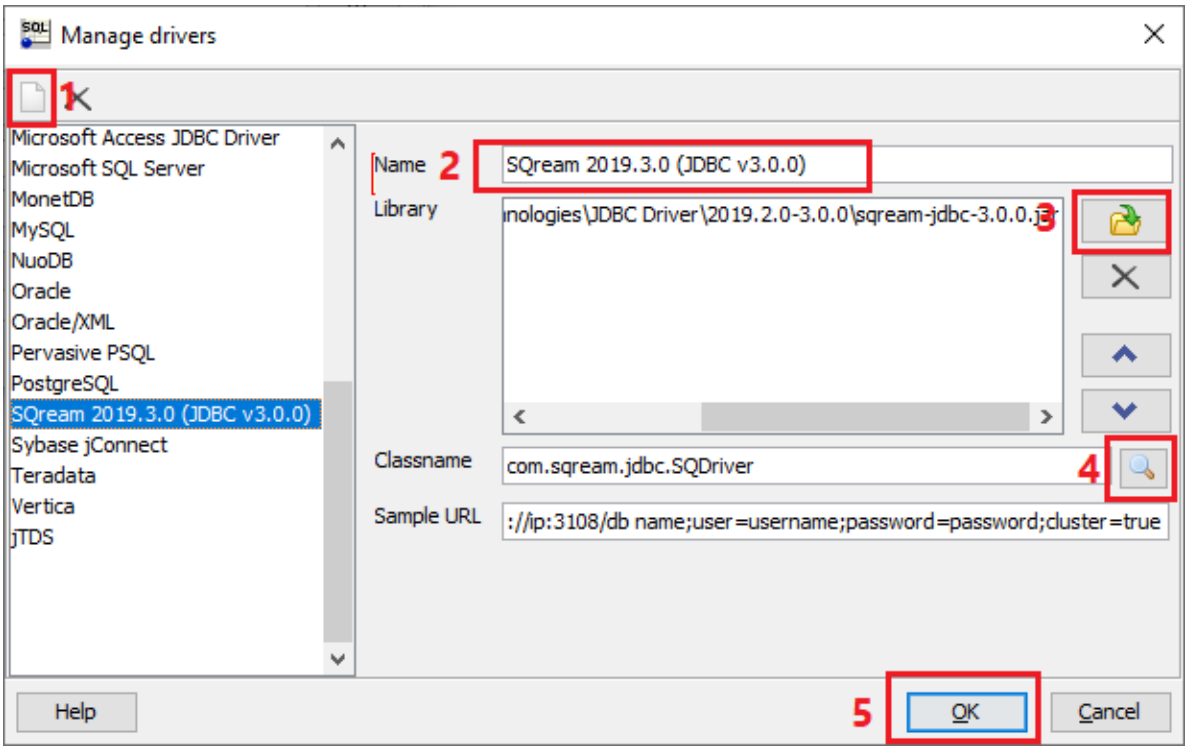
2. Open the drivers man-age-ment win-dow -

Manage

Drivers



3. Create the SQream DB driver profile



1. Click on the Add new driver button ("New" icon)

2. Name the driver as you see fit. We recommend calling it SQream DB <ver-

sion>, where <version> is the version you have installed.

3. Add the JDBC drivers from the location where you extracted the SQream DB JDBC JAR.

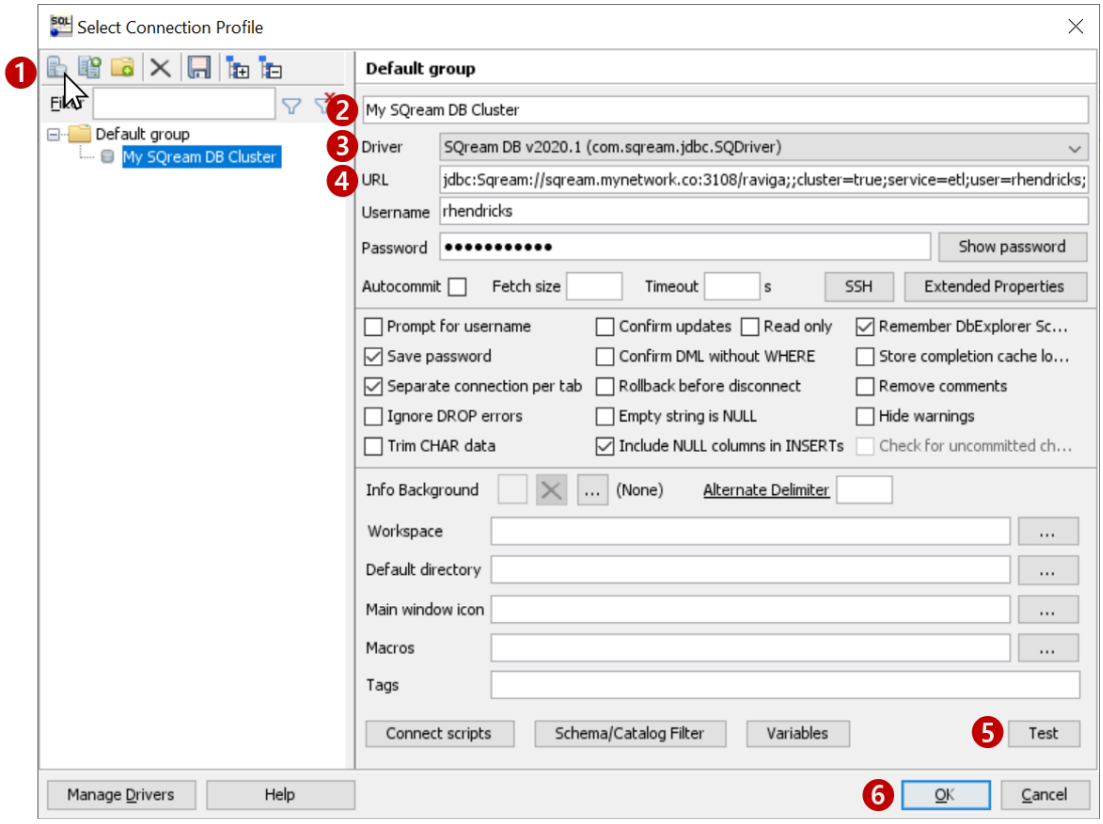
   If you used the SQream installer, the file will be in `C:\Program Files\SQream Technologies\JDBC Driver\`

4. Click the mag-

nifying glass button to detect the classname automatically. Other details are purely optional

5. Click OK to save and return to "new connection screen"

8.1.3.9.3 Create a New Connection Profile for Your Cluster



1. Create new connection by clicking the New icon

(top left)

2. Give your connection a descriptive name

3. Select the SQream Driver that was created in the previous screen

4. Type in your connection string. To find out more about your connection string (URL), see the Connec-

tion string documentation.

5. Text the connection details

6. Click OK to save the connection profile and connect to SQream DB

8.1.3.9.4 Suggested Optional Configuration

If you installed SQL Workbench manually, you can set

a customization to help SQL Workbench show information correctly in the DB Explorer panel.

1. Locate your workbench.settings file On Windows, typically: `C:\Users\<user name>\.sqlworkbench\workbench.settings` On Linux, `$HOME/.sqlworkbench`
2. Add the following line at the end of the file:

```
workbench.
↪db.
↪sqreamdb.
↪schema.
↪retrieve.
↪change.
↪catalog=true
```

3. Save the file and restart SQL Workbench

8.1.3.10 Tableau

SQream's Tableau connector, based on standard JDBC, enables storing and fast querying large volumes of data. This connector is useful for users who want to integrate and analyze data from various sources within the Tableau platform. With the Tableau connector, users can easily connect to databases and cloud applications and perform high-speed queries on large datasets. Additionally, the connector allows for seamless integration with Tableau, enabling users to visualize their data.

SQream supports both

Tableau Desktop and Tableau Server on Windows, MacOS, and Linux distributions.

For more information on SQream's integration with Tableau, see Tableau Connectors.

- *Prerequisites*
- *Setting Up JDBC*
- *Installing the Tableau Connector*

- *Connecting to SQream*

8.1.3.10.1 Prerequisites

It is essential that you have the following installed:

- Tableau version 9.2 or newer

8.1.3.10.2 Setting Up JDBC

1. Download the SQream JDBC Connector *.jar file*.
2. Place the JDBC `.jar`

file in the Tableau driver directory.

Based on your operating system, you may find the Tableau driver directory in one of the following locations:

- Tableau Desktop on MacOS: `~/Library/Tableau/Drivers`
- Tableau Desktop on Win-

dows: `C:\Program Files\Tableau\Drivers`

- Tableau on Linux: `/opt/tableau/tableau_driver/jdbc`

8.1.3.10.3 Installing the Tableau Connector

1. Download the *Tableau Connector* `SQreamDB.taco` file.
2. Based on the installation method that you used for installing Tableau, place the

Tableau Connector `SQreamDB.taco` file in the Tableau connector directory:

Product / Platform	Path
Tableau Desktop for Windows	`C:\Users[user]\Documents\My Tableau Repository\Connectors`
Tableau Desktop for Mac	`/Users/[user]/Documents/My Tableau Repository/Connectors`
Tableau Prep for Windows	`C:\Users[user]\Documents\My Tableau Prep Repository\Connectors`
Tableau Prep for Mac	`/Users/[user]/Documents/My Tableau Prep Repository/Connectors`
Flow web authoring on Tableau Server	`/data/tabsvc/flowqueryservice/Connectors`
Tableau Prep Conductor on Tableau Server	`/data/tabsvc/flowprocessor/Connectors`
Tableau Server	`C:\<directory_name>\Tableau\Tableau Server\<directory_name>\Connectors`

3. Restart Tableau Desktop or Tableau server.

8.1.3.10.4 Connecting to SQream

1. Start Tableau Desktop.

2. In the **Connect** menu,

under the **To a Server** option, click **More**.

Additional connection options are displayed.

3. Select **SQream DB by SQream Technologies**.

   A connection dialog box is displayed.

4. In the con-

nection dialog box, fill in the following fields:

Field name	Description	Example
Server	Defines the SQreamDB worker machine IP. Avoid using the loopback address (127.0.0.1) or "localhost" as a server address since it typically refers to the local machine where Tableau is installed and may create issues and limitations	`192.162.4.182` or `sqream.mynetwork.com`
Port	Defines the TCP port of the SQream worker	`3108` when using a load balancer, or `5100` when connecting directly to a worker with SSL
Database	Defines the database to establish a connection with	`master`
Cluster	Enables (`true`) or disables (`false`) the load balancer. After enabling or disabling the load balance, verify the connection	
Username	Specifies the username of a role to use when connecting	`rhendricks`
Password	Specifies the password of the selected role	`Tr0ub4dor&3`
Require SSL	Sets SSL as a requirement for establishing this connection	

5. Click **Sign In**.

   The connection is established, and the

data source page is displayed.

8.1.3.11 Talend

8.1.3.11.1 Overview

This page describes how to use Talend to interact with a SQream cluster. The Talend connector is used for reading data from a SQream cluster and loading data into SQream. In addition, this page provides a viability report on Talend's comparability with SQream for stakeholders.

The **Connecting to SQream Using Talend** describes the

following:

- *Creating a New Metadata JDBC DB Connection*
- *Supported SQream Drivers*
- *Supported Data Sources*
- *Known Issues*

8.1.3.11.1.1 Creating a New Metadata JDBC DB Connection

To create a new metadata JDBC DB con-

nec-
tion:

1. In the **Repos-i-tory** panel, na-gi-vate to **Meta-data** and right-click **Db con-nec-tions**.

2. Se-lect **Cre-ate con-nec-tion**.

3. In the **Name** field, type a name.

   Note that the name can-not

contain spaces.

4. In the **Purpose** field, type a purpose and click **Next**.

   Note that you cannot continue to the next step until you define both a Name and a Purpose.

5. In the **DB Type** field, select **JDBC**.

6. In the **JDBC URL** field, type the relevant connection string.

   For connection string examples, see Connection Strings.

7. In the **Drivers** field, click the **Add** button.

   The **"new-Line"** entry is added.

8. One the **"new-Line'**

en-
try,
click
the
el-
lip-
sis.

The
**Mod-
ule**
win-
dow
is
dis-
played.

9. From the Module window, select **Artifact repository(local m2/nexus)** and select **Install a new module**.

10. Click the ellipsis.

Your hard drive

is displayed.

11. Navigate to a **JDBC jar file** (such as **sqream-jdbc-4.5.3.jar**)and click **Open**.

12. Click **Detect the module install status**.

13. Click **OK**.

    The JDBC that you selected is displayed in the **Driver** field.

14. Click **Select class name**.

15. Click **Test connection**.

    If a driver class is not found (for example, you didn't select a JDBC jar file), the following error message is displayed:

    After creating a new metadata JDBC DB

connection, you can do the following:

- Use your new metadata connection.

- Drag it to the **job** screen.

- Build Talend components.

  For more information on loading data from JSON files to

the Talend Open Studio, see How to Load Data from JSON Files in Talend.

8.1.3.11.1.2 Supported SQream Drivers

The following list shows the supported SQream drivers and versions:

- **JDBC** - Version 4.3.3 and higher.

- **ODBC** - Version 4.0.0. This

version requires a Bridge to connect. For more information on the required Bridge, see Connecting Talend on Windows to an ODBC Database.

8.1.3.11.1.3 Supported Data Sources

Talend Cloud connectors let you create reusable connections with a wide variety of systems and environments, such as those shown below. This lets you access and read records of a range of diverse data.

•

Connections: Connections are environments or systems for storing datasets, including databases, file systems, distributed systems and platforms. Because these systems are reusable, you only need to establish connectivity with them once.

- **Datasets:** Datasets include database tables, file names, topics (Kafka), queues (JMS) and file paths (HDFS). For more informa-

tion on the complete list of connectors and datasets that Talend supports, see Introducing Talend Connectors.

8.1.3.11.1.4 Known Issues

As of 6/1/2021 schemas were not displayed for tables with identical names.

If you experience issues using Talend, see the SQream support portal.

8.1.3.12 TIBCO Spotfire

8.1.3.12.1 Overview

The **TIBCO Spotfire** software is an analytics solution that enables visualizing and exploring data through dashboards and advanced analytics.

This document is a Quick Start Guide that describes the following:

- *Establishing a Connection between TIBCO Spotfire and SQream*
- *Troubleshooting*

8.1.3.12.1.1 Establishing a Connection between TIBCO Spotfire and SQream

TIBCO Spotfire supports the following versions:

- **JDBC driver** - Version 4.5.2
- **ODBC**

driver -

Version 4.1.1

SQream supports TIBCO Spotfire version 7.12.0.

The **Establishing a JDBC Connection between TIBCO Spotfire and SQream** section describes the following:

- *Creating a JDBC Connection*
- *Creating*

an ODBC Connection

- *Creating the SQream Data Source Template*
- *Creating a Data Source*
- *Creating an Information Link*

8.1.3.12.1.2 Creating a JDBC Connection

For TIBCO Spotfire to recognize SQream, you must add the correct JDBC jar

file to Spotfire's loaded binary folder. The following is an example of a path to the Spotfire loaded binaries folder: `C:\tibco\tss\7.12.0\tomcat\bin`.

For the complete TIBCO Spotfire documentation, see TIBCO Spotfire® JDBC Data Access Connectivity Details.

8.1.3.12.1.3 Creating an ODBC Connection

To create an ODBC connection

1. Install and configure ODBC on Windows.

   For more information, see *Install and Configure ODBC on Windows*.

2. Launch the TIBCO Spotfire application.

3. From the **File** menu click **Add Data Tables**.

   The

Add Database Tables window is displayed.

4. Click **Add** and select **Database**.

   The **Open Database** window is displayed.

5. In the **Data source type** area, select **ODBC SQream** (Odbc Data Provider) and click **Configure**.

   The **Configure Data Source and Connec-**

tion window is displayed.

6. Select **System or user data source** and from the drop-down menu select the DSN of your data source (SQreamDB).

7. Provide your database username and password and click **OK**.

8. In the **Open**

Database window, click **OK**.

The **Specify Tables and Columns** window is displayed.

9. In the **Specify Tables and Columns** window, select the checkboxes corresponding to the tables and columns that you want to include in your SQL statement.

10. In the **Data source name**

field, set your data source name and click **OK**.

Your data source is displayed in the **Data tables** area.

11. In the **Add Data Tables** dialog, click **OK** to load the data from your ODBC data source into Spotfire.

Note: Verify that you have checked the SQL statement.

8.1.3.12.1.4 Creating the SQream Data Source Template

After creating a connection, you can create your SQream data source template.

To create your SQream data source template:

1. Log in to the TIBCO Spotfire Server Configuration Tool.

2. From the **Configuration** tab, in the **Configuration Start** menu, click **Data Source Templates**.

   The **Data Source Templates** list is displayed.

3. From the Data Source Templates list do one of the following:

- Override an

existing template:

1. In the template text field, select an existing template.

2. Copy and paste your data source template text.

- Create a new template:

1. Click **New**.

   The **Add Data Source Template**

window is displayed.

2. In the **Name** field, define your template name.

3. In the **Data Source Template** text field, copy and paste your data source template text.

   The following is an example of a data source template:

```
↪<jdbc-
↪type-
↪settings>
↪
␣
↪␣
↪
↪<type-
↪name>
↪SQream
↪</
↪type-
↪name>
↪
␣
↪␣
↪
↪<driver>
↪com.
↪sqream.
↪jdbc.
↪SQDriver
↪</
↪driver>
↪
␣
↪␣
↪
↪<connection-
↪url-
↪pattern>
↪jdbc:Sqream:/
↪/
↪&
↪lt;
↪host&
↪gt;
↪:&
↪lt;
↪port&
↪gt;
↪/
↪database;
↪user=sqream;
↪password=sqream;
↪cluster=true
↪</
↪connection-
↪url-
↪pattern>
↪
␣
↪␣
↪
↪<supports-
↪catalogs>
```

(continues on next page)

(continued from previous page)

```
↪true
↪</
↪supports-
↪catalogs>
↪
␣
↪␣
↪
↪<supports-
↪schemas>
↪true
↪</
↪supports-
↪schemas>
↪
␣
↪␣
↪
↪<supports-
↪procedures>
↪false
↪</
↪supports-
↪procedures>
↪
␣
↪␣
↪
↪<table-
↪types>
↪TABLE,
↪EXTERNAL_
↪TABLE
↪</
↪table-
↪types>
↪
␣
↪␣
↪
↪<java-
↪to-
↪sql-
↪type-
↪conversions>
↪
␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪
↪<from>
↪Bool
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪Integer
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪TEXT(2048)
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
```

(continues on next page)

(continued from previous page)

```
↪String
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪INT
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪Integer
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪BIGINT
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪LongInteger
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪Real
↪</
↪from>
```

(continues on next page)

(continued from previous page)

```
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪Real
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪Decimal
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪Float
↪</
```

(continues on next page)

(continued from previous page)

```
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪Numeric
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪Float
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪Date
↪</
↪from>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪DATE
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪
↪<type-
↪mapping>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<from>
↪DateTime
↪</
↪from>
↪
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪<to>
↪DateTime
↪</
↪to>
↪
␣
↪␣
↪␣
↪␣
↪
↪</
↪type-
↪mapping>
↪
␣
↪␣
↪␣
↪
↪</
↪java-
↪to-
↪sql-
↪type-
↪conversions>
↪
␣
↪␣
↪
↪<ping-
↪command>
↪
↪</
↪ping-
↪command>
↪

↪</
↪jdbc-
↪type-
↪settings>
↪
```

4. Click **Save configuration**.

5. Close and restart your Spotfire server.

8.1.3.12.1.5 Creating a Data Source

After creating the SQream data source template, you can create a data source.

To create a data source:

1. Launch the TIBCO Spotfire application.

2. From the **Tools** menu, select **Information Designer**.

   The **Information Designer** window is displayed.

3. From the **New** menu, click **Data Source**.

   The **Data Source** tab is displayed.

4. Provide the

following information:

- **Name** - define a unique name.

- **Type** - use the same type template name you used while configuring your template. See **Step 3** in *Creating the SQream Data Source Template*.

- **Connection**

URL
-

use the standard JDBC connection string, `<ip>:<port>/database`.

- **No. of connections**
-

define a number between **1** and **100**. SQream recommends setting your number of connections to **100**.

- **Username and Password**
-

define your SQream username and password.

8.1.3.12.1.6 Creating an Information Link

After creating a data source, you can create an information link.

To create an information link:

1. From the **Tools** menu, se-

lect **Information Designer**.

The **Information Designer** window is displayed.

2. From the **New** menu, click **Information Link**.

   The **Information link** tab is displayed.

3. From the **Elements** tab, select

a column type and click **Add**.

The column type is added to the **Elements** region as a filter.

Note the following:

- You can select procedures from the Elements region.

- You can remove

an element by selecting an element and clicking **Remove**.

Tip: If the Elements menu is not displayed, you can display it by clicking the **Elements** tab. You can simultaneously select multiple elements by pressing **Ctrl** and making additional selections, and select a range of elements by holding **Shift** and clicking two elements.

---

4. If the elements

you select originate from more than one data source table, specify a **Join path**.

5. *Optional* -

   In the **Description** region, type the description of the information link.

6. *Optional* -

   To fil-

ter your data, expand the **Filters** section and do the following:

1. From the **Information Link** region, select the element you added in Step 3 above.

2. Click **Add**.

   The **Add Column** window is displayed.

3. From the drop-down list, select a column to add a hard filter to and click **OK**.

   The selected column is added to the Filters list.

4. Repeat steps 2 and 3 to add filters to additional columns.

5.

For each column, from the **Filter Type** drop-down list, select **range** or **values**.

Note: Filtering by range means entering the upper and lower limits of the desired range. Filtering by values means entering the exact values that you want to include in the returned data, separated by semicolon.

---

6. In the **Values** field

type the desired values separated with semicolons, or set the upper and lower limits in the **Min Value** and **Max Value** fields. Alternatively, you can type `?param_name` in the Values field to use a parameter as the filter for the selected column, where `param_name` is the name used to identify the parameter.

Note: Because limits are inclusive, setting the lower limit to **1000** includes the value **1000** in the data table.

Note: When

setting upper and lower limits on **String** type columns, `A` precedes `AA`, and a lone letter precedes words beginning with that latter. For example, `S** precedes **Smith**, indicating that the name ``Smith` will not be present when you select names from `D` to `S`. The order of characters is standard ASCII.

For more information on adding filters, see Adding Hard Filters.

7. *Optional* -

   To add runtime filter-

ing prompts, expand the **Prompts** section and do the following:

1. Click **Add**.

   The **Add Column** window is displayed.

2. From the **Select column** list, select a column to add a prompt to and click **OK**.

   The selected column

is added to the Prompts list.

3. Repeat **Step 1** to add prompts to additional columns.

4. Do the following for each column:

- Make a selection from the **Prompt Type** drop-down list.
- Select or clear **Mandatory**.

- *Optional* -

  Set your **Max Selections**.

  For more information on adding prompts, see Adding Prompts.

8. *Optional* -

   Expand the **Conditioning** section and specify one of the following conditions:

- None

-

Distinct

- Pivot

  Note that you can edit the Pivot conditioning by selecting **Pivot** and clicking **Edit**.

9. *Optional* - Expand the **Parameters** section and define your parameters.

10. *Optional*

-

Expand the **Properties** section and define your properties.

11. *Optional* -

    Expand the **Caching** section and enable or disable whether your information link can be cached.

12. Click **Save**.

The **Save As** window is displayed.

13. In the tree, select where you want to save the information link.

14. In the **Name** field, type a name and description for the information link.

15. Click **Save**.

The new information link is added to the library and can be accessed by other users.

Tip: You can test the information link directly by clicking **Open Data**. You can also view and edit the SQL belonging to the information link by clicking **SQL**.

For more information on the Information Link attributes, see Information Link Tab.

8.1.3.12.1.7 Troubleshooting

The **Troubleshooting** section describes the following scenarios:

- *The JDBC Driver does not Support Boolean, Decimal, or Numeric Types*
- *Information Services do not Support Live Queries*

8.1.3.12.1.8 The JDBC Driver does not Support Boolean, Decimal, or Numeric Types

When attempting to load data, the the Boolean, Decimal, or Numeric column types are not supported and generate the following error:

```
Failed␣
↪to␣
↪execute␣
↪query:␣
↪Unsupported␣
↪JDBC␣
↪data␣
↪type␣
↪in␣
↪query␣
```

(continues on next page)

(continued from previous page)

```
↪result:␣
↪Bool␣
↪(HRESULT:␣
↪80131500)
```

The error above is resolved by casting the columns as follows:

- `Bool` columns to `INT`.
- `Decimal` and `Numeric` columns to `REAL`.

For more information, see the following:

- **Resolving this**

error - Details on Change Data Types.

- **Supported data types** - *Data Types*.

8.1.3.12.1.9 Information Services do not Support Live Queries

TIBCO Spotfire data connectors support live queries, but no APIs currently exist for creating

custom data connectors. This is resolved by creating a customized SQream adapter using TIBCO's **Data Virtualization (TDV)** or the **Spotfire Advanced Services (ADS)**. These can be used from the built-in TDV connector to enable live queries.

This resolution applies to JDBC and ODBC drivers.

8.2 Client Drivers

The guides on this page describe how to use the SqreamDB client drivers and client applications.

8.2.1 Client Driver Downloads

Driver	Download	Docs	Notes	Operating System
Apache Spark	Apache Spark Connector	spark		All
JDBC	sqream-jdbc 5.2.0	*JDBC*	Recommended installation via `mvn`	All
Node.JS	sqream-v4.2.4	*Node.JS*	Recommended installation via `npm`	All
ODBC	For the **ODBC installer**, please contact your SQreamDB representative	*Windows*, *Linux*		Windows, Linux
Power BI	Power BI Power Query Connector	*BI Desktop*		All
Python	pysqream v5.0.0	*Python (pysqream)*	Recommended installation via `pip`	All
SQream-Net	.NET .dll file	*SQream-NET*		All
Tableau	Tableau Connector	*Tableau*		All
Trino	Trino Connector	trino		All

8.2.1.1 JDBC

The SQream JDBC driver lets you connect to SQream using many Java applications and tools. This page describes how to

write a Java application using the JDBC interface. The JDBC driver requires Java 1.8 or newer.

- *Installing the JDBC Driver*
- *Connecting to SQream Using a JDBC Application*
- *Prepared Statements*

8.2.1.1.1 Installing the JDBC Driver

The **Installing the JDBC Driver** section describes the following:

- *Prerequisites*
-

Getting the JAR file

Setting Up the Class Path

8.2.1.1.1.1 Prerequisites

The SQream JDBC driver requires Java 1.8 or newer, and SQream recommends using Oracle Java or OpenJDK.:

- **Oracle Java** - Download and install Java 8

from Oracle for your platform.

- **OpenJDK** - Install OpenJDK
- **Windows** - SQream recommends installing Zulu 8

8.2.1.1.1.2 Getting the JAR file

The SQream JDBC driver is available for download from the *client drivers download*

page.
This JAR file can be integrated into your Java-based applications or projects.

8.2.1.1.1.3 Setting Up the Class Path

To use the driver, you must include the JAR named `sqream-jdbc-<version>.jar` in the class path, either by inserting it in the `CLASSPATH` environment variable, or by using flags on the relevant Java command line.

For example, if the JDBC

driver has been unzipped to `/home/sqream/sqream-jdbc-5.2.0.jar`, the following command is used to run application:

```
↪$␣
↪export␣
↪CLASSPATH=/
↪home/
↪sqream/
↪sqream-
↪jdbc-
↪5.
↪2.
↪0.
↪jar:
↪$CLASSPATH

↪$␣
↪java␣
↪my_
↪java_
↪app
```

Alternatively, you can pass `-classpath` to the Java executable file:

```
↪$␣
↪java␣
↪-
↪classpath␣
↪.
↪:/
↪home/
↪sqream/
↪sqream-
↪jdbc-
↪5.
↪2.
↪0.
↪jar␣
↪my_
↪java_
↪app
```

8.2.1.1.2 Connecting to SQream Using a JDBC Application

You can connect to SQream using one of the following JDBC applications:

- *Driver Class*
- *Connection String*
- *Java Program Sample*

8.2.1.1.2.1 Driver Class

Use `com.sqream.jdbc.SQDriver` as the driver class in the JDBC application.

8.2.1.1.2.2 Connection String

JDBC drivers rely on a connection string.

The following

is the syntax for SQream:

```
jdbc:Sqream:/
↪/
↪
↪<host␣
↪and␣
↪port>
↪/
↪
↪<database␣
↪name>
↪;
↪user=
↪<username>
↪;
↪password=
↪<password>
↪;
↪[
↪<optional␣
↪parameters>
↪;
↪␣
↪.
↪.
↪.
↪]
```

8.2.1.1.2.3 Connection Parameters

The following table shows the connection string parame-

ters:

Item	State	Default	Description
`<host and port>`	Mandatory	None	Hostname and port of the SQream DB worker. For example, `127.0.0.1:5000`, `sqream.mynetwork.co:3108`
`<database name>`	Mandatory	None	Database name to connect to. For example, `master`
`username=<username>`	Optional	None	Username of a role to use for connection. For example, `username=SqreamRole`
`password=<password>`	Optional	None	Specifies the password of the selected role. For example, `password=SqreamRolePassword2023`
`service=<service>`	Optional	`sqream`	Specifices service queue to use. For example, `service=etl`
`<ssl>`	Optional	`false`	Specifies SSL for this connection. For example, `ssl=true`
`<cluster>`	Optional	`true`	Connect via load balancer (use only if exists, and check port).
`<fetchSize>`	Optional	`true`	Enables on-demand loading, and defines double buffer size for the result. The `fetchSize` parameter is rounded according to chunk size. For example, `fetchSize=1` loads one row and is rounded to one chunk. If the `fetchSize` is 100,600, a chunk size of 100,000 loads, and is rounded to, two chunks.
`<insertBuffer>`	Optional	`true`	Defines the bytes size for inserting a buffer before flushing data to the server. Clients running a parameterized insert (network insert) can define the amount of data to collect before flushing the buffer.
`<loggerLevel>`	Optional	`true`	Defines the logger level as either `debug` or `trace`.
`<logFile>`	Optional	`true`	Enables the file appender and defines the file name. The file name can be set as either the file name or the file path.
`<idleconnectiontimeout>`	Optional	0	Sets the duration, in seconds, for which a database connection can remain idle before it is terminated. If the parameter is set to its default value, idle connections will not be terminated. The idle connection timer begins counting after the completion of query execution.

8.2.1.1.2.4 Connection String Examples

The following is an example of a SQream

cluster with a load balancer and no service queues (with SSL):

```
jdbc:Sqream:/
↪/
↪sqream.
↪mynetwork.
↪co:3108/
↪master;
↪user=rhendricks;
↪password=Tr0ub4dor&
↪3;
↪ssl=true;
↪cluster=true
```

The following is a minimal example of a local standalone SQream database:

```
jdbc:Sqream:/
↪/
↪127.
↪0.
↪0.
↪1:5000/
↪master;
```

(continues on next page)

(continued from previous page)

```
↪user=rhendricks;
↪password=Tr0ub4dor&
↪3
```

The following is an example of a SQream cluster with a load balancer and a specific service queue named `etl`, to the database named `raviga`

```
jdbc:Sqream:/
↪/
↪sqream.
↪mynetwork.
↪co:3108/
↪raviga;
↪user=rhendricks;
↪password=Tr0ub4dor&
↪3;
↪cluster=true;
↪service=etl
```

8.2.1.1.2.5 Java Program Sample

You can download the JDBC Application Sample File below by right-clicking and saving it to your computer.

Listing 1: JDBC Application Sample

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.Statement;
```

(continues on next page)

(continued from previous page)

```
import␣
↪java.sql.ResultSet;
↪␣
↪␣
↪

import␣
↪java.io.IOException;
↪␣
↪␣
↪
import␣
↪java.security.KeyManagementException;
↪␣
↪␣
↪
import␣
↪java.security.NoSuchAlgorithmException;
↪␣
↪␣
↪
import␣
↪java.sql.SQLException;
↪␣
↪␣
↪



public␣
↪␣
↪class␣
↪␣
↪SampleTest␣
↪␣
↪
↪{␣
↪␣
↪

␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Replace␣
↪with␣
↪your␣
↪connection␣
↪string
␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪static␣
↪␣
↪final␣
↪␣
↪String␣
↪␣
↪url␣
↪␣
↪=␣
↪␣
↪
↪"jdbc:Sqream:/
↪/
↪sqream.
↪mynetwork.
↪co:3108/
↪master;
↪user=rhendricks;
↪password=Tr0ub4dor&
↪3;
↪ssl=true;
↪cluster=true
↪";

␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Allocate␣
↪objects␣
↪for␣
↪result␣
↪set␣
↪and␣
↪metadata
␣
↪␣
↪␣
↪␣
↪Connection␣
↪␣
↪conn␣
↪␣
↪␣
↪␣
↪=␣
↪␣
↪null;
↪␣
↪␣
↪
␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪Statement␣
↪␣
↪stmt␣
↪␣
↪=␣
↪␣
↪null;
↪␣
↪␣
↪
␣
↪␣
↪␣
↪␣
↪ResultSet␣
↪␣
↪rs␣
↪␣
↪=␣
↪␣
↪null;
↪␣
↪␣
↪
␣
↪␣
↪␣
↪␣
↪DatabaseMetaData␣
↪␣
↪dbmeta␣
↪␣
↪=␣
↪␣
↪null;
↪␣
↪␣
↪

␣
↪␣
↪␣
↪␣
↪int␣
↪␣
↪res␣
↪␣
↪=␣
↪␣
↪0;
↪␣
↪␣
↪

␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪public␣
↪␣
↪void␣
↪␣
↪testJDBC()␣
↪␣
↪throws␣
↪␣
↪SQLException,
↪␣
↪␣
↪IOException␣
↪␣
↪
↪{␣
↪␣
↪
28
29 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Create␣
↪a␣
↪connection
30 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪conn␣
↪␣
↪=␣
↪␣
↪DriverManager.
↪getConnection(url,
↪
↪"rhendricks
↪",
↪
↪"Tr0ub4dor&
↪3
↪");
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪
31
32 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Create␣
↪a␣
↪table␣
↪with␣
↪a␣
↪single␣
↪integer␣
↪column
33 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪String␣
↪sql␣
↪␣
↪=␣
↪␣
↪
↪"CREATE␣
↪TABLE␣
↪test␣
↪(x␣
↪INT)
↪";
↪
34 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt␣
↪=␣
↪conn.
↪createStatement();
↪␣
↪␣
↪/
```

(continues on next page)

(continued from previous page)

```
↪/
↪␣
↪Prepare␣
↪the␣
↪statement
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt.
↪execute(sql);
↪␣
↪/
↪/
↪␣
↪Execute␣
↪the␣
↪statement
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt.
↪close();
↪␣
↪/
↪/
↪␣
↪Close␣
↪the␣
↪statement␣
↪handle

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Insert␣
↪some␣
↪values␣
↪into␣
↪the␣
```

(continues on next page)

(continued from previous page)

```
↪newly␣
↪created␣
↪table
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪sql␣
↪=␣
↪
↪"INSERT␣
↪INTO␣
↪test␣
↪VALUES␣
↪(5),
↪(6)
↪";
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt␣
↪=␣
↪conn.
↪createStatement();
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt.
↪execute(sql);
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt.
↪close();
↪
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Get␣
↪values␣
↪from␣
↪the␣
↪table
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪sql␣
↪=␣
↪
↪"SELECT␣
↪*␣
↪FROM␣
↪test
↪";
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt␣
↪=␣
↪conn.
↪createStatement();
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪rs␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪=␣
↪stmt.
↪executeQuery(sql);
↪
48 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Fetch␣
↪all␣
↪results␣
↪one-
↪by-
↪one
49 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪while(rs.
↪next())␣
↪
↪{
50 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪res␣
↪=␣
↪rs.
↪getInt(1);
↪
51 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪System.
↪out.
↪println(res);
↪␣
↪/
↪/
↪␣
↪Print␣
↪results␣
↪to␣
↪screen
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪}
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪rs.
↪close();
↪␣
↪/
↪/
↪␣
↪Close␣
↪the␣
↪result␣
↪set
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪stmt.
↪close();
↪␣
↪/
↪/
```

(continues on next page)

(continued from previous page)

```
↪␣
↪Close␣
↪the␣
↪statement␣
↪handle
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪conn.
↪close();
↪
␣
↪␣
↪␣
↪␣
↪}
↪


␣
↪␣
↪␣
↪␣
↪public␣
↪␣
↪static␣
↪␣
↪void␣
↪␣
↪main(String[]␣
↪␣
↪args)␣
↪␣
↪throws␣
↪␣
↪SQLException,
↪␣
↪␣
↪KeyManagementException,
↪␣
↪␣
↪NoSuchAlgorithmException,
↪␣
↪␣
↪IOException,
↪  ClassNotFoundException{

␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Load␣
↪SQream␣
↪DB␣
↪JDBC␣
↪driver
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪Class.
↪forName(
↪"com.
↪sqream.
↪jdbc.
↪SQDriver
↪");
↪␣
↪␣
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪Create␣
↪test␣
↪object␣
↪and␣
↪run
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪SampleTest␣
↪␣
↪test␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪=␣
↪␣
↪new␣
↪␣
↪SampleTest();
↪␣
↪␣
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪test.
↪testJDBC();
↪␣
↪␣
↪
␣
↪␣
↪␣
↪␣
↪}
↪␣
↪␣
↪
}
↪
```

8.2.1.1.3 Prepared Statements

Prepared statements, also known as parameterized queries, are a feature

of JDBC that enable the use of parameters to optimize query execution, enhance security, and enable query template reuse with different parameter values in Java applications.

8.2.1.1.3.1 Prepared Statement Sample

The following is a Java code snippet employing a JDBC prepared statement object to ingest a batch of one million records into SQreamDB.

You may download the `Prepared`

statement by right-clicking and saving it to your computer.

8.2.1.2 Python (pysqream)

The current Pysqream connector supports Python version 3.9 and newer. It includes a set of packages that allows Python programs to connect to SQream DB. The base `pysqream` package conforms to Python DB-API specifications PEP-249.

`pysqream` is a pure Python connector

that can be installed with `pip` on any operating system, including Linux, Windows, and macOS. `pysqream-sqlalchemy` is a SQLAlchemy dialect for `pysqream`.

- *Installing the Python Connector*
- *SQLAlchemy*
- *API*

8.2.1.2.1 Installing the Python Connector

8.2.1.2.1.1 Prerequisites

It is essential that you have

the following installed:

- *Python*
- *PIP*
- *OpenSSL for Linux*

8.2.1.2.1.2 Python

The connector requires Python version 3.9 or newer.

To see your current Python version, run the following command:

```
↪$␣
↪python␣
```

(continues on next page)

(continued from previous page)

```
↪-
↪-
↪version
```

8.2.1.2.1.3 PIP

The Python connector is installed via `pip`, the standard package manager for Python, which is used to install, upgrade and manage Python packages (libraries) and their dependencies.

We recommend upgrading to the latest version of `pip` before in-

stalling.

To verify that you have the latest version, run the following command:

```
↪$␣
↪python3␣
↪-
↪m␣
↪pip␣
↪install␣
↪-
↪-
↪upgrade␣
↪pip
Collecting␣
↪pip
␣
↪␣
↪␣
↪Downloading␣
↪https:/
↪/
↪files.
↪pythonhosted.
↪org/
↪packages/
↪00/
↪b6/
↪9cfa56b4081ad13874b0c6f96af8ce16cfbc1cb06bedf8e9164ce5551ec1/
↪pip-
↪19.
↪3.
↪1-
↪py2.
↪py3-
↪none-
↪any.
↪whl␣
↪(1.
```

(continues on next page)

(continued from previous page)

```
↪4MB)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪|▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉|␣
↪1.
↪4MB␣
↪1.
↪6MB/
↪s
Installing␣
↪collected␣
↪packages:␣
↪pip
␣
↪␣
↪Found␣
↪existing␣
↪installation:␣
↪pip␣
↪19.
↪1.
↪1
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪Uninstalling␣
↪pip-
↪19.
↪1.
↪1:
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪Successfully␣
↪uninstalled␣
↪pip-
↪19.
↪1.
```

(continues on next page)

(continued from previous page)

```
↪1
Successfully␣
↪installed␣
↪pip-
↪19.
↪3.
↪1
```

Note:

- On macOS, you may want to use virtualenv to install Python and the connector, to ensure compatibility with the built-in Python environment
- If you encounter an error including `SSLError` or `WARN-`

ING:
pip
is
con-
fig-
ured
with
lo-
ca-
tions
that
require TLS/SSL, however the ssl module in Python is not available. - please be sure to reinstall Python with SSL enabled, or use virtualenv or Anaconda.

8.2.1.2.1.4 OpenSSL for Linux

The Python connector relies on OpenSSL for secure connections to SQream DB. Some distributions of Python do not include OpenSSL.

To install OpenSSL on RHEL/CentOS,

run the following command:

```
↪$␣
↪sudo␣
↪yum␣
↪install␣
↪-
↪y␣
↪libffi-
↪devel␣
↪openssl-
↪devel
```

To install OpenSSL on Ubuntu, run the following command:

```
↪$␣
↪sudo␣
↪apt-
↪get␣
↪install␣
↪libssl-
↪dev␣
↪libffi-
↪dev␣
↪-
↪y
```

8.2.1.2.1.5 Installing via PIP with an internet connection

The Python connector is available via PyPi.

To install the connector using pip, it is advisable to use the `-U` or `--user` flags instead of sudo, as it ensures packages are installed per user. However, it is worth noting that the connector can only be accessed under the same user.

To install

`pysqream` and `pysqream-sqlalchemy` with the `--user` flag, run the following command:

```
↪$␣
↪pip3.
↪9␣
↪install␣
↪pysqream␣
↪pysqream-
↪sqlalchemy␣
↪-
↪-
↪user
```

`pip3` will automatically install all necessary libraries and modules.

8.2.1.2.1.6 Installing via PIP without an internet connection

1. To get the `.whl` package file, contact you SQream support representative.
2. Run the following command:

```
tar␣
↪-
↪xf␣
↪pysqream_
↪connector_
↪5.
↪0.
↪0.
↪tar.
↪gz
cd␣
```

(continues on next page)

(continued from previous page)

```
↪pysqream_
↪connector_
↪5.
↪0.
↪0

↪#Install␣
↪all␣
↪packages␣
↪with␣
↪-
↪-
↪no-
↪index␣
↪-
↪-
↪find-
↪links␣
↪.
↪
python3␣
↪-
↪m␣
↪pip␣
↪install␣
↪*.
↪whl␣
↪-
↪U␣
↪-
↪-
↪no-
↪index␣
↪-
↪-
↪find-
↪links␣
↪.
↪
python3.
↪9␣
↪-
↪m␣
↪pip␣
↪install␣
↪pysqream-
↪5.
↪0.
↪0.
↪zip␣
↪-
↪U␣
↪-
↪-
↪no-
↪index␣
↪-
```

(continues on next page)

(continued from previous page)

```
↪-
↪find-
↪links␣
↪.
↪
python3.
↪9␣
↪-
↪m␣
↪pip␣
↪install␣
↪pysqream-
↪sqlalchemy-
↪1.
↪0.
↪zip␣
↪␣
↪-
↪U␣
↪-
↪-
↪no-
↪index␣
↪-
↪-
↪find-
↪links␣
↪.
↪
```

8.2.1.2.1.7 Upgrading an Existing Installation

The Python drivers are updated periodically. To upgrade

an existing pysqream installation, use pip's `-U` flag:

```
↪$␣
↪pip3.
↪9␣
↪install␣
↪pysqream␣
↪pysqream-
↪sqlalchemy␣
↪-
↪U
```

8.2.1.2.2 SQLAlchemy

SQLAlchemy is an Object-Relational Mapper (ORM) for Python. When you install the SQream dialect (`pysqream-sqlalchemy`) you can use frameworks such as

Pandas, TensorFlow, and Alembic to query SQream directly.

- *Limitations*
- *Creating a Standard Connection*
- *Pulling a Table into Pandas*

8.2.1.2.2.1 Limitations

- Supports Pysqream 3.2.5
- Does not support the `ARRAY` data type

8.2.1.2.2.2 Creating a Standard Connection

Parameter	Description
`username`	Username of a role to use for connection
`password`	Specifies the password of the selected role
`host`	Specifies the hostname
`port`	Specifies the port number
`port_ssl`	An optional parameter
`database`	Specifies the database name
`clustered`	Establishing a multi-clustered connection. Input values: `True`, `False`. Default is `False`
`service`	Specifies service queue to use

```
import␣
↪sqlalchemy␣
↪as␣
↪sa
from␣
↪sqlalchemy.engine.url␣
↪import␣
↪URL


engine_
↪url␣
↪=␣
↪sa.
↪engine.
↪url.
↪URL(
↪'sqream
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      username='<user_name>',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      password='<password>',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      host='<host_name>',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      port=<port_number>,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪     port_ssl=<port_ssl>,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪     database='<database_name>')

engine␣
↪=␣
↪sa.
↪create_
↪engine(engine_
↪url,
↪connect_
↪args=
↪{
↪"clustered
↪":␣
↪False,
↪␣
↪
↪"service
↪":␣
↪
↪"
↪<service_
↪name>
↪
↪"}
↪)
session␣
```

(continues on next page)

(continued from previous page)

```
↪=␣
↪orm.
↪sessionmaker(bind=engine)()
```

8.2.1.2.2.3 Pulling a Table into Pandas

The following example shows how to pull a table in Pandas. This example uses the URL method to create the connection string:

```
import␣
↪sqlalchemy␣
↪as␣
↪sa
import␣
↪pandas␣
↪as␣
↪pd
from␣
↪sqlalchemy.engine.url␣
↪import␣
↪URL
```

(continues on next page)

(continued from previous page)

```
engine_
↪url␣
↪=␣
↪sa.
↪engine.
↪url.
↪URL(
↪'sqream
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪     username='sqream',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      password='12345',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪      host='127.0.0.1',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
```

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪     port=3108,
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪     database='master')

engine␣
↪=␣
↪sa.
↪create_
↪engine(engine_
↪url,
↪connect_
↪args=
↪{
↪"clustered
↪":␣
↪True,
↪␣
↪
↪"service
↪":␣
↪
↪"admin
↪"}
↪)
table_
↪df␣
↪=␣
```

(continues on next page)

(continued from previous page)

```
↪pd.
↪read_
↪sql(
↪"select␣
↪*␣
↪from␣
↪nba
↪",
↪␣
↪con=engine)
```

8.2.1.2.3 API

- *Using the Cursor*
- *Reading Result Metadata*
- *Loading Data into a Table*
- *Using SQLAlchemy ORM to Create and Populate Tables*

8.2.1.2.3.1 Using the Cursor

The DB-API specification includes several methods for fetching results from the cursor. This sections shows an example using the `nba` table, which looks as follows:

Table 1: nba

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	
R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0		5000000.0
Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0		12000000.0
Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0

As before, you must import the library and create a `Connection()`, followed by `execute()` on a simple `SELECT *` query:

```
import␣
↪pysqream
```

(continues on next page)

(continued from previous page)

```
con␣
↪=␣
↪pysqream.
↪connect(host=
↪'127.
↪0.
↪0.
↪1
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪port=3108,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪database=
↪'master
↪',
↪
␣
↪␣
↪␣
↪
```

(continued from previous page)

```
↪password=
↪'Tr0ub4dor&
↪3
↪',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪clustered=True)

cur␣
↪=␣
↪con.
↪cursor()␣
↪
↪#␣
↪Create␣
↪a␣
↪new␣
↪cursor

↪#␣
↪The␣
↪select␣
↪statement:
statement␣
↪=␣
↪
↪'SELECT␣
↪*␣
↪FROM␣
↪nba
↪'
cur.
↪execute(statement)
```

When
the

statement has finished executing, you have a `Connection` cursor object waiting. A cursor is iterable, meaning that it advances the cursor to the next row when fetched.

You can use `fetchone()` to fetch one record at a time:

```
first_row = cur.fetchone()

# Fetch one row at a time (first row)

second_
```

(continues on next page)

(continued from previous page)

```
↪row␣
↪=␣
↪cur.
↪fetchone()␣
↪
↪#␣
↪Fetch␣
↪one␣
↪row␣
↪at␣
↪a␣
↪time␣
↪(second␣
↪row)
```

To fetch several rows at a time, use `fetchmany()`:

```
↪#␣
↪executing␣
↪`fetchone`␣
↪twice␣
↪is␣
↪equivalent␣
↪to␣
↪this␣
↪form:
third_
↪and_
↪fourth_
↪rows␣
↪=␣
↪cur.
↪fetchmany(2)
```

To fetch all rows at once, use `fetchall()`:

```
```

(continues on next page)

(continued from previous page)

```
↪#␣
↪To␣
↪get␣
↪all␣
↪rows␣
↪at␣
↪once,
↪␣
↪use␣
↪`fetchall`
remaining_
↪rows␣
↪=␣
↪cur.
↪fetchall()

cur.
↪close()



↪#␣
↪Close␣
↪the␣
↪connection␣
↪when␣
↪done
con.
↪close()
```

The following is an example of the contents of the row variables used in our examples:

```
>
↪>
↪>
↪␣
↪print(first_
↪row)
(
↪'Avery␣
↪Bradley
↪',
↪␣
↪
↪'Boston␣
↪Celtics
↪',
↪␣
↪0,
↪␣
↪
↪'PG
↪',
↪␣
↪25,
↪␣
↪
↪'6-
↪2
↪',
↪␣
↪180,
↪␣
↪
↪'Texas', 7730337)
>
↪>
↪>
↪␣
↪print(second_
↪row)
(
↪'Jae␣
↪Crowder
↪',
↪␣
↪
↪'Boston␣
↪Celtics
↪',
↪␣
↪99,
↪␣
↪
↪'SF
↪',
↪␣
↪25,
↪␣
```

(continues on next page)

(continued from previous page)

```
↪
↪'6-
↪6
↪',
↪␣
↪235,
↪␣
↪
↪'Marquette', 6796117)
>
↪>
↪>
↪␣
↪print(third_
↪and_
↪fourth_
↪rows)
[(
↪'John␣
↪Holland
↪',
↪␣
↪
↪'Boston␣
↪Celtics
↪',
↪␣
↪30,
↪␣
↪
↪'SG
↪',
↪␣
↪27,
↪␣
↪
↪'6-
↪5
↪',
↪␣
↪205,
↪␣
↪
↪'Boston University', None), ('R.J. Hunter', 'Boston Celtics', 28, 'SG', 22, '6-5',␣
↪185, 'Georgia State', 1148640)]
>
↪>
↪>
↪␣
↪print(remaining_
↪rows)
[(
↪'Jonas␣
↪Jerebko
↪',
↪␣
↪
```

(continues on next page)

(continued from previous page)

```
↪'Boston␣
↪Celtics
↪',
↪␣
↪8,
↪␣
↪
↪'PF
↪',
↪␣
↪29,
↪␣
↪
↪'6-
↪10
↪',
↪␣
↪231,
↪␣
↪None,
↪ 5000000), ('Amir Johnson', 'Boston Celtics', 90, 'PF', 29, '6-9', 240, None,␣
↪12000000), ('Jordan Mickey', 'Boston Celtics', 55, 'PF', 21, '6-8', 235, 'LSU',␣
↪1170960), ('Kelly Olynyk', 'Boston Celtics', 41, 'C', 25, '7-0', 238, 'Gonzaga',␣
↪2165160),
[.
↪.
↪.
↪]
```

Note: Calling a fetch command after all rows have been fetched will return an empty array (`[]`).

8.2.1.2.3.2 Reading Result Metadata

When you execute a statement, the connection object also

contains metadata about the result set, such as **column names**, **types**, etc).

The metadata is stored in the `Connection.description` object of the cursor:

```
import␣
↪pysqream


con␣
↪=␣
↪pysqream.
↪connect(host=
↪'127.
↪0.
↪0.
↪1
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪port=3108,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪database=
↪'master
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪username=
↪'rhendricks
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪password=
↪'Tr0ub4dor&
↪3
↪',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪clustered=True)
cur␣
↪=␣
↪con.
↪cursor()
statement␣
↪=␣
↪
↪'SELECT␣
↪*␣
↪FROM␣
↪nba
↪'
cur.
↪execute(statement)
print(cur.
↪description)

↪#␣
↪[(
↪'Name
↪',
↪␣
↪
↪'STRING
↪',
↪␣
↪24,
↪␣
↪24,
↪␣
↪None,
↪␣
↪None,
↪␣
↪True),
↪␣
↪(
↪'Team
↪',
↪␣
↪
↪'STRING
↪', 22, 22, None, None, True), ('Number', 'NUMBER', 1, 1, None, None, True), (
↪'Position', 'STRING', 2, 2, None, None, True), ('Age (as of 2018)', 'NUMBER', 1, 1,␣
↪None, None, True), ('Height', 'STRING', 4, 4, None, None, True), ('Weight', 'NUMBER
↪', 2, 2, None, None, True), ('College', 'STRING', 21, 21, None, None, True), (
↪'Salary', 'NUMBER', 4, 4, None, None, True)]
```

You can fetch a list of

column names by iterating over the `description` list:

```
>
↪>
↪>
↪␣
↪[␣
↪i[0]␣
↪for␣
↪i␣
↪in␣
↪cur.
↪description␣
↪]
[
↪'Name
↪',
↪␣
↪
↪'Team
↪',
↪␣
↪
↪'Number
↪',
↪␣
↪
↪'Position
↪',
↪␣
↪
↪'Age␣
↪(as␣
↪of␣
↪2018)
↪',
↪␣
↪
↪'Height
↪',
↪ 'Weight', 'College', 'Salary']
```

8.2.1.2.3.3 Loading Data into a Table

This example shows how to load 10,000 rows of dummy data to an instance of SQream.

To load data 10,000 rows of dummy data to an instance of SQream:

1. Run the following:

```
import
↪pysqream
from
↪datetime
↪import
```

(continues on next page)

(continued from previous page)

```
↪date,
↪␣
↪datetime
from␣
↪time␣
↪import␣
↪time


con␣
↪=␣
↪pysqream.
↪connect(host=
↪'127.
↪0.
↪0.
↪1
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪port=3108,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪database=
↪'master
↪',
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
```

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪password=
↪'Tr0ub4dor&
↪3
↪',
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪clustered=True)

cur␣
↪=␣
↪con.
↪cursor()
```

2. Create a table for loading:

```
create␣
↪=␣
↪
↪'create␣
↪or␣
```

(continues on next page)

(continued from previous page)

```
↪replace␣
↪table␣
↪perf␣
↪(b␣
↪bool,
↪␣
↪t␣
↪tinyint,
↪␣
↪sm␣
↪smallint,
↪␣
↪i␣
↪int,
↪␣
↪bi␣
↪bigint,
↪␣
↪f␣
↪real,
↪␣
↪d double, s text(12), ss text, dt date, dtt datetime)'
cur.
↪execute(create)
```

3. Create a session:

```
session␣
↪=␣
↪orm.
↪sessionmaker(bind=engine)()
```

4. Load your data into table using the `INSERT` command.

5. Create dummy

data matching the table you created:

```
data␣
↪=␣
↪(False,
↪␣
↪2,
↪␣
↪12,
↪␣
↪145,
↪␣
↪84124234,
↪␣
↪3.
↪141,
↪␣
↪-
↪4.
↪3,
↪␣
↪
↪"Marty␣
↪McFly
↪"␣
↪,
↪␣
↪u
↪"□□□□□□□□□□□" , date(2019, 12, 17), datetime(1955, 11, 4, 1, 23, 0, 0))

row_
↪count␣
↪=␣
↪10**4
```

6. Get a new cursor:

```
insert␣
↪=␣
↪
↪'insert␣
↪into␣
↪perf␣
↪values␣
```

(continues on next page)

(continued from previous page)

```
↪(?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,
↪?
↪,?)'
start␣
↪=␣
↪time()
cur.
↪executemany(insert,
↪␣
↪[data]␣
↪*␣
↪row_
↪count)
print␣
↪(f
↪"Total␣
↪insert␣
↪time␣
↪for␣
↪
↪{row_
↪count}
↪␣
↪rows:␣
↪
↪{time()␣
↪-
↪␣
↪start}
↪␣
↪seconds
↪")
```

7. Close this cursor:

```
cur.
↪close()
```

8. Verify that the data was inserted correctly:

```
cur␣
↪=␣
↪con.
↪cursor()
cur.
↪execute(
↪'select␣
↪count(*)␣
↪from␣
↪perf
↪')
result␣
↪=␣
↪cur.
↪fetchall()␣
↪
↪#␣
↪`fetchall`␣
↪collects␣
↪the␣
↪entire␣
↪data␣
↪set
print␣
↪(f
↪"Count␣
↪of␣
↪inserted␣
↪rows:␣
↪
↪{result[0][0]}
↪
↪")
```

9. Close the cursor:

```
cur.
```

(continues on next page)

(continued from previous page)

```
↪close()
```

10. Close the connection:

```
con.
↪close()
```

8.2.1.2.3.4 Using SQLAlchemy ORM to Create and Populate Tables

This section shows how to use the ORM to create and populate tables from Python objects.

To use SQLAlchemy ORM

to create and populate tables:

1. Run the following:

```
import␣
↪sqlalchemy␣
↪as␣
↪sa
import␣
↪pandas␣
↪as␣
↪pd

engine_
↪url␣
↪=␣
↪
↪"sqream:/
↪/
↪rhendricks:secret_
↪password@localhost:5000/
↪raviga
↪"

engine␣
↪=␣
↪sa.
↪create_
↪engine(engine_
↪url)
```

2. Build a metadata object and bind it:

```
metadata␣
```

(continues on next page)

(continued from previous page)

```
↪=␣
↪sa.
↪MetaData()
metadata.
↪bind␣
↪=␣
↪engine
```

3. Create a table in the local metadata:

```
employees␣
↪=␣
↪sa.
↪Table(
␣
↪␣
↪␣
↪␣
↪
↪'employees
↪',
↪
␣
↪␣
↪␣
↪␣
↪metadata,
↪
␣
↪␣
↪␣
↪␣
↪sa.
↪Column(
↪'id
↪',
↪␣
↪sa.
↪Integer),
↪
␣
↪␣
↪␣
↪␣
↪sa.
```

(continues on next page)

(continued from previous page)

```
↪Column(
↪'name
↪',
↪␣
↪sa.
↪TEXT(32)),
↪
␣
↪␣
↪␣
↪␣
↪sa.
↪Column(
↪'lastname
↪',
↪␣
↪sa.
↪TEXT(32)),
↪
␣
↪␣
↪␣
↪␣
↪sa.
↪Column(
↪'salary
↪',
↪␣
↪sa.
↪Float)
)
```

The `create_all()` function uses the SQream engine object.

4. Create all the defined table objects:

```
metadata.
↪create_
↪all(engine)
```

5. Populate your table.

6. Build the data rows:

```
insert_
↪data␣
↪=␣
↪[␣
↪
↪{
↪'id
↪':␣
↪1,
↪␣
↪
↪'name
↪':␣
↪
↪'Richard
↪',
↪
↪'lastname
↪':␣
↪
↪'Hendricks
↪',
↪␣
↪␣
↪␣
↪
↪'salary': 12000.75},
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪
↪{
↪'id
↪':␣
↪3,
↪␣
↪␣
↪
↪'name
↪':␣
↪'Bertram', 'lastname': 'Gilfoyle', 'salary': 8400.0},
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪{
↪'id
↪':␣
↪8,
↪␣
↪␣
↪
↪'name
↪':␣
↪'Donald', 'lastname': 'Dunn', 'salary': 6500.40}]
```

7. Build the INSERT command:

```
ins␣
↪=␣
↪employees.
↪insert(insert_
↪data)
```

8. Execute the command:

```
result␣
↪=␣
↪session.
↪execute(ins)
```

8.2.1.3 Node.JS

The SQream DB Node.JS driver allows Javascript applications and tools connect to SQream DB. This tutorial shows you how to write a Node application using the Node.JS interface.

The driver requires Node 10 or newer.

In this topic:

- *Installing the Node.JS driver*
  - *Prerequisites*
  - *Install with NPM*
  - *Install from an offline package*
- *Connect to SQream DB with a Node.JS application*
  - *Create a simple test*
  - *Run the test*
- *API reference*

- *Connection parameters*
- *Events*
* *Example*
- *Input placeholders*
- *Examples*
- *Setting configuration flags*
- *Lazyloading*
- *Reusing a connection*
- *Using placeholders in queries*
- *Troubleshooting and recommended configu-*

ration

Preventing heap out of memory errors

BIGINT support

8.2.1.3.1 Installing the Node.JS driver

8.2.1.3.1.1 Prerequisites

- Node.JS 10 or newer. Follow instructions at nodejs.org.

8.2.1.3.1.2 Install with NPM

Installing with npm is the easiest

and most reliable method. If you need to install the driver in an offline system, see the offline method below.

```
↪$␣
↪npm␣
↪install␣
↪@sqream/
↪sqreamdb
```

8.2.1.3.1.3 Install from an offline package

The Node driver is provided as a tarball for download from the SQream Drivers page.

After downloading the tarball, use `npm` to install the offline package.

```
↪$␣
↪sudo␣
↪npm␣
↪install␣
↪sqreamdb-
↪4.
↪0.
↪0.
↪tgz
```

8.2.1.3.2 Connect to SQream DB with a Node.JS application

8.2.1.3.2.1 Create a simple test

Replace the connec-

tion
pa-
ram-
e-
ters
with
real
pa-
ram-
e-
ters
for
a
SQream
DB
in-
stal-
la-
tion.

Listing 2: sqreamdb-test.js

```
const␣
↪Connection␣
↪=␣
↪require(
↪'@sqream/
↪sqreamdb
↪');
↪

const␣
↪config␣
↪␣
↪=␣
↪␣
↪
↪{
␣
↪␣
↪host:␣
↪
↪'localhost
↪',
↪
␣
↪␣
↪port:␣
↪3109,
↪
␣
↪␣
↪username:␣
↪
↪'rhendricks
↪',
```

(continues on next page)

(continued from previous page)

```
↪
␣
↪␣
↪password:␣
↪
↪'super_
↪secret_
↪password
↪',
↪
␣
↪␣
↪connectDatabase:␣
↪
↪'raviga
↪',
↪
␣
↪␣
↪cluster:␣
↪true,
↪
␣
↪␣
↪is_
↪ssl:␣
↪true,
↪
␣
↪␣
↪service:␣
↪
↪'sqream
↪'
␣
↪␣
↪}
↪;
↪

const␣
↪query1␣
↪␣
↪=␣
↪␣
↪
↪'SELECT␣
↪1␣
↪AS␣
↪test,
↪␣
↪2*6␣
↪AS␣
↪
↪"dozen
↪"
↪';
```

(continues on next page)

(continued from previous page)

```
↪

const␣
↪sqream␣
↪=␣
↪new␣
↪Connection(config);
↪
sqream.
↪execute(query1).
↪then((data)␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪␣
↪console.
↪log(data);
↪
}
↪,
↪␣
↪(err)␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪␣
↪console.
↪error(err);
↪
}
↪);
↪
```

8.2.1.3.2.2 Run the test

A successful run should look like this:

```
↪$␣
↪node␣
↪sqreamdb-
↪test.
↪js
[␣
↪␣
↪
↪{␣
↪test:␣
↪1,
↪␣
↪dozen:␣
↪12␣
↪␣
↪}
↪␣
↪␣
↪]
```

8.2.1.3.3 API reference

8.2.1.3.3.1 Connection parameters

Item	Optional	Default	Description
`host`	✗	None	Hostname for SQream DB worker. For example, `127.0.0.1`, `sqream.mynetwork.co`
`port`	✗	None	Port for SQream DB end-point. For example, `3108` for the load balancer, `5000` for a worker.
`username`	✗	None	Username of a role to use for connection. For example, `rhendricks`
`password`	✗	None	Specifies the password of the selected role. For example, `Tr0ub4dor&3`
`connectDatabase`	✗	None	Database name to connect to. For example, `master`
`service`	✓	`sqream`	Specifices service queue to use. For example, `etl`
`is_ssl`	✓	`false`	Specifies SSL for this connection. For example, `true`
`cluster`	✓	`false`	Connect via load balancer (use only if exists, and check port). For example, `true`

8.2.1.3.3.2 Events

The connector handles event returns with an event emitter

getConnectionId

The `getConnectionId` event returns the executing connection ID.

getStatementId

The `getStatementId` event returns the executing statement ID.

getTypes

The `getTypes` event returns the results columns types.

8.2.1.3.3.3 Example

```
const
 ↪myConnection
 ↪
 ↪=
 ↪new
 ↪Connection(config);
 ↪

myConnection.
 ↪runQuery(query1,
 ↪
 ↪function
 ↪
 ↪(err,
 ↪
 ↪data)
 ↪{
 ↪
 ↪myConnection.
 ↪events.
 ↪on(
 ↪'getConnectionId
 ↪',
 ↪
 ↪function(data)
 ↪{
 ↪
 ↪
 ↪
 ↪
 ↪
 ↪console.
 ↪log(
 ↪'getConnectionId
 ↪',
 ↪
 ↪data);
 ↪
```

(continues on next page)

(continued from previous page)

```
↪␣
↪}
↪);
↪

␣
↪␣
↪myConnection.
↪events.
↪on(
↪'getStatementId
↪',
↪␣
↪function(data)
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪console.
↪log(
↪'getStatementId
↪',
↪␣
↪data);
↪
␣
↪␣
↪}
↪);
↪

␣
↪␣
↪myConnection.
↪events.
↪on(
↪'getTypes
↪',
↪␣
↪function(data)
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪console.
↪log(
↪'getTypes
↪',
↪␣
↪data);
↪
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪}
↪);
↪
}
↪);
↪
```

8.2.1.3.3.4 Input placeholders

The Node.JS driver can replace parameters in a statement.

Input placeholders allow values like user input to be passed as parameters into queries,

with proper escaping.

The valid placeholder formats are provided in the table below.

Placeholder	Type
`%i`	Identifier (e.g. table name, column name)
`%s`	A text string
`%d`	A number value
`%b`	A boolean value

See the *input placeholders example* below.

8.2.1.3.4 Examples

8.2.1.3.4.1 Setting configuration flags

SQreamDB

configuration flags can be set per statement, as a parameter to `runQuery`.

For example:

```
const setFlag = 'SET showfullexceptioninfo = true;';

const query_string = 'SELECT 1';

const myConnection = new
```

(continues on next page)

(continued from previous page)

```
↪Connection(config);
↪
myConnection.
↪runQuery(query_
↪string,
↪␣
↪function␣
↪␣
↪(err,
↪␣
↪data)
↪{
␣
↪␣
↪console.
↪log(err,
↪␣
↪data);
↪
}
↪,
↪␣
↪setFlag);
↪
```

8.2.1.3.4.2 Lazyloading

To process rows without keeping them in memory, you can lazyload the rows with an async:

```
const␣
↪Connection␣
↪=␣
↪require(
```

(continues on next page)

(continued from previous page)

```
↪'@sqream/
↪sqreamdb
↪');
↪

const␣
↪config␣
↪␣
↪=␣
↪␣
↪
↪{
␣
↪␣
↪host:␣
↪
↪'localhost
↪',
↪
␣
↪␣
↪port:␣
↪3109,
↪
␣
↪␣
↪username:␣
↪
↪'rhendricks
↪',
↪
␣
↪␣
↪password:␣
↪
↪'super_
↪secret_
↪password
↪',
↪
␣
↪␣
↪connectDatabase:␣
↪
↪'raviga
↪',
↪
␣
↪␣
↪cluster:␣
↪true,
↪
␣
↪␣
↪is_
↪ssl:␣
↪true,
```

(continues on next page)

(continued from previous page)

```
↪
␣
↪␣
↪service:␣
↪
↪'sqream
↪'
␣
↪␣
↪}
↪;
↪

const␣
↪sqream␣
↪=␣
↪new␣
↪Connection(config);
↪

const␣
↪query␣
↪=␣
↪
↪"SELECT␣
↪*␣
↪FROM␣
↪public.
↪a_
↪very_
↪large_
↪table
↪";
↪

(async␣
↪()␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪const␣
↪cursor␣
↪=␣
↪await␣
↪sqream.
↪executeCursor(query);
↪
␣
↪␣
↪let␣
↪count␣
↪=␣
↪0;
↪
```

(continues on next page)

(continued from previous page)

```
␣
↪␣
↪for␣
↪await␣
↪(let␣
↪rows␣
↪of␣
↪cursor.
↪fetchIterator(100))␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪fetch␣
↪rows␣
↪in␣
↪chunks␣
↪of␣
↪100
␣
↪␣
↪␣
↪␣
↪count␣
↪+=␣
↪rows.
↪length;
↪
␣
↪␣
↪}
↪
␣
↪␣
↪await␣
↪cursor.
↪close();
↪
␣
↪␣
↪return␣
↪count;
↪
}
↪)().
↪then((total)␣
↪=>
↪␣
↪
↪{
␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪console.
↪log(
↪'Total␣
↪rows
↪',
↪␣
↪total);
↪
}
↪,
↪␣
↪(err)␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪console.
↪error(err);
↪
}
↪);
↪
```

8.2.1.3.4.3 Reusing a connection

It is possible to execute multiple queries with the same connection (although only

one
query
can
be
executed at a time).

```
const␣
↪Connection␣
↪=␣
↪require(
↪'@sqream/
↪sqreamdb
↪');
↪

const␣
↪config␣
↪␣
↪=␣
↪␣
↪
↪{
␣
↪␣
↪host:␣
↪
↪'localhost
↪',
↪
␣
↪␣
↪port:␣
↪3109,
↪
␣
↪␣
↪username:␣
↪
↪'rhendricks
↪',
↪
␣
↪␣
↪password:␣
↪
↪'super_
↪secret_
↪password
↪',
↪
␣
↪␣
↪connectDatabase:␣
↪
↪'raviga
↪',
↪
␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪cluster:␣
↪true,
↪
␣
↪␣
↪is_
↪ssl:␣
↪true,
↪
␣
↪␣
↪service:␣
↪
↪'sqream
↪'
␣
↪␣
↪}
↪;
↪

const␣
↪sqream␣
↪=␣
↪new␣
↪Connection(config);
↪

(async␣
↪()␣
↪=>
↪␣
↪
↪{

␣
↪␣
↪const␣
↪conn␣
↪=␣
↪await␣
↪sqream.
↪connect();
↪
␣
↪␣
↪try␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪const␣
↪res1␣
↪=␣
```

(continues on next page)

(continued from previous page)

```
↪await␣
↪conn.
↪execute(
↪"SELECT␣
↪1
↪");
↪
␣
↪␣
↪␣
↪␣
↪const␣
↪res2␣
↪=␣
↪await␣
↪conn.
↪execute(
↪"SELECT␣
↪2
↪");
↪
␣
↪␣
↪␣
↪␣
↪const␣
↪res3␣
↪=␣
↪await␣
↪conn.
↪execute(
↪"SELECT␣
↪3
↪");
↪
␣
↪␣
↪␣
↪␣
↪conn.
↪disconnect();
↪
␣
↪␣
↪␣
↪␣
↪return␣
↪
↪{res1,
↪␣
↪res2,
↪␣
↪res3}
↪;
↪
␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪}
↪␣
↪catch␣
↪(err)␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪conn.
↪disconnect();
↪
␣
↪␣
↪␣
↪␣
↪throw␣
↪err;
↪
␣
↪␣
↪}
↪

}
↪)().
↪then((res)␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪console.
↪log(
↪'Results
↪',
↪␣
↪res)
}
↪,
↪␣
↪(err)␣
↪=>
↪␣
↪
↪{
␣
↪␣
↪console.
↪error(err);
↪
}
↪);
↪
```

8.2.1.3.4.4 Using placeholders in queries

Input placeholders allow values like user input to be passed as parameters into queries, with proper escaping.

```
const␣
↪Connection␣
↪=␣
↪require(
↪'@sqream/
↪sqreamdb
↪');
↪

const␣
↪config␣
↪␣
↪=␣
↪␣
↪
↪{
␣
↪␣
↪host:␣
↪
```

(continues on next page)

(continued from previous page)

```
↪'localhost
↪',
↪
␣
↪␣
↪port:␣
↪3109,
↪
␣
↪␣
↪username:␣
↪
↪'rhendricks
↪',
↪
␣
↪␣
↪password:␣
↪
↪'super_
↪secret_
↪password
↪',
↪
␣
↪␣
↪connectDatabase:␣
↪
↪'raviga
↪',
↪
␣
↪␣
↪cluster:␣
↪true,
↪
␣
↪␣
↪is_
↪ssl:␣
↪true,
↪
␣
↪␣
↪service:␣
↪
↪'sqream
↪'
␣
↪␣
↪}
↪;
↪

const␣
↪sqream␣
↪=␣
```

(continues on next page)

(continued from previous page)

```
↪new␣
↪Connection(config);
↪

const␣
↪sql␣
↪=␣
↪
↪"SELECT␣
↪
↪%i␣
↪FROM␣
↪public.
↪
↪%i␣
↪WHERE␣
↪name␣
↪=␣
↪
↪%s␣
↪AND␣
↪num␣
↪>
↪␣
↪
↪%d␣
↪AND␣
↪active␣
↪=␣
↪
↪%b";

sqream.
↪execute(sql,
↪␣
↪
↪"col1
↪",
↪␣
↪
↪"table2
↪",
↪␣
↪
↪"john
↪'s
↪",
↪␣
↪50,
↪␣
↪true);
↪
```

The query that will

run is `SELECT col1 FROM public.table2 WHERE name = 'john''s' AND num > 50 AND active = true`

8.2.1.3.5 Troubleshooting and recommended configuration

8.2.1.3.5.1 Preventing `heap out of memory` errors

Some workloads may cause Node.JS to fail with the

error:

```
FATAL␣
↪ERROR:␣
↪CALL_
↪AND_
↪RETRY_
↪LAST␣
↪Allocation␣
↪failed␣
↪-
↪␣
↪JavaScript␣
↪heap␣
↪out␣
↪of␣
↪memory
```

To prevent this error, modify the heap size configuration by setting the `--max-old-space-size` run flag.

For example, set the space size to 2GB:

(continues on next page)

(continued from previous page)

```
↪$␣
↪node␣
↪-
↪-
↪max-
↪old-
↪space-
↪size=2048␣
↪my-
↪application.
↪js
```

8.2.1.3.5.2 BIGINT support

The Node.JS connector supports fetching `BIGINT` values from SQream DB. However, some applications may encounter an error when trying to serialize those values.

The error that appears is:

..
code-

block::
none

TypeError: Do not know how to serialize a BigInt

This is because JSON specification do not support BIGINT values, even when supported by Javascript engines.

To resolve this issue, objects

with BIGINT values should be converted to string before serializing, and converted back after deserializing.

For example:

```
const␣
↪rows␣
↪=␣
↪[
↪{test:␣
↪1n}
↪]
const␣
↪json␣
↪=␣
↪JSON.
↪stringify(rows,
↪␣
↪,
↪␣
↪(key,
↪␣
↪value)␣
↪=>
↪
␣
↪␣
↪typeof␣
↪value␣
↪===␣
↪
↪'bigint
↪'
␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪?
↪␣
↪value.
↪toString()
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪:␣
↪value␣
↪/
↪/
↪␣
↪return␣
↪everything␣
↪else␣
↪unchanged
));
↪
console.
↪log(json);
↪␣
↪/
↪/
↪␣
↪[
↪{
↪"test
↪":␣
↪
↪"1
↪"}
↪]
```

8.2.1.4 ODBC

8.2.1.4.1 Install and Configure ODBC on Windows

The ODBC driver

for Windows is provided as a self-contained installer.

This tutorial shows you how to install and configure ODBC on Windows.

- *Installing the ODBC Driver*
- *Configuring the ODBC Driver DSN*
- *Troubleshooting*
- *Limitations*

8.2.1.4.1.1 Installing the ODBC Driver

8.2.1.4.1.2 Prerequisites

8.2.1.4.1.3 Visual Studio 2015 Redistributables

To install the ODBC driver you must first install Microsoft's **Visual C++ Redistributable for Visual Studio 2015**. To install Visual C++ Redistributable for Visual Studio 2015, see the Install Instructions.

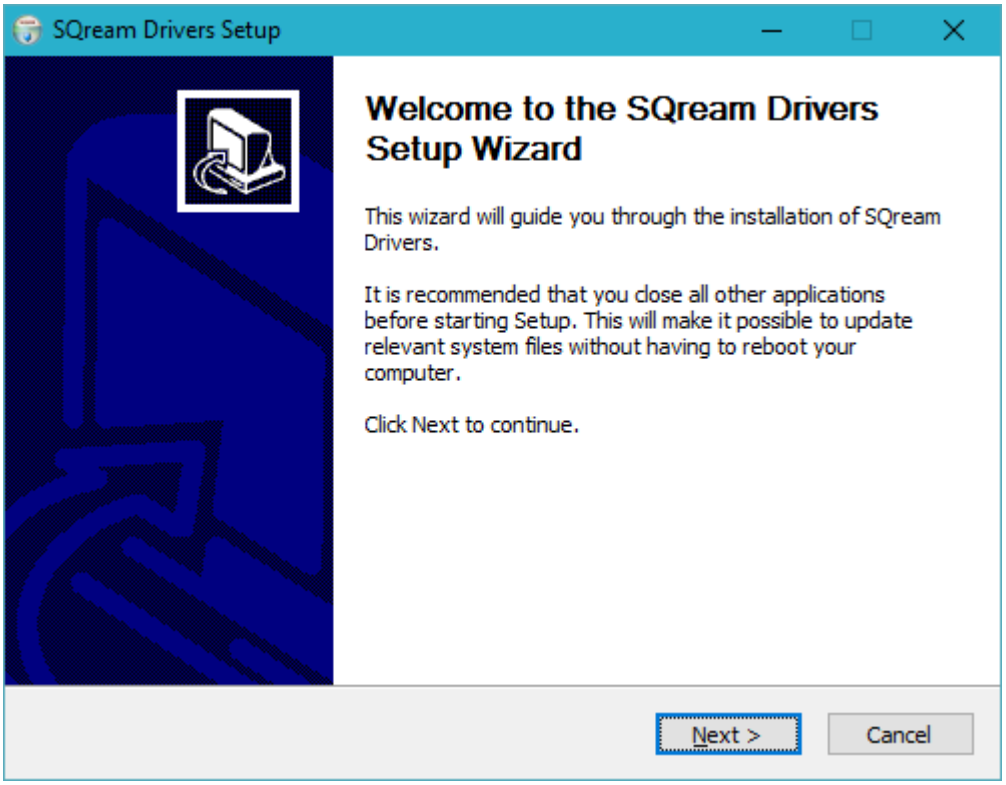
8.2.1.4.1.4 Administrator Privileges

The SQreamDB ODBC driver requires administrator privileges on your computer to add the DSNs (data source names).

8.2.1.4.1.5 Running the Windows Installer

Install the driver by following the on-screen instructions

in the easy-to-follow installer.

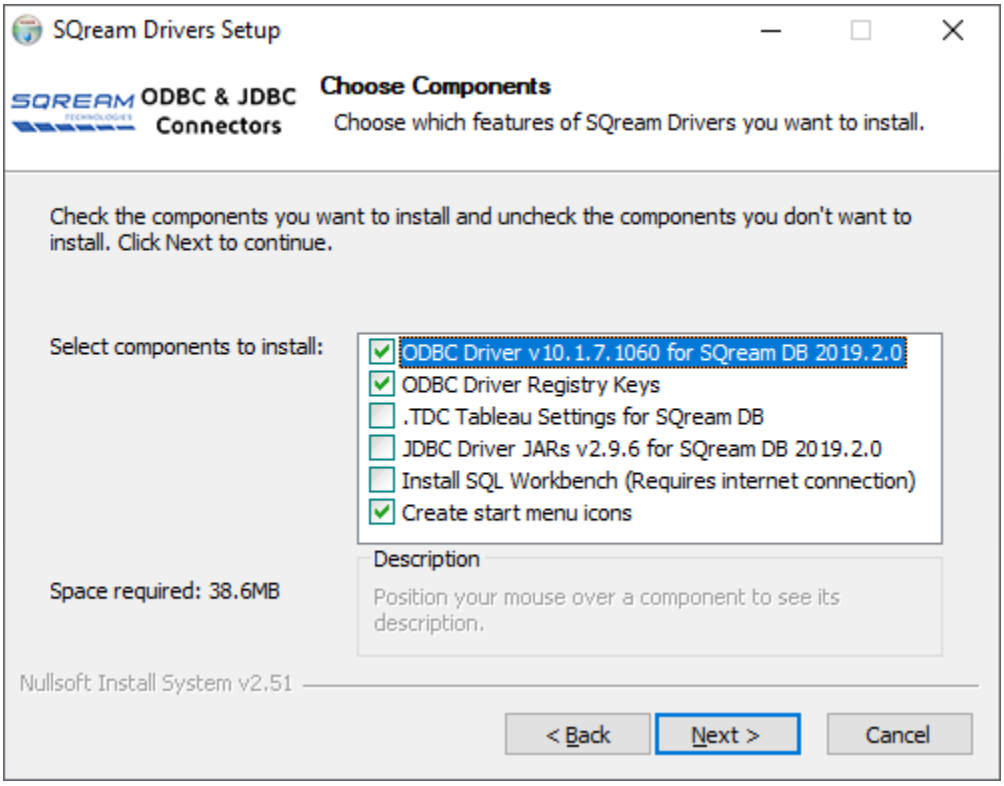


Note: The installer will install the driver in `C:\Program Files\SQream Technologies\ODBC Driver` by default. This path is changable during the installation.

8.2.1.4.1.6 Selecting Components

The installer includes additional components

nents, like JDBC and Tableau customizations.



You can deselect items you don't want to install, but the items named **ODBC Driver DLL**

and **ODBC Driver Registry Keys** must remain selected for a complete installation of the ODBC driver.

Once the installer finishes, you will be ready to configure the DSN for connection.

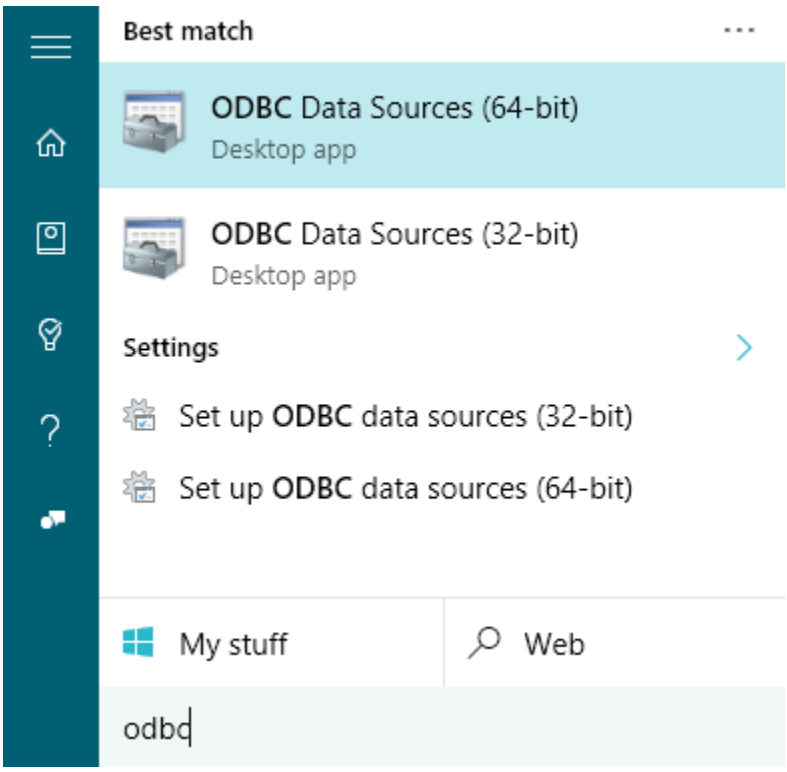
8.2.1.4.1.7 Configuring the ODBC Driver DSN

ODBC driver configurations are done via DSNs. Each DSN represents one SQream

DB database.

1. Open up the Windows menu by clicking the Windows button on your keyboard (⊞ `Win`) or pressing the Windows button with your mouse.

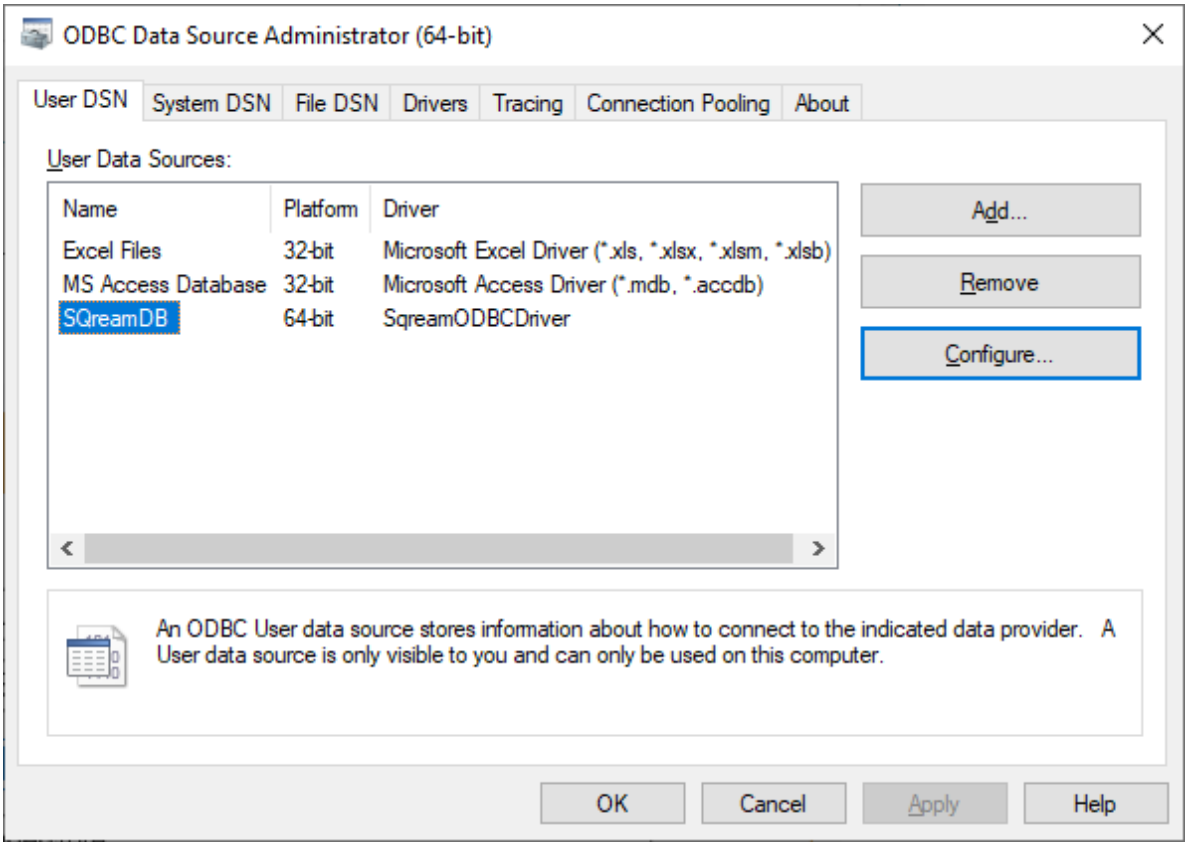
2. Type **ODBC** and select **ODBC Data Sources (64-bit)**. Click the item to open up the setup window.



3. The installer has created a sample User DSN named **SQreamDB**

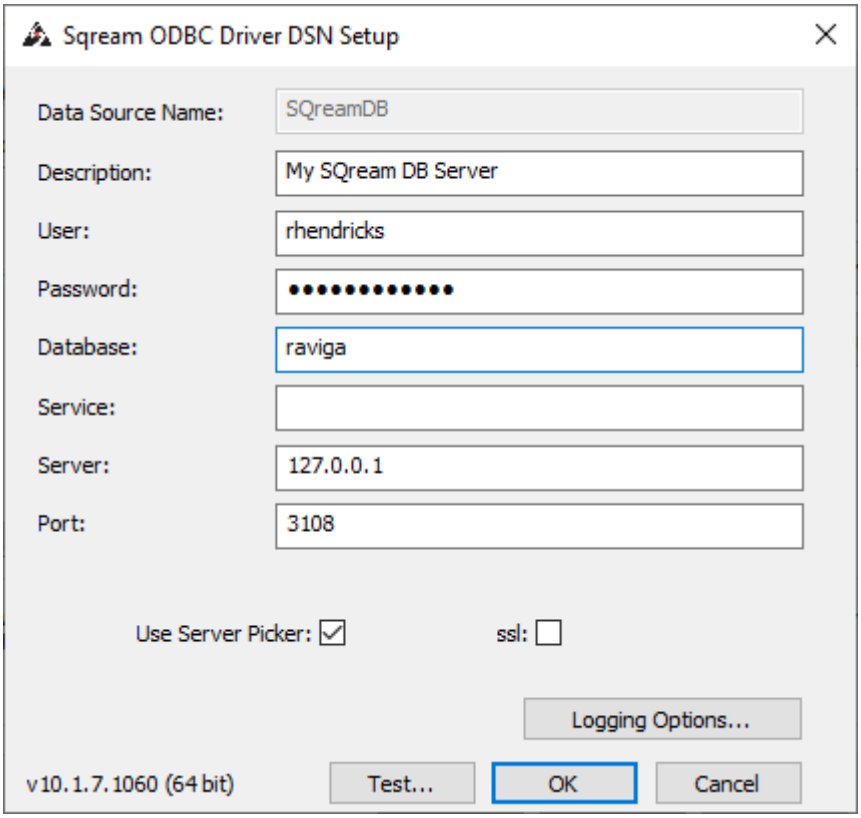
   You can modify this DSN, or create a new one (*Add ▸ SQream*

ODBC Driver ▸ *Next*)



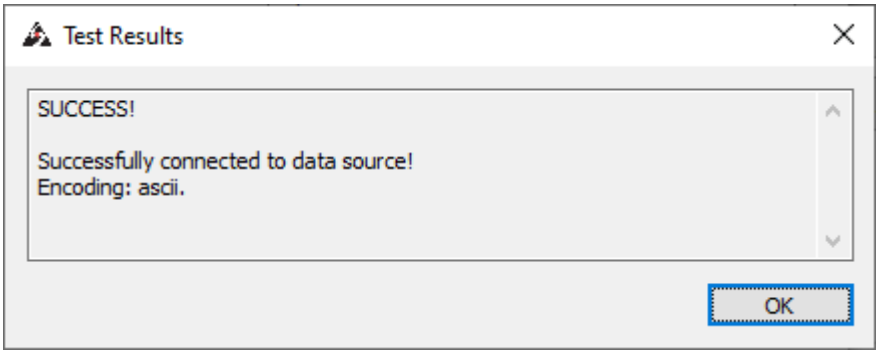
4. Enter your connection parameters. See the reference below for a descrip-

tion of the parameters.



5. When completed, save the DSN by selecting *OK*

Tip: Test the connection by clicking *Test* before saving. A successful test looks like this:



1. You can now use this DSN in ODBC applications like Tableau.

8.2.1.4.1.8 Connection Parameters

Item	Description
Data Source Name	An easily recognizable name that you'll use to reference this DSN. Once you set this, it can not be changed.
Description	A description of this DSN for your convenience. You can leave this blank.
User	Username of a role to use for connection. For example, `rhendricks`
Password	Specifies the password of the selected role. For example, `Tr0ub4dor&3`
Database	Specifies the database name to connect to. For example, `master`
Service	Specifies *service queue* to use. For example, `etl`. Leave blank for default service `sqream`.
Server	Hostname of the SQream DB worker. For example, `127.0.0.1` or `sqream.mynetwork.co`
Port	TCP port of the SQream DB worker. For example, `5000` or `3108`
User server picker	Connect via load balancer (use only if exists, and check port)
SSL	Specifies SSL for this connection
Logging options	Use this screen to alter logging options when tracing the ODBC connection for possible connection issues.

8.2.1.4.1.9 Troubleshooting

8.2.1.4.1.10 Solving "Code 126" ODBC errors

After installing the ODBC driver, you may experience the following error:

```
The
↪setup
↪routines
↪for
↪the
↪SQreamDriver64
↪ODBC
↪driver
↪could
↪not
↪be
↪loaded
↪due
↪to
↪system
↪error
code
↪126:
↪The
↪specified
↪module
↪could
↪not
↪be
↪found.
↪
```

(continues on next page)

(continued from previous page)

```
(c:\
↪Program␣
↪Files\
↪SQream␣
↪Technologies\
↪ODBC␣
↪Driver\
↪sqreamOdbc64.
↪dll)
```

This is an issue with the Visual Studio Redistributable packages. Verify you've correctly installed them, as described in the *Visual Studio 2015 Redistributables* section above.

8.2.1.4.1.11 Limitations

Please note that the SQreamDB ODBC connector does not sup-

port the use of ARRAY data types. If your database schema includes ARRAY columns, you may encounter compatibility issues when using ODBC to connect to the database.

8.2.1.4.2 Install and configure ODBC on Linux

The ODBC driver for Windows is provided as a shared library.

This tutorial shows how to install and configure

ODBC on Linux.

- *Prerequisites*
- *Install the ODBC driver with a script*
- *Install the ODBC driver manually*
- *Install the driver dependencies*
- *Testing the connection*
- *ODBC DSN Parameters*
- *Limitations*

8.2.1.4.2.1 Prerequisites

8.2.1.4.2.2 unixODBC

The ODBC driver requires a driver manager to manage the DSNs. SQream DB's driver is built for unixODBC.

Verify unixODBC is installed by running:

```
↪$␣
↪odbcinst␣
↪-
↪j
unixODBC␣
↪2.
↪3.
↪4
DRIVERS.
↪.
↪.
↪.
↪.
↪.
↪.
↪.
↪.
↪.
```

(continues on next page)

(continued from previous page)

```
↪.
↪.
↪:␣
↪/
↪etc/
↪odbcinst.
↪ini
SYSTEM␣
↪DATA␣
↪SOURCES:␣
↪/
↪etc/
↪odbc.
↪ini
FILE␣
↪DATA␣
↪SOURCES.
↪.
↪:␣
↪/
↪etc/
↪ODBCDataSources
USER␣
↪DATA␣
↪SOURCES.
↪.
↪:␣
↪/
↪home/
↪rhendricks/
↪.
↪odbc.
↪ini
SQLULEN␣
↪Size.
↪.
↪.
↪.
↪.
↪.
↪.
↪:␣
↪8
SQLLEN␣
↪Size.
↪.
↪.
↪.
↪.
↪.
↪.
↪.
↪:␣
↪8
SQLSETPOSIROW␣
↪Size.
↪:␣
```

(continues on next page)

(continued from previous page)

```
↪8
```

Take note of the location of `.odbc.ini` and `.odbcinst.ini`. In this case, `/etc`. If `odbcinst` is not installed, follow the instructions for your platform below:

Install unixODBC on:

- *Install unixODBC on RHEL 7 / CentOS 7*
- *Install unixODBC on Ubuntu*

8.2.1.4.2.3 Install unixODBC on RHEL 7 / CentOS 7

```
↪$␣
↪yum␣
↪install␣
↪-
↪y␣
↪unixODBC␣
↪unixODBC-
↪devel
```

8.2.1.4.2.4 Install unixODBC on Ubuntu

```
↪$␣
↪sudo␣
↪apt-
↪get␣
↪install␣
↪odbcinst1debian2␣
↪libodbc1␣
↪odbcinst␣
↪unixodbc
```

8.2.1.4.2.5 Install the ODBC driver with a script

Use this method if you have never used

ODBC on your machine before. If you have existing DSNs, see the manual install process below.

1. Unpack the tarball Copy the downloaded file to any directory, and untar it to a new directory:

```
↪$␣
↪mkdir␣
↪-
↪p␣
↪sqream_
↪odbc64
```

(continues on next page)

(continued from previous page)

```
↪$␣
↪tar␣
↪-
↪xf␣
↪sqream_
↪odbc_
↪vX.
↪Y_
↪x86_
↪64_
↪linux.
↪tar.
↪gz␣
↪␣
↪-
↪-
↪strip-
↪components=1␣
↪␣
↪-
↪C␣
↪sqream_
↪odbc64/
↪
```

2. Run the first-time installer. The installer will create an editable DSN.

```
↪$␣
↪cd␣
↪sqream_
↪odbc64
.
↪/
↪odbc_
↪install.
↪sh␣
↪-
```

(continues on next page)

(continued from previous page)

```
↪-
↪install
```

3. Edit the DSN created by editing `/etc/.odbc.ini`. See the parameter explanation in the section *ODBC DSN Parameters*.

8.2.1.4.2.6 Install the ODBC driver manually

Use this method when you have existing ODBC DSNs on your

machine.

1. Unpack the tarball Copy the file you downloaded to the directory where you want to install it, and untar it:

```
↪$␣
↪tar␣
↪xf␣
↪sqream_
↪2019.
↪2.
↪1_
↪odbc_
↪3.
↪0.
↪0_
↪x86_
↪64_
↪linux.
↪tar.
↪gz␣
↪-
↪C␣
↪sqream_
↪odbc64
```

Take note of

the directory where the driver was unpacked. For example, `/home/rhendricks/sqream_odbc64`

2. Locate the `.odbc.ini` and `.odbcinst.ini` files, using `odbcinst -j`.

1. In `.odbcinst.ini`, add the following lines to register the driver (change the

high-
lighted
paths
to
match
your
spe-
cific driver):

```
[ODBC␣
↪Drivers]
SqreamODBCDriver=Installed

[SqreamODBCDriver]
Description=Driver␣
↪DSII␣
↪SqreamODBC␣
↪64bit
Driver=/
↪home/
↪rhendricks/
↪sqream_
↪odbc64/
↪sqream_
↪odbc64.
↪so
Setup=/
↪home/
↪rhendricks/
↪sqream_
↪odbc64/
↪sqream_
↪odbc64.
↪so
APILevel=1
ConnectFunctions=YYY
DriverODBCVer=03.
↪80
SQLLevel=1
IconvEncoding=UCS-
↪4LE
```

2. In `.odbc.ini`, add the following lines to configure

the DSN (change the highlighted parameters to match your installation):

```
[ODBC␣
↪Data␣
↪Sources]
MyTest=SqreamODBCDriver

[MyTest]
Description=64-
↪bit␣
↪Sqream␣
↪ODBC
Driver=/
↪home/
↪rhendricks/
↪sqream_
↪odbc64/
↪sqream_
↪odbc64.
↪so
Server=
↪"127.
↪0.
↪0.
↪1
↪"
Port=
↪"5000
↪"
Database=
↪"raviga
↪"
Service=
↪"
↪"
User=
↪"rhendricks
↪"
Password=
↪"Tr0ub4dor&
↪3
↪"
Cluster=false
Ssl=false
```

Pa-
ram-

e-
ters
are
in
the
form
of
`pa-`
`ram-`
`e-`
`ter`
`=`
`value`.
For
de-
tails
about
the
pa-
ram-
e-
ters
that
can
be set for each DSN, see the section *ODBC DSN Parameters*.

3. Cre-
ate
a
file
called
`.`
`sqream_odbc.`
`ini`
for
man-
ag-
ing
the
driver
set-
tings
and
log-
ging.
This
file
should
be
cre-
ated
alongside the other files, and add the following lines (change the highlighted parameters to match your installation):

```
↪#␣
↪Note␣
↪that␣
↪this␣
↪default␣
↪DriverManagerEncoding␣
↪of␣
↪UTF-
↪32␣
↪is␣
↪for␣
↪iODBC.
↪␣
↪unixODBC␣
↪uses␣
↪UTF-
↪16␣
↪by␣
↪default.
↪

↪#␣
↪If␣
↪unixODBC␣
↪was␣
↪compiled␣
↪with␣
↪-
↪DSQL_
↪WCHART_
↪CONVERT,
↪␣
↪then␣
↪UTF-
↪32␣
↪is␣
↪the␣
↪correct␣
↪value.
↪

↪#␣
↪Execute␣
↪
↪'odbc_
↪config␣
↪-
↪-
↪cflags
↪'␣
↪to␣
↪determine␣
↪if␣
↪you␣
↪need␣
↪UTF-
```

(continues on next page)

(continued from previous page)

```
↪32␣
↪or␣
↪UTF-
↪16␣
↪on␣
↪unixODBC
[Driver]
DriverManagerEncoding=UTF-
↪16
DriverLocale=en-
↪US
ErrorMessagesPath=/
↪home/
↪rhendricks/
↪sqream_
↪odbc64/
↪ErrorMessages
LogLevel=0
LogNamespace=
LogPath=/
↪tmp/
↪
ODBCInstLib=libodbcinst.
↪so
```

8.2.1.4.2.7 Install the driver dependencies

Add the ODBC driver path to `LD_LIBRARY_PATH`:

```
↪$␣
↪export␣
↪LD_
↪LIBRARY_
↪PATH=
↪$LD_
↪LIBRARY_
↪PATH:/
↪home/
↪rhendricks/
↪sqream_
↪odbc64/
```

(continues on next page)

(continued from previous page)

```
↪lib
```

You can also add this previous command line to your `~/.bashrc` file in order to keep this installation working between reboots without re-entering the command manually

8.2.1.4.2.8 Testing the connection

Test the driver using `isql`.

If the DSN created is called `MyTest` as

the example, run isql in this format:

```
↪$␣
↪isql␣
↪MyTest
```

8.2.1.4.2.9 ODBC DSN Parameters

Item	Default	Description
Data Source Name	None	An easily recognizable name that you'll use to reference this DSN.
Description	None	A description of this DSN for your convenience. This field can be left blank
User	None	Username of a role to use for connection. For example, `User="rhendricks"`
Password	None	Specifies the password of the selected role. For example, `User="Tr0ub4dor&3"`
Database	None	Specifies the database name to connect to. For example, `Database="master"`
Service	`sqream`	Specifices *service queue* to use. For example, `Service="etl"`. Leave blank (`Service=""`) for default service `sqream`.
Server	None	Hostname of the SQream DB worker. For example, `Server="127.0.0.1"` or `Server="sqream.mynetwork.co"`
Port	None	TCP port of the SQream DB worker. For example, `Port="5000"` or `Port="3108"` for the load balancer
Cluster	`false`	Connect via load balancer (use only if exists, and check port). For example, `Cluster=true`
Ssl	`false`	Specifies SSL for this connection. For example, `Ssl=true`
DriverManagerEncoding	`UTF-16`	Depending on how unixODBC is installed, you may need to change this to `UTF-32`.
ErrorMessagesPath	None	Location where the driver was installed. For example, `ErrorMessagePath=/home/rhendricks/sqream_odbc64/ErrorMessages`.
LogLevel	0	Set to 0-6 for logging. Use this setting when instructed to by SQream Support. For example, `LogLevel=1` • 0 = Disable • 2 = Error • Info tracing • 5

8.2.1.4.2.10 Limitations

Please note that the SQreamDB ODBC connector does not support the use of ARRAY data types. If your database schema includes ARRAY columns, you may encounter compatibility issues when using ODBC to connect to the database.

SQream has an ODBC driver to connect to SQream DB. This tutorial shows how to install the ODBC driver for

Linux
or
Windows for use with applications like Tableau, PHP, and others that use ODBC.

Platform	Versions supported
Windows	• Windows 7 (64 bit) • Windows 8 (64 bit) • Windows 10 (64 bit) • Windows Server 2008 R2 (64 bit) • Windows Server 2012 • Windows Server 2016 • Windows Server 2019
Linux	• Red Hat Enterprise Linux (RHEL) 7 • CentOS 7 • Ubuntu 16.04 • Ubuntu 18.04

Other distributions may also work, but are not officially supported by SQream.

8.2.1.4.3 Getting the ODBC driver

The SQream ODBC driver is distributed by

your SQream account manager. Before contacting your account manager, verify which platform the ODBC driver will be used on. Go to SQream Support or contact your SQream account manager to get the driver.

The driver is provided as an executable installer for Windows, or a compressed tarball for Linux platforms. After downloading the driver, follow the relevant instructions to install and configure the driver for your platform:

8.2.1.4.4 Install and configure the ODBC driver

Continue based on your platform:

- *Install and Configure ODBC on Windows*
- *Install and configure ODBC on Linux*

8.2.1.5 SQreamNET

The SQreamNET ADO.NET Data Provider lets you connect to SQream

through your .NET environment.

- *Before You Begin*
- *Integrating SQreamNET*
- *Connecting to SQream For the First Time*
- *Limitations*

8.2.1.5.1 Before You Begin

- The SQreamNET provider requires a .NET version 6 or newer
- Down-

load the SQreamNET driver from the *client drivers page*

8.2.1.5.2 Integrating SQreamNET

1. After downloading the .NET driver, save the archived file to a known location.
2. In your IDE, add a SQreamNET.dll reference to your project.
3. If you wish

to up-grade SQream-NET within an ex-ist-ing project, re-place your ex-ist-ing .dll file with an up-dated one or change the project's reference location to a new one.

8.2.1.5.3 Connecting to SQream For the First Time

An ini-tial con-nec-tion to SQream must be es-tab-lished by cre-at-ing a **Sqream-Con-nec-**

tion
ob-
ject
us-
ing
a connection string.

8.2.1.5.3.1 Connection String Syntax

```
Data␣
↪Source=
↪<hostname␣
↪or␣
↪ip>
↪,
↪
↪<port>
↪;
↪User=
↪<username>
↪;
↪Password=
↪<password>
↪;
↪Initial␣
↪Catalog=
↪<database␣
↪name>
↪;
↪Integrated␣
↪Security=true;
↪
```

8.2.1.5.3.2 Connection Parameters

Item	State	Default	Description
`<data source>`	Mandatory	None	Hostname/IP/FQDN and port of the SQream DB worker. For example, `127.0.0.1:5000`, `sqream.mynetwork.co:3108`
`<initial catalog>`	Mandatory	None	Database name to connect to. For example, `master`
`<username>`	Mandatory	None	Username of a role to use for connection. For example, `username=rhendricks`
`<password>`	Mandatory	None	Specifies the password of the selected role. For example, `password=Tr0ub4dor&3`
`<service>`	Optional	`sqream`	Specifices service queue to use. For example, `service=etl`
`<ssl>`	Optional	`false`	Specifies SSL for this connection. For example, `ssl=true`
`<cluster>`	Optional	`true`	Connect via load balancer (use only if exists, and check port).

8.2.1.5.3.3 Connection String Examples

The following is an example of a SQream cluster with load balancer and no service

queues (with SSL):

```
Data␣
↪Source=sqream.
↪mynetwork.
↪co,
↪3108;
↪User=rhendricks;
↪Password=Tr0ub4dor&
↪3;
↪Initial␣
↪Catalog=master;
↪Integrated␣
↪Security=true;
↪ssl=true;
↪cluster=true;
↪
```

The following is a minimal example for a local standalone SQream database:

```
Data␣
↪Source=127.
↪0.
↪0.
↪1,
↪5000;
↪User=rhendricks;
↪Password=Tr0ub4dor&
↪3;
↪Initial␣
↪Catalog=master;
↪
```

The following

ing is an example of a SQream cluster with load balancer and a specific service queue named `etl`, to the database named `raviga`

```
Data␣
↪Source=sqream.
↪mynetwork.
↪co,
↪3108;
↪User=rhendricks;
↪Password=Tr0ub4dor&
↪3;
↪Initial␣
↪Catalog=raviga;
↪Integrated␣
↪Security=true;
↪service=etl;
↪cluster=true;
↪
```

8.2.1.5.3.4 Sample C# Program

You can download the `.NET Ap-`

pli-
ca-
tion
Sam-
ple
File
be-
low
by
right-
clicking
and
sav-
ing
it
to
your
com-
puter.

Listing 3: .NET Application Sample

```
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪public␣
↪void␣
↪Test()
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
```

(continues on next page)

(continued from previous page)

```
↪connection␣
↪=␣
↪OpenConnection(
↪"192.
↪168.
↪4.
↪62
↪",
↪␣
↪5000,
↪␣
↪
↪"sqream
↪", "sqream", "master");

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪ExecuteSQLCommand(connection,
↪␣
↪
↪"create␣
↪or␣
↪replace␣
↪table␣
↪tbl_
↪example␣
↪as␣
↪select␣
↪1␣
↪as␣
↪x␣
↪, 'a' as y;");

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪tableData␣
```

(continues on next page)

(continued from previous page)

```
↪=␣
↪ReadExampleData(connection,
↪␣
↪
↪"select␣
↪*␣
↪from␣
↪tbl_
↪example;
↪
↪");
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪}
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<summary>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪Builds␣
↪a␣
↪connection␣
↪string␣
↪to␣
↪sqream␣
↪server␣
```

(continues on next page)

(continued from previous page)

```
↪and␣
↪opens␣
↪a␣
↪connection
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪</
↪summary>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"ipAddress
↪">
↪host␣
↪to␣
↪connect
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
```

(continues on next page)

(continued from previous page)

```
↪name=
↪"port
↪">
↪port␣
↪sqreamd␣
↪is␣
↪running␣
↪on
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"username
↪">
↪role␣
↪username
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"password
↪">
↪role␣
↪password
↪</
↪param>
↪
␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"databaseName
↪">
↪database␣
↪name
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"isCluster
↪">
↪optional␣
↪-
↪␣
↪set␣
↪to␣
↪true␣
↪when␣
↪the␣
↪ip,
↪port endpoint is a server picker process</param>
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
```

(continues on next page)

(continued from previous page)

```
↪/
↪␣
↪
↪<returns>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪SQream␣
↪connection␣
↪object
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪Throws␣
↪SqreamException␣
↪if␣
↪fails␣
↪to␣
↪open␣
↪a␣
↪connction
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪</
↪returns>
↪
␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪public␣
↪SqreamConnection␣
↪OpenConnection(string␣
↪ipAddress,
↪␣
↪int␣
↪port,
↪␣
↪string␣
↪username,
↪␣
↪string␣
↪password,
↪␣
↪string␣
↪databaseName,
↪␣
↪bool␣
↪isCluster = false)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪create␣
↪the␣
↪connection␣
↪string␣
↪according␣
↪to␣
```

(continues on next page)

(continued from previous page)

```
↪the␣
↪format
26 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪connectionString␣
↪=␣
↪string.
↪Format(
27 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"Data␣
↪Source=
↪{0}
↪,
↪
↪{1}
↪;
↪User=
↪{2}
↪;Password={3};Initial Catalog={4};Cluster={5}",
28 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪ipAddress,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪port,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪username,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪password,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪databaseName,
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪isCluster
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪);
↪
```

(continues on next page)

(continued from previous page)

```
35
36 ␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪/
   ↪/
   ↪␣
   ↪create␣
   ↪a␣
   ↪sqeram␣
   ↪connection␣
   ↪object
37 ␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪var␣
   ↪connection␣
   ↪=␣
   ↪new␣
   ↪SqreamConnection(connectionString);
   ↪
38
39 ␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪␣
   ↪/
   ↪/
   ↪␣
   ↪open␣
   ↪a␣
```

(continues on next page)

(continued from previous page)

```
↪connection
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪connection.
↪Open();
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪returns␣
↪the␣
↪connection␣
↪object
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪return␣
↪connection;
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪}
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<summary>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪Executes␣
↪a␣
↪SQL␣
↪command␣
↪to␣
↪sqream␣
↪server
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪</
↪summary>
↪
␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"connection
↪">
↪connection␣
↪to␣
↪sqream␣
↪server
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"sql
↪">
↪sql␣
↪command
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
```

(continues on next page)

(continued from previous page)

```
↪<exception␣
↪cref=
↪"InvalidOperationException
↪">
↪␣
↪thrown␣
↪when␣
↪the␣
↪connection␣
↪is␣
↪not␣
↪open
↪</
↪exception>
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪public␣
↪void␣
↪ExecuteSQLCommand(SqreamConnection␣
↪connection,
↪␣
↪string␣
↪sql)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪validates␣
↪the␣
↪connection␣
```

(continues on next page)

(continued from previous page)

```
↪is␣
↪open␣
↪and␣
↪throws␣
↪exception␣
↪if␣
↪not
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪if␣
↪(connection.
↪State␣
↪!
↪=␣
↪System.
↪Data.
↪ConnectionState.
↪Open)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪throw␣
↪new␣
↪InvalidOperationException(string.
↪Format(
↪"connection␣
↪to␣
↪sqream␣
↪is␣
↪not␣
↪open.
↪ connection.State: {0}", connection.State));

␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪creates␣
↪a␣
↪new␣
↪command␣
↪object␣
↪utilizing␣
↪the␣
↪sql␣
↪and␣
↪the␣
↪connection
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪command␣
↪=␣
↪new␣
↪SqreamCommand(sql,
↪␣
↪connection);
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪/
↪/
↪␣
↪executes␣
↪the␣
↪command
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪command.
↪ExecuteNonQuery();
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪}
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<summary>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
```

(continues on next page)

(continued from previous page)

```
↪/
↪/
↪␣
↪Executes␣
↪a␣
↪SQL␣
↪command␣
↪to␣
↪sqream␣
↪server,
↪␣
↪and␣
↪reads␣
↪the␣
↪result␣
↪set␣
↪usiing␣
↪DataReader
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪</
↪summary>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"connection
↪">
↪connection␣
↪to␣
↪sqream␣
↪server
↪</
↪param>
```

(continues on next page)

(continued from previous page)

```
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<param␣
↪name=
↪"sql
↪">
↪sql␣
↪command
↪</
↪param>
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪/
↪␣
↪
↪<exception␣
↪cref=
↪"InvalidOperationException
↪">
↪␣
↪thrown␣
↪when␣
↪the␣
↪connection␣
↪is␣
↪not␣
↪open
↪</
↪exception>
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪public␣
↪List
↪<Tuple
↪<int,
↪␣
↪string>
↪>
↪␣
↪ReadExampleData(SqreamConnection␣
↪connection,
↪␣
↪string␣
↪sql)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪validates␣
↪the␣
↪connection␣
↪is␣
↪open␣
↪and␣
↪throws␣
↪exception␣
↪if␣
↪not
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪if␣
↪(connection.
↪State␣
↪!
↪=␣
↪System.
↪Data.
↪ConnectionState.
↪Open)
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪throw␣
↪new␣
↪InvalidOperationException(string.
↪Format(
↪"connection␣
↪to␣
↪sqream␣
↪is␣
↪not␣
↪open.
↪ connection.State: {0}", connection.State));

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪creates␣
```

(continues on next page)

(continued from previous page)

```
↪a␣
↪new␣
↪command␣
↪object␣
↪utilizing␣
↪the␣
↪sql␣
↪and␣
↪the␣
↪connection
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪command␣
↪=␣
↪new␣
↪SqreamCommand(sql,
↪␣
↪connection);
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪creates␣
↪a␣
↪reader␣
↪object␣
↪to␣
↪iterate␣
↪over␣
↪the␣
↪result␣
↪set
␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪reader␣
↪=␣
↪(SqreamDataReader)command.
↪ExecuteReader();
↪
82
83 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪list␣
↪of␣
↪results
84 ␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪var␣
↪result␣
↪=␣
↪new␣
↪List
↪<Tuple
↪<int,
↪␣
↪string>
```

(continues on next page)

(continued from previous page)

```
↪>
↪();
↪

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪iterate␣
↪the␣
↪reader␣
↪and␣
↪read␣
↪the␣
↪table␣
↪int,
↪string␣
↪values␣
↪into␣
↪a␣
↪result tuple object
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪while␣
↪(reader.
↪Read())
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪␣
↪␣
↪␣
↪result.
↪Add(new␣
↪Tuple
↪<int,
↪␣
↪string>
↪(reader.
↪GetInt32(0),
↪␣
↪reader.
↪GetString(1)));

␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪/
↪/
↪␣
↪return␣
↪the␣
↪result␣
↪set
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪return␣
↪result;
↪
␣
↪␣
↪␣
↪␣
↪␣
↪␣
```

(continues on next page)

(continued from previous page)

```
↪␣
↪␣
↪}
↪
```

8.2.1.5.4 Limitations

- Unicode characters are not supported when using `INSERT INTO AS SELECT`
- To avoid possible casting issues, use `getDouble` when using `FLOAT`
- The `ARRAY` data types

is not supported. If your database schema includes `ARRAY` columns, you may encounter compatibility issues when using SQreamNET to connect to the database.

Need help?

If you couldn't find what you're looking for, contact SQream Support

Need help?

If you need a tool that SQream does not support,

contact SQream Support or your SQreamDB account manager for more information.

CHAPTER

NINE

DATA INGESTION SOURCES

The **Data Ingestion Sources** page provides information about the following:

9.1 Overview

The **Ingesting Data Overview** page provides basic information useful when ingesting data into SQream from a variety of sources and locations, and describes the following:

- *Getting Started*
- *Data Loading Considerations*
- *Further Reading and Migration Guides*

9.1.1 Getting Started

SQream supports ingesting data using the following methods:

- Executing the `INSERT` statement using a client driver.

- Executing the `COPY FROM` statement or ingesting data from foreign tables:
  - Local filesystem and locally mounted network filesystems
  - Ingesting Data using the Amazon S3 object storage service
  - Ingesting Data using an HDFS data storage system

SQream supports loading files from the following formats:

- Text - CSV, TSV, and PSV
- Parquet
- ORC
- Avro
- JSON

For more information, see the following:

- Using the `INSERT` statement - insert

- Using client drivers - *Client drivers*
- Using the `COPY FROM` statement - copy_from
- Using the Amazon S3 object storage service - *Inserting Data Using Amazon S3*
- Using the HDFS data storage system - *Using SQream in an HDFS Environment*
- Loading data from foreign tables - *Foreign Tables*

9.1.2 Data Loading Considerations

The **Data Loading Considerations** section describes the following:

- *Verifying Data and Performance after Loading*
- *File Soure Location when Loading*
- *Supported Load Methods*
- *Unsupported Data Types*
- *Handing Extended Errors*
- *Best Practices for CSV*
- *Best Practices for Parquet*
- *Best Practices for ORC*

9.1.2.1 Verifying Data and Performance after Loading

Like many RDBMSs, SQream recommends its own set of best practices for table design and query optimization. When using SQream, verify the following:

- That your data is structured as you expect (row counts, data types, formatting, content).
- That your query performance is adequate.
- That you followed the table design best practices (*Optimization and Best Practices*).
- That you've tested and verified that your applications work (such as Tableau).
- That your data types have not been not over-provisioned.

9.1.2.2 File Soure Location when Loading

While you are loading data, you can use the `COPY FROM` command to let statements run on any worker. If you are running multiple nodes, verify that all nodes can see the source the same. Loading data from a local file that is only on one node and not on shared storage may cause it to fail. If required, you can also control which node a statement runs on using the Workload Manager).

For more information, see the following:

- copy_from
- *Workload Manager*

9.1.2.3 Supported Load Methods

You can use the `COPY FROM` syntax to load CSV files.

Note: The `COPY FROM` cannot be used for loading data from Parquet and ORC files.

You can use foreign tables to load text files, Parquet, and ORC files, and to transform your data before generating a full table, as described in the following table:

Method/File Type	Text (CSV)	Parquet	ORC	Streaming Data
COPY FROM	Supported	Not supported	Not supported	Not supported
Foreign tables	Supported	Supported	Supported	Not supported
INSERT	Not supported	Not supported	Not supported	Supported (Python, JDBC, Node.JS)

For more information, see the following:

- COPY FROM
- *Foreign tables*
- INSERT

9.1.2.4 Unsupported Data Types

SQream does not support certain features that are supported by other databases, such as `ARRAY`, `BLOB`, `ENUM`, and `SET`. You must convert these data types before loading them. For example, you can store `ENUM` as `TEXT`.

9.1.2.5 Handing Extended Errors

While you can use foreign tables to load CSVs, the `COPY FROM` statement provides more fine-grained error handling options and extended support for non-standard CSVs with multi-character delimiters, alternate timestamp formats, and more.

For more information, see *foreign tables*.

9.1.2.6 Best Practices for CSV

Text files, such as CSV, rarely conform to RFC 4180 , so you may need to make the following modifications:

- Use `OFFSET 2` for files containing header rows.
- You can capture failed rows in a log file for later analysis, or skip them. See capturing_rejected_rows for information on skipping rejected rows.
- You can modify record delimiters (new lines) using the *RECORD DELIMITER* syntax.
- If the date formats deviate from ISO 8601, refer to the copy_date_parsers section for overriding the default parsing.
- *(Optional)* You can quote fields in a CSV using double-quotes (`"`).

Note: You must quote any field containing a new line or another double-quote character.

- If a field is quoted, you must double quote any double quote, similar to the **string literals quoting rules**. For example, to encode `What are "birds"?`, the field should appear as `"What are ""birds""?"`. For more information, see string literals quoting rules.
- Field delimiters do not have to be a displayable ASCII character. For all supported field delimiters, see field_delimiters.

9.1.2.7 Best Practices for Parquet

The following list shows the best practices when ingesting data from Parquet files:

- You must load Parquet files through *Foreign Tables*. Note that the destination table structure must be identical to the number of columns between the source files.
- Parquet files support **predicate pushdown**. When a query is issued over Parquet files, SQream uses row-group metadata to determine which row-groups in a file must be read for a particular query and the row indexes can narrow the search to a particular set of rows.

9.1.2.7.1 Supported Types and Behavior Notes

Unlike the ORC format, the column types should match the data types exactly, as shown in the table below:

SQream DB type → Parquet source	BOOL	TINYINT	SMALLINT	INT	BIGINT	REAL	DOUBLE	Text [#f0]_	DATE	DATETIME
BOOLEAN	Supported									
INT16			Supported							
INT32				Supported						
INT64					Supported					
FLOAT						Supported				
DOUBLE							Supported			
BYTE_ARRAY[1]								Supported		
INT96[2]										Supported[3]

If a Parquet file has an unsupported type, such as `enum`, `uuid`, `time`, `json`, `bson`, `lists`, `maps`, but the table does not reference this data (i.e., the data does not appear in the SELECT query), the statement will succeed. If the table **does** reference a column, an error will be displayed explaining that the type is not supported, but the column may be omitted.

[1] With UTF8 annotation
[2] With `TIMESTAMP_NANOS` or `TIMESTAMP_MILLIS` annotation
[3] Any microseconds will be rounded down to milliseconds.

9.1.2.8 Best Practices for ORC

The following list shows the best practices when ingesting data from ORC files:

- You must load ORC files through *Foreign Tables*. Note that the destination table structure must be identical to the number of columns between the source files.
- ORC files support **predicate pushdown**. When a query is issued over ORC files, SQream uses ORC metadata to determine which stripes in a file need to be read for a particular query and the row indexes can narrow the search to a particular set of 10,000 rows.

9.1.2.8.1 Type Support and Behavior Notes

You must load ORC files through a foreign table. Note that the destination table structure must be identical to the number of columns between the source files.

For more information, see *Foreign Tables*.

The types should match to some extent within the same "class", as shown in the following table:

SQream DB Type → ORC Source	BOOL	TINYINT	SMALLINT	INT	BIGINT	REAL	DOUBLE	TEXT	DATE	DATETIME
boolean	Supported	Supported[4]	Supported[4]	Supported[4]	Supported[4]					
tinyint	○[5]	Supported	Supported	Supported	Supported					
smallint	○[5]	○[6]	Supported	Supported	Supported					
int	○[5]	○[6]	○[6]	Supported	Supported					
bigint	○[5]	○[6]	○[6]	○[6]	Supported					
float						Supported	Supported			
double						Supported	Supported			
string / char / varchar								Supported		
date									Supported	Supported
timestamp, timestamp with timezone										Supported

- If an ORC file has an unsupported type like `binary`, `list`, `map`, and `union`, but the data is not referenced in the table (it does not appear in the SELECT query), the statement will succeed. If the column is referenced, an error will be thrown to the user, explaining that the type is not supported, but the column may be omitted.

[4] Boolean values are cast to 0, 1
[5] Will succeed if all values are 0, 1
[6] Will succeed if all values fit the destination type

9.1.3 Further Reading and Migration Guides

For more information, see the following:

- copy_from
- insert
- *Foreign Tables*

9.2 Avro

Avro is a well-known data serialization system that relies on schemas. Due to its flexibility as an efficient data storage method, SQream supports the Avro binary data format as an alternative to JSON. Avro files are represented using the **Object Container File** format, in which the Avro schema is encoded alongside binary data. Multiple files loaded in the same transaction are serialized using the same schema. If they are not serialized using the same schema, an error message is displayed. SQream uses the **.avro** extension for ingested Avro files.

- *Foreign Data Wrapper Prerequisites*
- *Making Avro Files Accessible to Workers*
- *Preparing Your Table*
- *Mapping Between SQream and Avro Data Types*
- *Mapping Objects to Rows*
- *Ingesting Data into SQream*
- *Parameters*
- *Best Practices*
- *Additional Examples*

9.2.1 Foreign Data Wrapper Prerequisites

Before proceeding, ensure the following Foreign Data Wrapper (FDW) prerequisites:

- **File Existence:** Verify that the file you are ingesting data from exists at the specified path.
- **Path Accuracy:** Confirm that all path elements are present and correctly spelled. Any inaccuracies may lead to data retrieval issues.
- **Bucket Access Permissions:** Ensure that you have the necessary access permissions to the bucket from which you are ingesting data. Lack of permissions can hinder the data retrieval process.
- **Wildcard Accuracy:** If using wildcards, double-check their spelling and configuration. Misconfigured wildcards may result in unintended data ingestion.

9.2.2 Making Avro Files Accessible to Workers

To give workers access to files, every node must have the same view of the storage being used.

The following apply for Avro files to be accessible to workers:

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream servers have access to the HDFS name node with the correct **user-id**. For more information, see *Using SQream in an HDFS Environment*.
- For S3, ensure network access to the S3 endpoint. For more information, see *Inserting Data Using Amazon S3*.

For more information about restricted worker access, see *Workload Manager*.

9.2.3 Preparing Your Table

You can build your table structure on both local and foreign tables:

- *Creating a Table*
- *Creating a Foreign Table*

9.2.3.1 Creating a Table

Before loading data, you must build the `CREATE TABLE` to correspond with the file structure of the inserted table.

The example in this section is based on the source `nba.avro` table shown below:

Table 1: nba.avro

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	44714	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	44718	235	Marquette	6796117
John Holland	Boston Celtics	30	SG	27	44717	205	Boston University	
R.J. Hunter	Boston Celtics	28	SG	22	44717	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	44722	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	44721	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	44720	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	36708	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	44714	190	Louisville	1824360

The following example shows the correct file structure used to create the `CREATE TABLE` statement based on the **nba.avro** table:

```
CREATE TABLE ext_nba
(

     Name        TEXT(40),
     Team        TEXT(40),
     Number      BIGINT,
     Position    TEXT(2),
     Age         BIGINT,
     Height      TEXT(4),
     Weight      BIGINT,
     College     TEXT(40),
     Salary      FLOAT
 )
 WRAPPER avro_fdw
 OPTIONS
 (
   LOCATION =  's3://sqream-demo-data/nba.avro'
 );
```

Tip: An exact match must exist between the SQream and Avro types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

Note: The **nba.avro** file is stored on S3 at `s3://sqream-demo-data/nba.avro`.

9.2.3.2 Creating a Foreign Table

Before loading data, you must build the `CREATE FOREIGN TABLE` to correspond with the file structure of the inserted table.

The example in this section is based on the source `nba.avro` table shown below:

Table 2: nba.avro

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	44714	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	44718	235	Marquette	6796117
John Holland	Boston Celtics	30	SG	27	44717	205	Boston University	
R.J. Hunter	Boston Celtics	28	SG	22	44717	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	44722	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	44721	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	44720	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	36708	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	44714	190	Louisville	1824360

The following example shows the correct file structure used to create the `CREATE FOREIGN TABLE` statement based on the **nba.avro** table:

```
CREATE FOREIGN TABLE ext_nba
(

     Name        TEXT(40),
     Team        TEXT(40),
     Number      BIGINT,
     Position    TEXT(2),
     Age         BIGINT,
     Height      TEXT(4),
     Weight      BIGINT,
     College     TEXT(40),
     Salary      FLOAT
 )
 WRAPPER avro_fdw
 OPTIONS
 (
   LOCATION =  's3://sqream-demo-data/nba.avro'
 );
```

Tip: An exact match must exist between the SQream and Avro types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

Note: The **nba.avro** file is stored on S3 at `s3://sqream-demo-data/nba.avro`.

Note: The examples in the sections above are identical except for the syntax used to create the tables.

9.2.4 Mapping Between SQream and Avro Data Types

Mapping between SQream and Avro data types depends on the Avro data type:

- *Primitive Data Types*
- *Complex Data Types*
- *Logical Data Types*

9.2.4.1 Primitive Data Types

The following table shows the supported **Primitive** data types:

Avro Type	SQream Type			
	Number	Date/Datetime	String	Boolean
`null`	Supported	Supported	Supported	Supported
`boolean`			Supported	Supported
`int`	Supported		Supported	
`long`	Supported		Supported	
`float`	Supported		Supported	
`double`	Supported		Supported	
`bytes`				
`string`		Supported	Supported	

9.2.4.2 Complex Data Types

The following table shows the supported **Complex** data types:

Avro Type	SQream Type			
	Number	Date/Datetime	String	Boolean
`record`				
`enum`			Supported	
`array`				
`map`				
`union`	Supported	Supported	Supported	Supported
`fixed`				

9.2.4.3 Logical Data Types

The following table shows the supported **Logical** data types:

Avro Type	SQream Type			
	Number	Date/Datetime	String	Boolean
`decimal`	Supported		Supported	
`uuid`			Supported	
`date`		Supported	Supported	
`time-millis`				
`time-micros`				
`timestamp-millis`		Supported	Supported	
`timestamp-micros`		Supported	Supported	
`local-timestamp-millis`				
`local-timestamp-micros`				
`duration`				

Note: Number types include **tinyint**, **smallint**, **int**, **bigint**, **real** and **float**, and **numeric**. String types include **text**.

9.2.5 Mapping Objects to Rows

When mapping objects to rows, each Avro object or message must contain one `record` type object corresponding to a single row in SQream. The `record` fields are associated by name to their target table columns. Additional unmapped fields will be ignored. Note that using the JSONPath option overrides this.

9.2.6 Ingesting Data into SQream

- *Syntax*
- *Example*

9.2.6.1 Syntax

Before ingesting data into SQream from an Avro file, you must create a table using the following syntax:

```
COPY [schema name.]table_name
  FROM WRAPPER fdw_name
;
```

After creating a table you can ingest data from an Avro file into SQream using the following syntax:

```
avro_fdw
```

9.2.6.2 Example

The following is an example of creating a table:

```
COPY t
  FROM WRAPPER fdw_name
  OPTIONS
  (
    [ copy_from_option [, ...] ]
  )
;
```

The following is an example of loading data from an Avro file into SQream:

```
WRAPPER avro_fdw
OPTIONS
(
  LOCATION =  's3://sqream-demo-data/nba.avro'
);
```

For more examples, see *Additional Examples*.

9.2.7 Parameters

The following table shows the Avro parameter:

Parameter	Description
`schema_name`	The schema name for the table. Defaults to `public` if not specified.

9.2.8 Best Practices

Because foreign tables do not automatically verify the file integrity or structure, SQream recommends manually verifying your table output when ingesting Avro files into SQream. This lets you determine if your table output is identical to your originally inserted table.

The following is an example of the output based on the **nba.avro** table:

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name           | Team            | Number | Position | Age | Height | Weight | College ␣
↪          | Salary
---------------+-----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas   ␣
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |         ␣
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |         ␣
↪          | 12000000
```

(continues on next page)

(continued from previous page)

```
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU     ␣
↪         |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga ␣
↪         |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State    |  3431040
```

Note: If your table output has errors, verify that the structure of the Avro files correctly corresponds to the foreign table structure that you created.

9.2.9 Additional Examples

This section includes the following additional examples of loading data into SQream:

- *Omitting Unsupported Column Types*
- *Modifying Data Before Loading*
- *Loading a Table from a Directory of Avro Files on HDFS*
- *Loading a Table from a Directory of Avro Files on S3*

9.2.9.1 Omitting Unsupported Column Types

When loading data, you can omit columns using the `NULL as` argument. You can use this argument to omit unsupported columns from queries that access foreign tables. By omitting them, these columns will not be called and will avoid generating a "type mismatch" error.

In the example below, the `Position` column is not supported due its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;
```

9.2.9.2 Modifying Data Before Loading

One of the main reasons for staging data using the `FOREIGN TABLE` argument is to examine and modify table contents before loading it into SQream.

For example, we can replace pounds with kilograms using the create_table_as statement

In the example below, the `Position` column is set to the default `NULL`.

```
CREATE FOREIGN TABLE nba AS
   SELECT name, team, number, NULL as Position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

9.2.9.3 Loading a Table from a Directory of Avro Files on HDFS

The following is an example of loading a table from a directory of Avro files on HDFS:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER avro_fdw
OPTIONS
  (
     LOCATION =  'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.avro'
  );

CREATE TABLE users AS SELECT * FROM ext_users;
```

For more configuration option examples, navigate to the create_foreign_table page and see the **Parameters** table.

9.2.9.4 Loading a Table from a Directory of Avro Files on S3

The following is an example of loading a table from a directory of Avro files on S3:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER avro_fdw
OPTIONS
  ( LOCATION = 's3://pp-secret-bucket/users/*.avro',
    AWS_ID = 'our_aws_id',
    AWS_SECRET = 'our_aws_secret'
   );

CREATE TABLE users AS SELECT * FROM ext_users;
```

9.3 CSV

This guide covers ingesting data from CSV files into SQream DB using the copy_from method.

- *Foreign Data Wrapper Prerequisites*
- *Prepare CSVs*
- *Place CSVs where SQream DB workers can access*
- *Figure out the table structure*
- *Bulk load the data with COPY FROM*
- *Loading different types of CSV files*

9.3.1 Foreign Data Wrapper Prerequisites

Before proceeding, ensure the following Foreign Data Wrapper (FDW) prerequisites:

- **File Existence:** Verify that the file you are ingesting data from exists at the specified path.
- **Path Accuracy:** Confirm that all path elements are present and correctly spelled. Any inaccuracies may lead to data retrieval issues.
- **Bucket Access Permissions:** Ensure that you have the necessary access permissions to the bucket from which you are ingesting data. Lack of permissions can hinder the data retrieval process.
- **Wildcard Accuracy:** If using wildcards, double-check their spelling and configuration. Misconfigured wildcards may result in unintended data ingestion.

9.3.2 Prepare CSVs

Prepare the source CSVs, with the following requirements:

- Files should be a valid CSV. By default, SQream DB's CSV parser can handle RFC 4180 standard CSVs , but can also be modified to support non-standard CSVs (with multi-character delimiters, unquoted fields, etc).
- Files are UTF-8 or ASCII encoded
- Field delimiter is an ASCII character or characters
- Record delimiter, also known as a new line separator, is a Unix-style newline (`\n`), DOS-style newline (`\r\n`), or Mac style newline (`\r`).
- Fields are optionally enclosed by double-quotes, or mandatory quoted if they contain one of the following characters:
  - The record delimiter or field delimiter
  - A double quote character
  - A newline
- If a field is quoted, any double quote that appears must be double-quoted (similar to the string literals quoting rules. For example, to encode `What are "birds"?`, the field should appear as `"What are ""birds""?"`.

  Other modes of escaping are not supported (e.g. `1,"What are \"birds\"?"` is not a valid way of escaping CSV values).
- `NULL` values can be marked in two ways in the CSV:
  - An explicit null marker. For example, `col1,\N,col3`
  - An empty field delimited by the field delimiter. For example, `col1,,col3`

Note: If a text field is quoted but contains no content (`""`) it is considered an empty text field. It is not considered `NULL`.

9.3.3 Place CSVs where SQream DB workers can access

During data load, the copy_from command can run on any worker (unless explicitly speficied with the *Workload Manager*). It is important that every node has the same view of the storage being used - meaning, every SQream DB worker should have access to the files.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream DB servers can access the HDFS name node with the correct user-id. See our *Using SQream in an HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Inserting Data Using Amazon S3* guide for more information.

9.3.4 Figure out the table structure

Prior to loading data, you will need to write out the table structure, so that it matches the file structure.

For example, to import the data from `nba.csv`, we will first look at the file:

Table 3: nba.csv

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	44714	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	44718	235	Marquette	6796117
John Holland	Boston Celtics	30	SG	27	44717	205	Boston University	
R.J. Hunter	Boston Celtics	28	SG	22	44717	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	44722	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	44721	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	44720	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	36708	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	44714	190	Louisville	1824360

- The file format in this case is CSV, and it is stored as an S3 object.
- The first row of the file is a header containing column names.
- The record delimiter was a DOS newline (`\r\n`).
- The file is stored on S3, at `s3://sqream-demo-data/nba.csv`.

We will make note of the file structure to create a matching `CREATE TABLE` statement.

```
CREATE TABLE nba
(
   Name text(40),
   Team text(40),
```

(continues on next page)

(continued from previous page)

```
   Number tinyint,
   Position text(2),
   Age tinyint,
   Height text(4),
   Weight real,
   College text(40),
   Salary float
 );
```

9.3.5 Bulk load the data with COPY FROM

The CSV is a standard CSV, but with two differences from SQream DB defaults:

- The record delimiter is not a Unix newline (`\n`), but a Windows newline (`\r\n`)
- The first row of the file is a header containing column names, which we'll want to skip.

```
COPY nba
   FROM 's3://sqream-demo-data/nba.csv'
   WITH RECORD DELIMITER '\r\n'
        OFFSET 2;
```

Repeat steps 3 and 4 for every CSV file you want to import.

9.3.6 Loading different types of CSV files

copy_from contains several configuration options. See more in the COPY FROM elements section.

9.3.6.1 Loading a standard CSV file from a local filesystem

```
COPY table_name FROM '/home/rhendricks/file.csv';
```

9.3.6.2 Loading a PSV (pipe separated value) file

```
COPY table_name FROM '/home/rhendricks/file.psv' WITH DELIMITER '|';
```

9.3.6.3 Loading a TSV (tab separated value) file

```
COPY table_name FROM '/home/rhendricks/file.tsv' WITH DELIMITER '\t';
```

9.3.6.4 Loading a text file with non-printable delimiter

In the file below, the separator is `DC1`, which is represented by ASCII 17 decimal or 021 octal.

```
COPY table_name FROM 'file.txt' WITH DELIMITER E'\021';
```

9.3.6.5 Loading a text file with multi-character delimiters

In the file below, the separator is `'|`.

```
COPY table_name FROM 'file.txt' WITH DELIMITER '''|';
```

9.3.6.6 Loading files with a header row

Use `OFFSET` to skip rows.

Note: When loading multiple files (e.g. with wildcards), this setting affects each file separately.

```
COPY  table_name FROM 'filename.psv' WITH DELIMITER '|' OFFSET  2;
```

9.3.6.7 Loading files formatted for Windows (`\r\n`)

```
COPY table_name FROM 'filename.psv' WITH DELIMITER '|' RECORD DELIMITER '\r\n';
```

9.3.6.8 Loading a file from a public S3 bucket

Note: The bucket must be publicly available and objects can be listed

```
COPY nba FROM 's3://sqream-demo-data/nba.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n';
```

9.3.6.9 Loading files from an authenticated S3 bucket

```
COPY nba FROM 's3://secret-bucket/*.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n' AWS_ID
↪'12345678' AWS_SECRET 'super_secretive_secret';
```

9.3.6.10 Loading files from an HDFS storage

```
COPY nba FROM 'hdfs://hadoop-nn.piedpiper.com/rhendricks/*.csv' WITH OFFSET 2 RECORD␣
↪DELIMITER '\r\n';
```

9.3.6.11 Saving rejected rows to a file

See capturing_rejected_rows for more information about the error handling capabilities of `COPY FROM`.

```
COPY table_name FROM WRAPPER csv_fdw OPTIONS (location = '/tmp/file.psv'
        ,delimiter = '|'
                        ,continue_on_error = True
        ,error_log = '/temp/load_error.log' -- Save error log
        ,rejected_data = '/temp/load_rejected.log' -- Only save rejected rows
        );
```

9.3.6.12 Stopping the load if a certain amount of rows were rejected

```
COPY  table_name  FROM  'filename.csv'   WITH  delimiter  '|'
             ERROR_LOG  '/temp/load_err.log' -- Save error log
             OFFSET 2 -- skip header row
             LIMIT  100 -- Only load 100 rows
             STOP AFTER 5 ERRORS; -- Stop the load if 5 errors reached
```

9.3.6.13 Load CSV files from a set of directories

Use glob patterns (wildcards) to load multiple files to one table.

```
COPY table_name  from  '/path/to/files/2019_08_*/*.csv';
```

9.3.6.14 Rearrange destination columns

When the source of the files does not match the table structure, tell the `COPY` command what the order of columns should be

```
COPY table_name (fifth, first, third) FROM '/path/to/files/*.csv';
```

Note: Any column not specified will revert to its default value or `NULL` value if nullable

9.3.6.15 Loading non-standard dates

If files contain dates not formatted as `ISO8601`, tell `COPY` how to parse the column. After parsing, the date will appear as `ISO8601` inside SQream DB.

In this example, `date_col1` and `date_col2` in the table are non-standard. `date_col3` is mentioned explicitly, but can be left out. Any column that is not specified is assumed to be `ISO8601`.

```
COPY table_name FROM '/path/to/files/*.csv' WITH PARSERS 'date_col1=YMD,date_col2=MDY,
↪date_col3=default';
```

Tip: The full list of supported date formats can be found under the Supported date formats section of the copy_from reference.

9.4 Parquet

Ingesting Parquet files into SQream is generally useful when you want to store the data permanently and perform frequent queries on it. Ingesting the data can also make it easier to join with other tables in your database. However, if you wish to retain your data on external Parquet files instead of ingesting it into SQream due to it being an open-source column-oriented data storage format, you may also execute *FOREIGN TABLE* queries.

- *Foreign Data Wrapper Prerequisites*
- *Preparing Your Parquet Files*
- *Making Parquet Files Accessible to Workers*
- *Creating a Table*
- *Ingesting Data into SQream*
- *Best Practices*

9.4.1 Foreign Data Wrapper Prerequisites

Before proceeding, ensure the following Foreign Data Wrapper (FDW) prerequisites:

- **File Existence:** Verify that the file you are ingesting data from exists at the specified path.
- **Path Accuracy:** Confirm that all path elements are present and correctly spelled. Any inaccuracies may lead to data retrieval issues.
- **Bucket Access Permissions:** Ensure that you have the necessary access permissions to the bucket from which you are ingesting data. Lack of permissions can hinder the data retrieval process.
- **Wildcard Accuracy:** If using wildcards, double-check their spelling and configuration. Misconfigured wildcards may result in unintended data ingestion.

9.4.2 Preparing Your Parquet Files

Prepare your source Parquet files according to the requirements described in the following table:

SQream Type → Parquet Source ↓	BOOLEAN	TINYINT	SMALLINT	INT	BIGINT	REAL	DOUBLE	TEXT[1]	VARCHAR	DATE	DATETIME
BOOLEAN	Supported										
INT16			Supported								
INT32				Supported							
INT64					Supported						
FLOAT						Supported					
DOUBLE							Supported				
BYTE_ARRAY / FIXED_LEN_BYTE_ARRAY[2]								Supported			
INT96[3]											Supported[4]

Your statements will succeed even if your Parquet file contains unsupported types, such as `enum`, `uuid`, `time`, `json`, `bson`, `lists`, `maps`, but the data is not referenced in the table (it does not appear in the SELECT query). If the column containing the unsupported type is referenced, an error message is displayed explaining that the type is not supported and that the column may be ommitted. For solutions to this error message, see more information in **Managing Unsupported Column Types** example in the **Example** section.

9.4.3 Making Parquet Files Accessible to Workers

To give workers access to files, every node must have the same view of the storage being used.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream servers have access to the HDFS name node with the correct user-id. For more information, see *Using SQream in an HDFS Environment* guide.
- For S3, ensure network access to the S3 endpoint. For more information, see *Inserting Data Using Amazon S3* guide.

[1] Text values include `TEXT`
[2] With UTF8 annotation
[3] With `TIMESTAMP_NANOS` or `TIMESTAMP_MILLIS` annotation
[4] Any microseconds will be rounded down to milliseconds.

9.4.4 Creating a Table

Before loading data, you must create a table that corresponds to the file structure of the table you wish to insert.

The example in this section is based on the source nba.parquet table shown below:

Table 4: nba.parquet

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	44714	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	44718	235	Marquette	6796117
John Holland	Boston Celtics	30	SG	27	44717	205	Boston University	
R.J. Hunter	Boston Celtics	28	SG	22	44717	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	44722	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	44721	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	44720	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	36708	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	44714	190	Louisville	1824360

The following example shows the correct file structure used for creating a *FOREIGN TABLE* based on the nba.parquet table:

```
CREATE FOREIGN TABLE ext_nba
(
     Name        TEXT(40),
     Team        TEXT(40),
     Number      BIGINT,
     Position    TEXT(2),
     Age         BIGINT,
     Height      TEXT(4),
     Weight      BIGINT,
     College     TEXT(40),
     Salary      FLOAT
 )
 WRAPPER parquet_fdw
 OPTIONS
 (
   LOCATION =  's3://sqream-demo-data/nba.parquet'
 );
```

Tip: An exact match must exist between the SQream and Parquet types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

Note: The **nba.parquet** file is stored on S3 at `s3://sqream-demo-data/nba.parquet`.

9.4.5 Ingesting Data into SQream

9.4.5.1 Syntax

You can use the create_table_as statement to load the data into SQream, as shown below:

```
CREATE TABLE nba AS
   SELECT * FROM ext_nba;
```

9.4.5.2 Examples

- *Omitting Unsupported Column Types*
- *Modifying Data Before Loading*
- *Loading a Table from a Directory of Parquet Files on HDFS*
- *Loading a Table from a Directory of Parquet Files on S3*

9.4.5.2.1 Omitting Unsupported Column Types

When loading data, you can omit columns using the NULL as argument. You can use this argument to omit unsupported columns from queries that access external tables. By omitting them, these columns will not be called and will avoid generating a "type mismatch" error.

In the example below, the `Position column` is not supported due its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;
```

9.4.5.2.2 Modifying Data Before Loading

One of the main reasons for staging data using the `EXTERNAL TABLE` argument is to examine and modify table contents before loading it into SQream.

For example, we can replace **pounds** with **kilograms** using the `CREATE TABLE AS` statement.

In the example below, the `Position column` is set to the default `NULL`.

```
CREATE TABLE nba AS
   SELECT name, team, number, NULL as position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

9.4.5.2.3 Loading a Table from a Directory of Parquet Files on HDFS

The following is an example of loading a table from a directory of Parquet files on HDFS:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER parquet_fdw
OPTIONS
  (
     LOCATION =  'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.parquet'
  );

CREATE TABLE users AS SELECT * FROM ext_users;
```

9.4.5.2.4 Loading a Table from a Directory of Parquet Files on S3

The following is an example of loading a table from a directory of Parquet files on S3:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER parquet_fdw
OPTIONS
  ( LOCATION = 's3://pp-secret-bucket/users/*.parquet',
    AWS_ID = 'our_aws_id',
    AWS_SECRET = 'our_aws_secret'
   );

CREATE TABLE users AS SELECT * FROM ext_users;
```

For more configuration option examples, navigate to the create_foreign_table page and see the **Parameters** table.

9.4.6 Best Practices

Because external tables do not automatically verify the file integrity or structure, SQream recommends manually verifying your table output when ingesting Parquet files into SQream. This lets you determine if your table output is identical to your originally inserted table.

The following is an example of the output based on the **nba.parquet** table:

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name           | Team           | Number | Position | Age | Height | Weight | College ␣
↪          | Salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas   ␣
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |         ␣
↪          |  5000000
```

(continues on next page)

(continued from previous page)

```
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |         ␣
↪         | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU     ␣
↪         |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga ␣
↪         |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State    |  3431040
```

Note: If your table output has errors, verify that the structure of the Parquet files correctly corresponds to the external table structure that you created.

9.5 ORC

This guide covers ingesting data from ORC files into SQream DB using FOREIGN TABLE.

- *Foreign Data Wrapper Prerequisites*
- *Prepare the files*
- *Place ORC files where SQream DB workers can access them*
- *Figure out the table structure*
- *Verify table contents*
- *Copying data into SQream DB*
- *Further ORC loading examples*

9.5.1 Foreign Data Wrapper Prerequisites

Before proceeding, ensure the following Foreign Data Wrapper (FDW) prerequisites:

- **File Existence:** Verify that the file you are ingesting data from exists at the specified path.
- **Path Accuracy:** Confirm that all path elements are present and correctly spelled. Any inaccuracies may lead to data retrieval issues.
- **Bucket Access Permissions:** Ensure that you have the necessary access permissions to the bucket from which you are ingesting data. Lack of permissions can hinder the data retrieval process.
- **Wildcard Accuracy:** If using wildcards, double-check their spelling and configuration. Misconfigured wildcards may result in unintended data ingestion.

9.5.2 Prepare the files

Prepare the source ORC files, with the following requirements:

SQreamDB type → ORC source	TINYINT	SMALLINT	INT	BIGINT	REAL	DOUBLE	NVARCHAR	DATE	DATETIME
boolean	Supported[2]	Supported[2]	Supported[2]	Supported[2]					
tinyint	Supported	Supported	Supported	Supported					
smallint	○[3] ○[4]	Supported	Supported	Supported					
int	○[3] ○[4]	○[4]	Supported	Supported					
bigint	○[3] ○[4]	○[4]	○[4]	Supported					
float					Supported	Supported			
double					Supported	Supported			
string / char / text							Supported		
date								Supported	Supported
timestamp, timestamp with timezone									Supported

- If an ORC file has an unsupported type like `binary`, `list`, `map`, and `union`, but the data is not referenced in the table (it does not appear in the SELECT query), the statement will succeed. If the column is referenced, an error will be thrown to the user, explaining that the type is not supported, but the column may be ommited. This can be worked around. See more information in the examples.

[1] Text values include `TEXT`
[2] Boolean values are cast to 0, 1
[3] Will succeed if all values are 0, 1
[4] Will succeed if all values fit the destination type

9.5.3 Place ORC files where SQream DB workers can access them

Any worker may try to access files (unless explicitly speficied with the *Workload Manager*). It is important that every node has the same view of the storage being used - meaning, every SQream DB worker should have access to the files.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream DB servers can access the HDFS name node with the correct user-id. See our *Using SQream in an HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Inserting Data Using Amazon S3* guide for more information.

9.5.4 Figure out the table structure

Prior to loading data, you will need to write out the table structure, so that it matches the file structure.

For example, to import the data from `nba.orc`, we will first look at the source table:

Table 5: nba.orc

Name	Team	Number	Position	Age	Height	Weight	College	Salary
Avery Bradley	Boston Celtics	0	PG	25	44714	180	Texas	7730337
Jae Crowder	Boston Celtics	99	SF	25	44718	235	Marquette	6796117
John Holland	Boston Celtics	30	SG	27	44717	205	Boston University	
R.J. Hunter	Boston Celtics	28	SG	22	44717	185	Georgia State	1148640
Jonas Jerebko	Boston Celtics	8	PF	29	44722	231		5000000
Amir Johnson	Boston Celtics	90	PF	29	44721	240		12000000
Jordan Mickey	Boston Celtics	55	PF	21	44720	235	LSU	1170960
Kelly Olynyk	Boston Celtics	41	C	25	36708	238	Gonzaga	2165160
Terry Rozier	Boston Celtics	12	PG	22	44714	190	Louisville	1824360

- The file is stored on S3, at `s3://sqream-demo-data/nba.orc`.

We will make note of the file structure to create a matching `CREATE FOREIGN TABLE` statement.

```
CREATE FOREIGN TABLE ext_nba
(
     Name        TEXT(40),
     Team        TEXT(40),
     Number      BIGINT,
     Position    TEXT(2),
     Age         BIGINT,
     Height      TEXT(4),
     Weight      BIGINT,
     College     TEXT(40),
```

(continues on next page)

(continued from previous page)

```
    Salary     FLOAT
)
  WRAPPER orc_fdw
  OPTIONS
    (
       LOCATION = 's3://sqream-demo-data/nba.orc'
    );
```

Tip: Types in SQream DB must match ORC types according to the table above.

If the column type isn't supported, a possible workaround is to set it to any arbitrary type and then exclude it from subsequent queries.

9.5.5 Verify table contents

External tables do not verify file integrity or structure, so verify that the table definition matches up and contains the correct data.

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name           | Team            | Number | Position | Age | Height | Weight | College ␣
↪          | Salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas   ␣
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |         ␣
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |         ␣
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU     ␣
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga ␣
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State    |  3431040
```

If any errors show up at this stage, verify the structure of the ORC files and match them to the external table structure you created.

9.5.6 Copying data into SQream DB

To load the data into SQream DB, use the create_table_as statement:

```
CREATE TABLE nba AS
   SELECT * FROM ext_nba;
```

9.5.6.1 Working around unsupported column types

Suppose you only want to load some of the columns - for example, if one of the columns isn't supported.

By ommitting unsupported columns from queries that access the `EXTERNAL TABLE`, they will never be called, and will not cause a "type mismatch" error.

For this example, assume that the `Position` column isn't supported because of its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;

-- We ommitted the unsupported column `Position` from this query, and replaced it␣
↪with a default ``NULL`` value, to maintain the same table structure.
```

9.5.6.2 Modifying data during the copy process

One of the main reasons for staging data with `EXTERNAL TABLE` is to examine the contents and modify them before loading them.

Assume we are unhappy with weight being in pounds, because we want to use kilograms instead. We can apply the transformation as part of the create_table_as statement.

Similar to the previous example, we will also set the `Position` column as a default `NULL`.

```
CREATE TABLE nba AS
   SELECT name, team, number, NULL as position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

9.5.7 Further ORC loading examples

create_foreign_table contains several configuration options. See more in the CREATE FOREIGN TABLE parameters section.

9.5.7.1 Loading a table from a directory of ORC files on HDFS

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER orc_fdw
  OPTIONS
    (
      LOCATION = 'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.ORC'
    );

CREATE TABLE users AS SELECT * FROM ext_users;
```

9.5.7.2 Loading a table from a bucket of files on S3

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER orc_fdw
OPTIONS
  (  LOCATION = 's3://pp-secret-bucket/users/*.ORC',
     AWS_ID = 'our_aws_id',
     AWS_SECRET = 'our_aws_secret'
   )
;

CREATE TABLE users AS SELECT * FROM ext_users;
```

9.6 JSON

JSON (Java Script Object Notation) is used both as a file format and as a serialization method. The JSON file format is flexible and is commonly used for dynamic, nested, and semi-structured data representations.

The SQream DB JSON parser supports the RFC 8259 data interchange format and supports both JSON objects and JSON object arrays.

Only the JSON Lines data format is supported by SQream.

- *Foreign Data Wrapper Prerequisites*
- *Making JSON Files Accessible to Workers*
- *Mapping between JSON and SQream*
- *Ingesting JSON Data into SQream*

9.6.1 Foreign Data Wrapper Prerequisites

Before proceeding, ensure the following Foreign Data Wrapper (FDW) prerequisites:

- **File Existence:** Verify that the file you are ingesting data from exists at the specified path.
- **Path Accuracy:** Confirm that all path elements are present and correctly spelled. Any inaccuracies may lead to data retrieval issues.
- **Bucket Access Permissions:** Ensure that you have the necessary access permissions to the bucket from which you are ingesting data. Lack of permissions can hinder the data retrieval process.
- **Wildcard Accuracy:** If using wildcards, double-check their spelling and configuration. Misconfigured wildcards may result in unintended data ingestion.

9.6.2 Making JSON Files Accessible to Workers

To give workers access to files, every node in your system must have access to the storage being used.

The following are required for JSON files to be accessible to workers:

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream servers have access to the HDFS NameNode with the correct **user-id**. For more information, see *Using SQream in an HDFS Environment*.
- For S3, ensure network access to the S3 endpoint. For more information, see *Inserting Data Using Amazon S3*.

For more information about configuring worker access, see *Workload Manager*.

9.6.3 Mapping between JSON and SQream

A JSON field consists of a key name and a value.

Key names, which are case sensitive, are mapped to SQream columns. Key names which do not have corresponding SQream table columns are treated as errors by default, unless the `IGNORE_EXTRA_FIELDS` parameter is set to `true`, in which case these key names will be ignored during the mapping process.

SQream table columns which do not have corresponding JSON fields are automatically set to `null` as a value.

Values may be one of the following reserved words (lower-case): `false`, `true`, or `null`, or any of the following data types:

JSON Data Type	Representation in SQream
Number	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`, `FLOAT`, `DOUBLE`, `NUMERIC`
String	`TEXT`
JSON Literal	`NULL`, `TRUE`, `FALSE`
JSON Array	`TEXT`
JSON Object	`TEXT`

9.6.3.1 Character Escaping

The ASCII 10 character (LF) marks the end of JSON objects. Use `\\n` to escape the `\n` character when you do not mean it be a new line.

9.6.4 Ingesting JSON Data into SQream

In this topic:

- *Syntax*
- *Parameters*
- *Automatic Schema Inference*
- *Examples*

9.6.4.1 Syntax

To access JSON files, use the `json_fdw` with a `COPY FROM`, `COPY TO`, or `CREATE FOREIGN TABLE` statement.

The Foreign Data Wrapper (FDW) syntax is:

```
json_fdw [OPTIONS(option=value[,...])]
```

9.6.4.2 Parameters

The following parameters are supported by `json_fdw`:

Parameter	Description
DATETIME_FORMAT	Default format is yyyy-mm-dd. Other supported date formats are:iso8601, iso8601c, dmy, ymd, mdy, yyyymmdd, yyyy-m-d, yyyy-mm-dd, yyyy/m/d, yyyy/mm/dd, d/m/yyyy, dd/mm/yyyy, mm/dd/yyyy, dd-mon-yyyy, yyyy-mon-dd.
IGNORE_EXTRA_FIELDS	Default value is false. When value is true, key names which do not have corresponding SQream table columns will be ignored. Parameter may be used with the COPY TO and IGNORE FOREIGN TABLE statements.
COMPRESSION	Supported values are auto, gzip, and none. auto means that the compression type is automatically detected upon import. Parameter is not supported for exporting. gzip means that a gzip compression is applied. none means that no compression or an attempt to decompress will take place.
LOCATION	A path on the local filesystem, on S3, or on HDFS URI. The local path must be an absolute path that SQream DB can access.
LIMIT	When specified, tells SQream DB to stop ingesting after the specified number of rows. Unlimited if unset.
OFFSET	The row number from which to start ingesting.
ERROR_LOG	If when using the COPY command, copying a row fails, the ERROR LOG command writes error information to the error log specified in the ERROR LOG command. • If an existing file path is specified, the file will be overwritten. • Specifying the same file for ERROR_LOG and REJECTED_DATA is not allowed and will result in error. • Specifying an error log when creating a foreign table will write a new error log for every query on the foreign table.
CONTINUE_ON_ERROR	Specifies if errors should be ignored or skipped. When set to true, the transaction will continue despite rejected data. This parameter should be set together with ERROR_COUNT. When reading multiple files, if an entire file cannot be opened, it will be skipped.
ERROR_COUNT	Specifies the maximum number of faulty records that will be ignored. This setting must be used in conjunction with continue_on_error.
MAX_FILE_SIZE	Sets the maximum file size (bytes).
ENFORCE_SINGLE_FILE	Permitted values are true or false. When set to true, a single file of unlimited size is created. This single file is not limited by the MAX_FILE_SIZE parameter. false permits creating several files together limited by the MAX_FILE_SIZE parameter. Default value: false.
AWS_ID, AWS_SECRET	Specifies the authentication details for secured S3 buckets.

9.6.4.3 Automatic Schema Inference

You may let SQream DB automatically infer the schema of a foreign table when using `json_fdw`.

For more information, follow the *Automatic Foreign Table DDL Resolution* page.

Automatic Schema Inference example:

```
CREATE FOREIGN TABLE t
  WRAPPER json_fdw
  OPTIONS
  (
    location = 'somefile.json'
  )
;
```

9.6.4.4 Examples

JSON objects:

```
[
{ "name":"Avery Bradley", "age":25, "position":"PG" },
{ "name":"Jae Crowder", "age":25, "position":"SF" },
{ "name":"John Holland", "age":27, "position":"SG" }
]
```

Using the `COPY FROM` statement:

```
COPY t
  FROM WRAPPER json_fdw
  OPTIONS
  (
    location = 'somefile.json'
  )
;
```

Note that JSON files generated using the `COPY TO` statement will store objects, and not object arrays.

```
COPY t
  TO WRAPPER json_fdw
  OPTIONS
  (
    location = 'somefile.json'
  )
;
```

When using the `CREATE FOREIGN TABLE` statement, make sure that the table schema corresponds with the JSON file structure.

```
CREATE FOREIGN TABLE t
      (
        id int not null
      )
  WRAPPER json_fdw
  OPTIONS
  (
    location = 'somefile.json'
```

(continues on next page)

(continued from previous page)

```
  )
;
```

The following is an example of loading data from a JSON file into SQream:

```
WRAPPER json_fdw
OPTIONS
(
  LOCATION =  'somefile.json'
);
```

Tip: An exact match must exist between the SQream and JSON types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

9.7 External Databases

The **SQLoader** is a CLI program that enables you to load data into SQreamDB from other DBMS and DBaaS.

SQLoader supports Oracle, Postgresql, Teradata, Microsoft SQL Server, and SAP HANA.

- *Before You Begin*
- *Installation and Connection*
- *Loading Data into SQreamDB*
- *Creating Summary Tables*
- *CLI Examples*

9.7.1 Before You Begin

It is essential that you have the following:

- Java 17
- SQLoader configuration files
- SQLoader.jar file

9.7.1.1 Minimum Hardware Requirements

Component	Type
CPU cores	16
RAM	32GB

9.7.1.2 Sizing Guidelines

The SQLoader sizing is determined by the number of concurrent tables and threads based on the available CPU cores, limiting it to the number of cores minus one, with the remaining core reserved for the operating system. Each SQLoader instance runs on a single table, meaning concurrent imports of multiple tables require multiple instances. Additionally, when dealing with partitioned tables, each partition consumes a thread, so users should consider the table's partition count when managing thread allocation for efficient performance.

9.7.2 Installation and Connection

9.7.2.1 Getting the SQLoader Configuration and JAR Files

1. Download the `.tar` file using the following command:

```
curl -O https://sq-ftp-public.s3.amazonaws.com/sqloader-7.8.tar
```

2. Extract the `.tar` file using the following command:

```
tar -xf sqloader-7.8.tar.gz
```

A folder named `sqloader` with the following files is created:

File	Description
`sqream-mapping.json`	Maps foreign DBMS and DBaaS data types into SQreamDB data types during ingestion
`sqload-jdbc.properties`	Used for defining a connection string and may also be used to reconfigure data loading
`reserved_words.txt`	A list of reserved words which cannot be used as table and/or column names.
`sqloader.jar`	The SQLoader package file

9.7.2.2 Establishing a Connection

The `sqload-jdbc.properties` file contains a connection string that must be configured to enable data loading into SQreamDB.

1. Open the `sqload-jdbc.properties` file.
2. Configure connection parameters for:
   a. Either Postgresql, Oracle, Teradata, Microsoft SQL Server, SAP HANA or SQreamDB connection strings
   b. Optionally, Oracle or SQreamDB catalogs (recommended)

Table 6: Connection String Parameters

Parameter	Description
`HostIp:port`	The host and IP address number
`database_name`	The name of the database from which data is loaded
`user`	Username of a role to use for connection
`password`	Specifies the password of the selected role
`ssl`	Specifies SSL for this connection

Listing 1: Properties File Sample

```
####mandatory!

##Postgresql, Oracle, Teradata, SAP HANA, Microsoft SQL Server, and SQreamDB Connection Strings

#postgres
source.jdbc.connectionstring=jdbc:postgresql://<HostIp:port>/<database_name>?user=<user_name>&password=<password>&ssl=<true/false>

#oracle
source.jdbc.connectionstring=jdbc:oracle:thin:@//<HostIp:port>/<database_name>?user=<user_name>&password=<password>&ssl=<true/false>

#teradata
source.jdbc.connectionstring=jdbc:teradata://<HostIp>/DATABASE=<database_name>,DBS_PORT=<port>,user=<user_name>,password=<password>

#sap hana
source.jdbc.connectionstring=jdbc:sap://<HostIp>:<port>/?user=<user_name>&password=<password>

#microsoft sql server
source.jdbc.connectionstring=jdbc:sqlserver://<HostIp>:<port>;databaseName=<database_name>;user=<user_name>;password=<password>;encrypt=<true/false>;trustServerCertificate=<true/false>

#sqream
sqream.jdbc.connectionstring=jdbc:Sqream://<HostIp:port>/<database_name>;cluster=<true/false>;user=<user_name>;password=<password>

##CDC and Incremental Parameter

cdc_catalog_table=public.CDC_TABLES
cdc_tracking_table=public.CDC_TRACKING
cdc_primary_keys_table=public.CDC_TABLE_PRIMARY_KEYS



####Optional!

#Oracle Catalog
#catalog.jdbc.connectionstring=jdbc:oracle:thin:@//<HostIp:port>/<database_name>?user=<user_name>&password=<password>

#SQreamDB Catalog
#catalog.jdbc.connectionstring=jdbc:Sqream://<HostIp:port>/<database_name>;cluster=<true/false>;user=<user_name>;password=<password>

#Summary Parameter
#load_summary_table=public.SQLOAD_SUMMARY
```

9.7.2.3 Starting SQLoader

To start SQLoader, run the `sqloader.jar` file:

```
java -jar sqloader.jar
```

9.7.3 Loading Data into SQreamDB

To load data into SQreamDB using SQLoader, you must specify a source table name at minimum. Executing the command below will generate a SQreamDB table using the specified source table's DDL and load all its data:

```
java -jar sqloader.jar -table <source_table_name>
```

- *Customizing Data Load*
- *Data Types and Mapping*

9.7.3.1 Customizing Data Load

While specifying a source table name is mandatory for data loading, you have the option to customize the loading process by utilizing *optional* parameters. These parameters can be configured either within a CLI command or by adjusting settings in the properties file.

Please note that any customization done through the CLI will override configurations made using the properties file.

Table 7: SQLoader CLI Parameters

CLI Parameter	State	Default	Type	Description
`-batchsize`	Optional	`10.000`		The number of records to be inserted into SQreamDB at once. Please note that the configured batch size may impact chunk sizes
`-casesensative`	Optional	`false`		If `true`, keeps table name uppercase and lowercase characters when table is created in SQreamDB
`-check_cdc_chain`	Optional	`false`		Check CDC chain between tracking table and source table
`-chunkSize`	Optional	`0`		The number of records read at once from the source database

continues on next page

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-columnlist`	Optional	*None*	`.txt`	The name of the file that contains all column names. Columns must be separated using `\n`
`-columns`	Optional	All columns		The name or names of columns to be loaded into SQreamDB ("col1,col2, …"). For column names containing upper-case characters, maintain the upper-case format, avoid using double quotes or apostrophes, and ensure that the `caseSensitive` parameter is set to true
`-config`	Optional	`/home/username/downloads/config/sqload-jdbc.properties`		Defines the path to the configuration file you wish to use. This parameter may be defined using only the CLI
`-config_dir`	Optional	`/home/username/downloads/config`		Defines the path to the folder containing both the data type mapping and the reserved words files. The defined folder must contain both files or else you will receive an error
`-count`	Optional	`true`		Defines whether or not table rows will be counted before being loaded into SQreamDB
`-delete`	Optional	`true`		Defines whether or not loading using Change Data Capture (CDC) includes deleted rows

continues on next page

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-drop`	Optional	`true`		Assuming we're loading into table `x`: • `true` essentially allows for the replacement of the existing table with the newly loaded data by triggering the following actions: – Creating or replacing a temporary table named `x_temp` – The data you intend to load is inserted into the `x_temp` table – If a table named `x` already exists, it will be renamed to `x_old`. (If `x` exists as a view, the view will be dropped) – The `x_temp` table, which now contains your loaded data, is renamed to `x` – Any previously existing

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-fetchsize`	Optional	`100000`		The number of records to be read at once from source database
`-filter`	Optional	`1=1`		Defines whether or not only records with SQL conditions are loaded
`-h, --help`	Optional	*No input*		Displays the help menu and exits
`-limit`	Optional	`0` (no limit)		Limits the number of rows to be loaded
`-load_dttm`	Optional	`true`		Add an additional `load_dttm` column that defines the time and date of loading
`-lock_check`	Optional	`true`		Defines whether or not SQLoader will check source table is locked before the loading starts
`-lock_table`	Optional	`true`		Defines whether or not SQLoader will lock target table before the loading starts
`-log_dir`	Optional	`logs`		Defines the path of log directory created when loading data. If no value is specified, a `logs` folder is created under the same location as the `sqloader.jar` file

continues on next page

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-partition`	Optional	*None*	Partition identifier `string`	Specifies the number of table partitions. If configured, `-partition` ensures that data is loaded according to the specified partition. You may configure the `-thread` parameter for parallel loading of your table partitions. If you do, please ensure that the number of threads does not exceed the number of partitions
`-rowid`	Optional	`false`		Defines whether or not SQLoader will get row IDs from Oracle tables
`-source_db`	Optional	`ORCL`		Defines the source database name. It does not modify the database connection string but impacts the storage and retrieval of data within catalog tables
`-split`	Optional	*None*	Column name `string`	Column name for split (required for multi-thread loads)
`-table`	Mandatory	*None*	Table name `string`	Source table name to load data from
`-target`	Optional	Target table name	Table name `string`	Target table name to load data into

continues on next page

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-thread`	Optional	`1`		Number of threads to use for loading. Using multiple threads can significantly improve the loading performance, especially when dealing with columns that have metadata statistics (e.g., min/max values). SQLoader will automatically divide the data into batches based on the specified thread number, allowing for parallel processing. You may use `-thread` both for tables that are partitioned and tables that are not. See *Sizing Guidelines*
`-truncate`	Optional	`false`		Truncate target table before loading

continues on next page

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-type`	Optional	`full`		Defines a loading type that affects the table that is created in SQreamDB. Options are: • `full`: The entire data of the source table is loaded into SQreamDB • `cdc` (Change Data Capture): Only changes made to the source table data since last load will be loaded into SQreamDB. Changes include transactions of `INSERT`, `UPDATE`, and `DELETE` statements.

Table 7 – continued from previous page

CLI Parameter	State	Default	Type	Description
`-use_dbms_lob`	Optional	`true`		Defines whether or not SQLoader uses `dbms_lob_substr` function for `CLOB` and `BLOB` data types
`-use_partitions`	Optional	`true`		Defines whether or not SQLoader uses partitions in `SELECT` statements

9.7.3.2 Data Types and Mapping

SQLoader automatically assigns data types during the data loading process. Nevertheless, you retain the choice to manually specify the preferred data type you want to map to during the loading operation.

- *Automatic Mapping*
- *Manual Mapping*

9.7.3.2.1 Automatic Mapping

9.7.3.2.1.1 Oracle

Oracle Type	SQreamDB Type
`BIGINT`, `INT`, `SMALLINT`, `INTEGE`	`BIGINT`
`CHAR`, `NCHAR`, `VARCHAR`, `VARCHAR2`, `NVARCHAR`, `CHARACTER`	`TEXT`
`DATE`, `DATETIME`	`DATETIME`
`TIMESTAMP`	`DATETIME`
`DATE`	`DATE`
`BOOLEAN`	`BOOL`
`NUMERIC`	`NUMERIC`
`FLOAT`, `DOUBLE`	`DOUBLE`
`CLOB`	`TEXT`
`BLOB`	`TEXT`
`RAW`	`TEXT`

9.7.3.2.1.2 Postgresql

Postgresql Type	SQreamDB Type
CHAR, VARCHAR, CHARACTER	TEXT
TEXT	TEXT
INT, SMALLINT, BIGINT, INT2, INT4, INT8	BIGINT
DATETIME, TIMESTAMP	DATETIME
DATE	DATE
BIT, BOOL	BOOL
DECIMAL, NUMERIC	NUMERIC
FLOAT, DOUBLE	DOUBLE
REAL, FLOAT4	REAL

9.7.3.2.1.3 Teradata

Teradata Type	SQreamDB Type
F	DOUBLE
N, D	NUMERIC
CO	TEXT
BO	TEXT
A1, AN, AT, BF, BV, CF, CV, JN, PD, PM, PS, PT, PZ, SZ, TZ	TEXT
I, I4, I(4)	INT
I2, I(2)	SMALLINT
I1, I(1)	TINYINT
DH, DM, DS, DY, HM, HS, HR, I8, MO, MS, MI, SC, YM, YR	BIGINT
TS, DATETIME	DATETIME
DA	DATE
BIT	BOOL
REAL, DOUBLE	DOUBLE

9.7.3.2.1.4 Microsoft SQL Server

Microsoft SQL Server Type	SQreamDB Type
CHAR, NCHAR, VARCHAR, NVARCHAR, NVARCHAR2, CHARACTER, TEXT, NTEXT	TEXT
BIGINT, INT, SMALLINT, INT, TINYINT	BIGINT
DATETIME, TIMESTAMP, SMALLDATETIME, DATETIMEOFFSET, DATETIME2	DATETIME
DATE	DATE
BIT	BOOL
DECIMAL, NUMERIC	NUMERIC
FLOAT, DOUBLE	DOUBLE
REAL	REAL
VARBINARY	TEXT

9.7.3.2.1.5 SAP HANA

SAP HANA Type	SQreamDB Type
BIGINT, INT, SMALLINT, INTEGER, TINYINT	BIGINT
CHAR, VARCHAR, NVARCHAR, TEXT, VARCHAR2, NVARCHAR2	TEXT
DATETIME, TIMESTAMP, SECONDDATE	DATETIME
DATE	DATE
BOOLEAN	TEXT
DECIMAL, SMALLDECIMAL, BIGDECIMAL	NUMERIC
DOUBLE, REAL	FLOAT
TEXT	TEXT
BIGINT	BIGINT
INT	INT
SMALLINT	SMALLINT
TINYINT	TINYINT
DATETIME	DATETIME
DATE	DATE
BOOL	BOOL
NUMERIC	NUMERIC
DOUBLE	DOUBLE
FLOAT	FLOAT
REAL	REAL

9.7.3.2.2 Manual Mapping

You have the possibility to adjust the mapping process according to your specific needs, using any of the following methods.

9.7.3.2.2.1 `names` Method

To specify that you want to map one or more columns in your table to a specific data type, duplicate the code block which maps to the SQreamDB data type you want and include the `names` parameter in your code block. The SQLoader will map the specified columns to the specified SQreamDB data type. After the specified columns are mapped, the SQLoader continue to search for how to convert other data types to the same data type of the specified columns.

In this example, `column1`, `column2`, and `column3` are mapped to `BIGINT` and the Oracle data types `BIGINT`, `INT`, `SMALLINT`, `INTEGER` are also mapped to `BIGINT`.

```
{
  "oracle": [
        {
          "names": ["column1", "column2", "column3"],
          "sqream": "bigint",
          "java": "int",
          "length": false
        },
        {
          "type": ["bigint","int","smallint","integer"],
          "sqream": "bigint",
          "java": "int",
          "length": false
```

(continues on next page)

(continued from previous page)

```
        }
}
```

```
        {
          "type": ["char","nchar","varchar","varchar2","nvarchar","nvarchar2",
↪"character"],
          "sqream": "text",
          "java": "string",
          "length": true
        },
        {
          "type": ["date","datetime"],
          "sqream": "datetime",
          "java": "datetime",
          "length": false
        },
        {
          "type": ["timestamp"],
          "sqream": "datetime",
          "java": "timestamp",
          "length": false
        },
        {
          "type": ["date"],
          "sqream": "date",
          "java": "datetime",
          "length": false
        },
        {
          "type": ["boolean"],
          "sqream": "bool",
          "java": "boolean",
          "length": false
        },
        {
          "type": ["number"],
          "sqream": "numeric",
          "java": "bigdecimal",
          "length": true,
          "prec": true
        },
        {
          "type": ["float","double"],
          "sqream": "double",
          "java": "double",
          "length": false
        },
        {
          "type": ["clob"],
          "sqream": "text",
          "java": "clob",
          "length": false
        },
        {
          "type": ["blob"],
          "sqream": "text",
```

(continues on next page)

(continued from previous page)

```
            "java": "blob",
            "length": false
          }
  ]
}
```

9.7.4 Creating Summary Tables

Summary tables are pre-aggregated tables that store summarized or aggregated data, which can help improve query performance and reduce the need for complex calculations during runtime.

Summary tables are part of the schema within the database catalog.

9.7.4.1 Examples

The following examples use Oracle syntax.

9.7.4.1.1 A Summary Table

```
CREATE TABLE public.SQLOAD_SUMMARY (
  DB_NAME TEXT(200 BYTE) VISIBLE,
  SCHEMA_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME_FULL TEXT(200 BYTE) VISIBLE,
  LOAD_TYPE TEXT(200 BYTE) VISIBLE,
  UPDATED_DTTM_FROM DATE VISIBLE,
  UPDATED_DTTM_TO DATE VISIBLE,
  LAST_VAL_INT NUMBER(22,0) VISIBLE,
  LAST_VAL_TS DATE VISIBLE,
  START_TIME TIMESTAMP(6) VISIBLE,
  FINISH_TIME TIMESTAMP(6) VISIBLE,
  ELAPSED_SEC NUMBER VISIBLE,
  ROW_COUNT NUMBER VISIBLE,
  SQL_FILTER TEXT(200 BYTE) VISIBLE,
  PARTITION TEXT(200 BYTE) VISIBLE,
  STMT_TYPE TEXT(200 BYTE) VISIBLE,
  STATUS TEXT(200 BYTE) VISIBLE,
  LOG_FILE TEXT(200 BYTE) VISIBLE,
  DB_URL TEXT(200 BYTE) VISIBLE,
  PARTITION_COUNT NUMBER VISIBLE DEFAULT 0,
  THREAD_COUNT NUMBER VISIBLE DEFAULT 1,
  ELAPSED_MS NUMBER VISIBLE DEFAULT 0,
  STATUS_CODE NUMBER VISIBLE DEFAULT 0,
  ELAPSED_SOURCE_MS NUMBER(38,0) DEFAULT NULL,
  ELAPSED_SOURCE_SEC NUMBER(38,0) DEFAULT NULL,
  ELAPSED_TARGET_MS NUMBER(38,0) DEFAULT NULL,
  ELAPSED_TARGET_SEC NUMBER(38,0) DEFAULT NULL,
  TARGET_DB_URL VARCHAR2(200) DEFAULT NULL,
  SQLOADER_VERSION VARCHAR2(20) DEFAULT NULL,
  HOST VARCHAR2(200) DEFAULT NULL
);
```

9.7.4.2 Change Data Capture Summary Tables

Change Data Capture (CDC) tables are supported only for Oracle.

```
CREATE TABLE public.CDC_TABLES (
  DB_NAME TEXT(200 BYTE) VISIBLE,
  SCHEMA_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME_FULL TEXT(200 BYTE) VISIBLE,
  TABLE_NAME_CDC TEXT(200 BYTE) VISIBLE,
  INC_COLUMN_NAME TEXT(200 BYTE) VISIBLE,
  INC_COLUMN_TYPE TEXT(200 BYTE) VISIBLE,
  LOAD_TYPE TEXT(200 BYTE) VISIBLE,
  FREQ_TYPE TEXT(200 BYTE) VISIBLE,
  FREQ_INTERVAL NUMBER(22,0) VISIBLE,
  IS_ACTIVE NUMBER VISIBLE DEFAULT 0,
  STATUS_LOAD NUMBER VISIBLE DEFAULT 0,
  INC_GAP_VALUE NUMBER VISIBLE DEFAULT 0
);

CREATE TABLE public.CDC_TRACKING (
  DB_NAME TEXT(200 BYTE) VISIBLE,
  SCHEMA_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME_FULL TEXT(200 BYTE) VISIBLE,
  LAST_UPDATED_DTTM DATE VISIBLE,
  LAST_VAL_INT NUMBER(22,0) VISIBLE DEFAULT 0,
  LAST_VAL_TS TIMESTAMP(6) VISIBLE,
  LAST_VAL_DT DATE VISIBLE
);

CREATE TABLE public.CDC_TABLE_PRIMARY_KEYS (
  DB_NAME TEXT(200 BYTE) VISIBLE,
  SCHEMA_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME TEXT(200 BYTE) VISIBLE,
  TABLE_NAME_FULL TEXT(200 BYTE) VISIBLE,
  CONSTRAINT_NAME TEXT(200 BYTE) VISIBLE,
  COLUMN_NAME TEXT(200 BYTE) VISIBLE,
  IS_NULLABLE NUMBER VISIBLE DEFAULT 0
);
```

9.7.5 CLI Examples

Loading data into a CDC table using the `type` and `limit` parameters:

```
java -jar sqloader.jar -table <source_table_name> -type cdc -limit 100
```

Loading data into a table using your own configuration file (this will override the default configuration file):

```
java -jar sqloader.jar -config path/to/your/config/file
```

Loading data into a table using a custom configuration file:

```
java -jar -config MyConfigFile.properties -table <source_table_name> -type cdc -
↪target <target_table_name> -drop true -lock_check false
```

Loading data into a table using a the `filter` parameter:

```
java -jar sqloader.jar -table <source_table_name> -filter column_name>50
```

For information about database tools and interfaces that SQream supports, see *Third Party Tools*.

CHAPTER

TEN

EXTERNAL STORAGE PLATFORMS

SQream supports the following external storage platforms:

10.1 Azure Blob Storage

Azure Blob Storage (ABS) is a scalable object storage solution within Microsoft Azure, designed to store and manage vast amounts of unstructured data.

10.1.1 ABS Bucket File Location

ABS syntax to be used for specifying a single or multiple file location within an ABS bucket:

```
azure://accountname.core.windows.net/path
```

10.1.2 Connection String

Connection String Example:

```
"DefaultEndpointsProtocol=https;AccountName=myaccount101;AccountKey=##################
↪#####################==;EndpointSuffix=core.windows.net"
```

Use the following parameters within your SQreamDB legacy configuration file for authentication:

Parameter	Description
`DefaultEndpointsProtocol`	Specifies the protocol (e.g., https or http) used for accessing the storage service
`AccountName`	Represents the unique name of your Azure Storage account
`AccountKey`	Acts as the primary access key for securely authenticating and accessing resources within the storage account
`EndpointSuffix`	Denotes the Azure Storage service endpoint suffix for a specific region or deployment, such as `.core.windows.net`

10.1.3 Examples

```
COPY table_name FROM WRAPPER csv_fdw OPTIONS(location = 'azure://sqreamrole.core.
↪windows.net/sqream-demo-data/file.csv');
```

10.2 Google Cloud Platform

Ingesting data using Google Cloud Platform (GCP) requires configuring Google Cloud Storage (GCS) bucket access. You may configure SQreamDB to separate source and destination by granting read access to one bucket and write access to a different bucket. Such separation requires that each bucket be individually configured.

10.2.1 GCP Bucket File Location

GCP syntax to be used for specifying a single or multiple file location within a GCP bucket:

```
gs://<gcs path>/<gcs_bucket>/
```

10.2.2 GCP Access

10.2.2.1 Before You Begin

It is essential that you have a GCP service account string.

String example:

```
sample_service_account@sample_project.iam.gserviceaccount.com
```

10.2.2.2 Granting GCP Access

1. In your Google Cloud console, go to **Select a project** and select the desired project.
2. From the **PRODUCTS** menu, select **Cloud Storage > Buckets**.
3. Select the bucket you wish to configure; or create a new bucket by selecting **CREATE** and following the **Create a bucket** procedure, and select the newly created bucket.
4. Select **UPLOAD FILES** and upload the data files you wish SQreamDB to ingest.
5. Go to **PERMISSIONS** and select **GRANT ACCESS**.
6. Under **Add principals**, in the **New principals** box, paste your service account string.
7. Under **Assign roles**, in the **Select a role** box, select **Storage Admin**.
8. Select **ADD ANOTHER ROLE** and in the newly created **Select a role** box, select **Storage Object Admin**.
9. Select **SAVE**.

Note: Optimize access time to your data by configuring the location of your bucket according to Google Cloud location considerations.

10.2.3 Examples

Using the `COPY FROM` command:

```
CREATE TABLE nba
  (
    name     TEXT,
    team     TEXT,
    number   TEXT,
    position TEXT,
    age      TEXT,
    height   TEXT,
    weight   TEXT,
    college  TEXT,
    salary   TEXT
  );
```

```
COPY nba FROM
WRAPPER csv_fdw
OPTIONS(location = 'gs://blue_docs/nba.csv');
```

Using the `CREATE FOREIGN TABLE` command:

```
CREATE FOREIGN TABLE nba
(
  Name       TEXT,
  Team       TEXT,
  Number     TEXT,
  Position   TEXT,
  Age        TEXT,
  Height     TEXT,
  Weight     TEXT,
  College    TEXT,
  Salary     TEXT
 )
 WRAPPER csv_fdw
 OPTIONS
 (
   LOCATION =  'gs://blue_docs/nba.csv'
 );
```

10.3 HDFS Environment

10.3.1 Configuring an HDFS Environment for the User sqream

This section describes how to configure an HDFS environment for the user **sqream** and is only relevant for users with an HDFS environment.

To configure an HDFS environment for the user sqream:

1. Open your **bash_profile** configuration file for editing:

   ```
   $ vim /home/sqream/.bash_profile
   ```

2. Verify that the edits have been made:

```
source /home/sqream/.bash_profile
```

3. Check if you can access Hadoop from your machine:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

4. Verify that an HDFS environment exists for SQream services:

```
$ ls -l /etc/sqream/sqream_env.sh
```

5. If an HDFS environment does not exist for SQream services, create one (sqream_env.sh):

```
$ #!/bin/bash

$ SQREAM_HOME=/usr/local/sqream
$ export SQREAM_HOME

$ export JAVA_HOME=${SQREAM_HOME}/hdfs/jdk
$ export HADOOP_INSTALL=${SQREAM_HOME}/hdfs/hadoop
$ export CLASSPATH=`${HADOOP_INSTALL}/bin/hadoop classpath --glob`
$ export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_INSTALL}/lib/native
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${SQREAM_HOME}/lib:$HADOOP_COMMON_LIB_
↪NATIVE_DIR


$ PATH=$PATH:$HOME/.local/bin:$HOME/bin:${SQREAM_HOME}/bin/:${JAVA_HOME}/bin:
↪$HADOOP_INSTALL/bin
$ export PATH
```

Back to top

10.3.2 Authenticating Hadoop Servers that Require Kerberos

If your Hadoop server requires Kerberos authentication, do the following:

1. Create a principal for the user **sqream**.

```
$ kadmin -p root/admin@SQ.COM
$ addprinc sqream@SQ.COM
```

2. If you do not know yor Kerberos root credentials, connect to the Kerberos server as a root user with ssh and run **kadmin.local**:

```
$ kadmin.local
```

   Running **kadmin.local** does not require a password.

3. If a password is not required, change your password to **sqream@SQ.COM**.

```
$ change_password sqream@SQ.COM
```

4. Connect to the hadoop name node using ssh:

```
$ cd /var/run/cloudera-scm-agent/process
```

5. Check the most recently modified content of the directory above:

```
$ ls -lrt
```

6. Look for a recently updated folder containing the text **hdfs**.

The following is an example of the correct folder name:

```
cd <number>-hdfs-<something>
```

This folder should contain a file named **hdfs.keytab** or another similar .keytab file.

7. Copy the .keytab file to user **sqream's** Home directory on the remote machines that you are planning to use Hadoop on.
8. Copy the following files to the **sqream sqream@server:<sqream folder>/hdfs/hadoop/etc/hadoop:** directory:
   - core-site.xml
   - hdfs-site.xml
9. Connect to the sqream server and verify that the .keytab file's owner is a user sqream and is granted the correct permissions:

```
$ sudo chown sqream:sqream /home/sqream/hdfs.keytab
$ sudo chmod 600 /home/sqream/hdfs.keytab
```

10. Log into the sqream server.
11. Log in as the user **sqream**.
12. Navigate to the Home directory and check the name of a Kerberos principal represented by the following .keytab file:

```
$ klist -kt hdfs.keytab
```

The following is an example of the correct output:

```
$ sqream@Host-121 ~ $ klist -kt hdfs.keytab
$ Keytab name: FILE:hdfs.keytab
$ KVNO Timestamp           Principal
$ ---- ------------------- ------------------------------------------------------
↪----
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
```

13. Verify that the hdfs service named **hdfs/nn1@SQ.COM** is shown in the generated output above.
14. Run the following:

```
$ kinit -kt hdfs.keytab hdfs/nn1@SQ.COM
```

15. Check the output:

```
$ klist
```

The following is an example of the correct output:

```
$ Ticket cache: FILE:/tmp/krb5cc_1000
$ Default principal: sqream@SQ.COM
$
$ Valid starting       Expires              Service principal
$ 09/16/2020 13:44:18  09/17/2020 13:44:18  krbtgt/SQ.COM@SQ.COM
```

16. List the files located at the defined server name or IP address:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

17. Do one of the following:
    - If the list below is output, continue with Step 16.
    - If the list is not output, verify that your environment has been set up correctly.

If any of the following are empty, verify that you followed *Step 6* in the **Configuring an HDFS Environment for the User sqream** section above correctly:

```
$ echo $JAVA_HOME
$ echo $SQREAM_HOME
$ echo $CLASSPATH
$ echo $HADOOP_COMMON_LIB_NATIVE_DIR
$ echo $LD_LIBRARY_PATH
$ echo $PATH
```

18. Verify that you copied the correct keytab file.
19. Review this procedure to verify that you have followed each step.

Back to top

10.4 Amazon Web Services

SQreamDB uses a native Amazon Simple Storage Services (S3) connector for inserting data.

10.4.1 S3 Bucket File Location

S3 syntax to be used for specifying a single or multiple file location within an S3 bucket:

```
s3://bucket_name/path
```

10.4.2 S3 Access

A best practice for granting access to AWS S3 is by creating an Identity and Access Management (IAM) user account. If creating an IAM user account is not possible, you may follow AWS guidelines for using the global configuration object and setting an AWS region.

10.4.3 Authentication

After being granted access to an S3 bucket, you'll be able to execute statements using the `AWS ID` and `AWS SECRET` parameters for authentication.

10.4.4 Connecting to S3 Using SQreamDB Legacy Configuration File

You may use the following parameters within your SQreamDB legacy configuration file:

Parameter	Description	Parameter Value	Example

`AwsEndpointOverride`	Overrides the AWS S3 HTTP endpoint when using Virtual Private Cloud (VPC)	**URL** Default: None	`sqream_config_legacy.json:`
			`{`
			`...,`
			`"AwsEndpointOverride": "https://my.endpoint.local"`
			`}`

`AwsObjectAccessStyle`	Enables configuration of S3 object access styles, which determine how you can access and interact with the objects stored in an S3 bucket	`virtual-host` or `path`. Default is `virtual-host`	`sqream_config_legacy.json:`
			`{`
			`...,`
			`"AwsObjectAccessStyle": "path"`
			`}`

10.4.5 Examples

10.4.5.1 Creating a Foreign Table

Based on the source file's structure, you can create a foreign table with the appropriate structure, and point it to your file as shown in the following example:

```
CREATE FOREIGN TABLE nba
(
   Name text(40),
   Team text(40),
   Number tinyint,
   Position text(2),
   Age tinyint,
```

(continues on next page)

(continued from previous page)

```
  Height text(4),
  Weight real,
  College text(40),
  Salary float
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = 's3://sqream-demo-data/nba_players.csv',
     RECORD_DELIMITER = '\r\n' -- DOS delimited file
  )
;
```

In the example above the file format is CSV, and it is stored as an S3 object. If the path is on HDFS, you must change the URI accordingly. Note that the record delimiter is a DOS newline (`\r\n`).

10.4.5.2 Querying Foreign Tables

The following shows the data in the foreign table:

```
t=> SELECT * FROM nba LIMIT 10;
name           | team           | number | position | age | height | weight | college
↪          | salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma
↪State     |  3431040
```

10.4.5.3 Bulk Loading a File from a Public S3 Bucket

The `COPY FROM` command can also be used to load data without staging it first.

The bucket must be publicly available and objects must be listed.

```
COPY nba FROM 's3://sqream-demo-data/nba.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n';
```

10.4.5.4 Loading Files from an Authenticated S3 Bucket

```
COPY nba FROM 's3://secret-bucket/*.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n'
AWS_ID '12345678'
AWS_SECRET 'super_secretive_secret';
```

For more information, see the following:

- *Foreign Tables*
- copy_from
- copy_to

CHAPTER

ELEVEN

FEATURE GUIDES

The **Feature Guides** section describes background processes that SQreamDB uses to manage several areas of operation, such as data ingestion, load balancing, and access control.

11.1 Automatic Foreign Table DDL Resolution

SQream must be able to access a schema when reading and mapping external files to a foreign table. To facilitate this, you must specify the correct schema in the statement that creates the foreign table, which must also include the correct list of columns. To avoid human error related to this complex process SQream can now automatically identify the corresponding schema, saving you the time and effort required to build your schema manually. This is especially useful for particular file formats, such as Parquet, which include a built-in schema declaration.

- *Usage Notes*
- *Syntax*
- *Example*
- *Permissions*

11.1.1 Usage Notes

The automatic foreign table DDL resolution feature supports Parquet, ORC, JSON, and Avro files, while using it with CSV files generates an error. You can activate this feature when you create a foreign table by omitting the column list, described in the **Syntax** section below.

Using this feature the path you specify in the `LOCATION` option must point to at least one existing file. If no files exist for the schema to read, an error will be generated. You can specify the schema manually even in the event of the error above.

Note: When using this feature, SQream assumes that all files in the path use the same schema.

11.1.2 Syntax

The following is the syntax for using the automatic foreign table DDL resolution feature:

```
CREATE FOREIGN TABLE table_name
[FOREIGN DATA] WRAPPER fdw_name
[OPTIONS (...)]
```

11.1.3 Example

The following is an example of using the automatic foreign table DDL resolution feature:

```
create foreign table parquet_table
wrapper parquet_fdw
options (location = '/tmp/file.parquet');
```

11.1.4 Permissions

The automatic foreign table DDL resolution feature requires **Read** permissions.

11.2 Query Healer

The **Query Healer** periodically examines the progress of running statements, creating a log entry for all statements exceeding a defined time period.

11.2.1 Configuration

The following worker flags are required to configure the Query Healer:

Flag	Description
`is_healer_on`	The is_healer_on enables and disables the Query Healer.
`maxStatementInactivitySeconds`	The healer_max_statement_inactivity_seconds worker level flag defines the threshold for creating a log recording a slow statement. The log includes information about the log memory, CPU and GPU. The default setting is five hours.
`healerDetectionFrequencySeconds`	The healer_detection_frequency_seconds worker level flag triggers the healer to examine the progress of running statements. The default setting is one hour.

11.2.2 Query Log

The following is an example of a log record for a query stuck in the query detection phase for more than five hours:

```
|INFO|0x00007f9a497fe700:Healer|192.168.4.65|5001|-1|master|sqream|-1|sqream|0|
↪"[ERROR]|cpp/SqrmRT/healer.cpp:140 |"Stuck query found. Statement ID: 72, Last␣
↪chunk producer updated: 1.
```

Once you identify the stuck worker, you can execute the `shutdown_server` utility function from this specific worker, as described in the next section.

11.2.3 Activating a Graceful Shutdown

You can activate a graceful shutdown if your log entry says `Stuck query found`, as shown in the example above. You can do this by setting the **shutdown_server** utility function to `select shutdown_server();`.

To activte a graceful shutdown:

1. Locate the IP and the Port of the stuck worker from the logs.

   Note: The log in the previous section identifies the IP (**192.168.4.65**) and port (**5001**) referring to the stuck query.

2. From the machine of the stuck query (IP: **192.168.4.65**, port: **5001**), connect to SQream SQL client:

```
./sqream sql --port=$STUCK_WORKER_IP --username=$SQREAM_USER --password=$SQREAM_
↪PASSWORD databasename=$SQREAM_DATABASE
```

3. Execute `shutdown_server`.

For more information, see the shutdown_server_command utility function. This page describes all of `shutdown_server` options.

11.3 Compression

SQreamDB uses a variety of compression and encoding methods to optimize query performance and to save disk space.

- *Encoding*
- *Lossless Compression*
- *Best Practices*

11.3.1 Encoding

Encoding is an automatic operation used to convert data into common formats. For example, certain formats are often used for data stored in columnar format, in contrast with data stored in a CSV file, which stores all data in text format.

Encoding enhances performance and reduces data size by using specific data formats and encoding methods. SQream encodes data in a number of ways in accordance with the data type. For example, a **date** is stored as an **integer**, starting with **March 1st 1CE**, which is significantly more efficient than encoding the date as a string. In addition, it offers a wider range than storing it relative to the Unix Epoch.

11.3.2 Lossless Compression

Compression transforms data into a smaller format without sacrificing accuracy, known as **lossless compression**.

After encoding a set of column values, SQream packs the data and compresses it and decompresses it to make it accessible to users. Depending on the compression scheme used, these operations can be performed on the CPU or the GPU. Some users find that GPU compression provide better performance.

11.3.2.1 Automatic Compression

By default, SQream automatically compresses every column (see *Specifying Compression Strategies* below for overriding default compression). This feature is called **automatic adaptive compression** strategy.

When loading data, SQreamDB automatically decides on the compression schemes for specific chunks of data by trying several compression schemes and selecting the one that performs best. SQreamDB tries to balance more aggressive compression with the time and CPU/GPU time required to compress and decompress the data.

11.3.2.2 Compression Methods

The following table shows the supported compression methods:

Compression Method	Supported Data Types	Description	Location
`FLAT`	All types	No compression (forced)	NA
`DEFAULT`	All types	Automatic scheme selection	NA
`DICT`	All types	Dictionary compression with RLE. For each chunk, SQreamDB creates a dictionary of distinct values and stores only their indexes. Works best for integers and texts shorter than 120 characters, with <10% unique values. Useful for storing ENUMs or keys, stock tickers, and dimensions. If the data is optionally sorted, this compression will perform even better.	GPU
`P4D`	`Integer`, `dates`, `timestamps`, and `float` types	Patched frame-of-reference + Delta Based on the delta between consecutive values. Works best for monotonously increasing or decreasing numbers and timestamps	GPU
`LZ4`	Text types	Lempel-Ziv general purpose compression, used for texts	CPU
`SNAPPY`	Text types	General purpose compression, used for texts	CPU
`RLE`	`integer` types, `dates`, `timestamps`, and `text`	Run-Length Encoding. This replaces sequences of values with a single pair. It is best for low cardinality columns that are used to sort data (`ORDER BY`).	GPU
`SEQUENCE`	`Integer`, `date`, and `timestamp`	Optimized RLE + Delta type for built-in identity columns.	GPU

11.3.2.3 Specifying Compression Strategies

When you create a table without defining any compression specifications, SQream defaults to automatic adaptive compression (`"default"`). However, you can prevent this by specifying a compression strategy when creating a table.

This section describes the following compression strategies:

- *Explicitly Specifying Automatic Compression*
- *Forcing No Compression*
- *Forcing Compression*

11.3.2.3.1 Explicitly Specifying Automatic Compression

When you explicitly specify automatic compression, the following two are equivalent:

```
CREATE TABLE t (
   x INT,
   y TEXT(50)
);
```

In this version, the default compression is specified explicitly:

```
CREATE TABLE t (
   x INT CHECK('CS "default"'),
   y TEXT(50) CHECK('CS "default"')
);
```

11.3.2.3.2 Forcing No Compression

Forcing no compression is also known as "flat", and can be used in the event that you want to remove compression entirely on some columns. This may be useful for reducing CPU or GPU resource utilization at the expense of increased I/O.

The following is an example of removing compression:

```
CREATE TABLE t (
   x INT NOT NULL CHECK('CS "flat"'), -- This column won't be compressed
   y TEXT(50) -- This column will still be compressed automatically
);
```

11.3.2.3.3 Forcing Compression

In other cases, you may want to force SQream to use a specific compression scheme based on your knowledge of the data, as shown in the following example:

```
CREATE TABLE t (
   id BIGINT NOT NULL CHECK('CS "sequence"'),
   y TEXT(110) CHECK('CS "lz4"'), -- General purpose text compression
   z TEXT(80) CHECK('CS "dict"'), -- Low cardinality column
```

(continues on next page)

(continued from previous page)

```
);
```

However, if SQream finds that the given compression method cannot effectively compress the data, it will return to the default compression type.

11.3.2.4 Examining Compression Effectiveness

Queries made on the internal metadata catalog can expose how effective the compression is, as well as what compression schemes were selected.

This section describes the following:

- *Querying the Catalog*
- *Example Subset from "Ontime" Table*
- *Notes on Reading the "Ontime" Table*

11.3.2.4.1 Querying the Catalog

The following is a sample query that can be used to query the catalog:

```
SELECT c.column_name AS "Column",
       cc.compression_type AS "Actual compression",
       AVG(cc.compressed_size) "Compressed",
       AVG(cc.uncompressed_size) "Uncompressed",
       AVG(cc.uncompressed_size::FLOAT/ cc.compressed_size) -1 AS "Compression␣
↪effectiveness",
       MIN(c.compression_strategy) AS "Compression strategy"
 FROM sqream_catalog.chunk_columns cc
   INNER JOIN sqream_catalog.columns c
           ON cc.table_id = c.table_id
          AND cc.database_name = c.database_name
          AND cc.column_id = c.column_id

   WHERE c.table_name = 'some_table'  -- This is the table name which we want to␣
↪inspect

   GROUP BY 1,
            2;
```

11.3.2.4.2 Example Subset from "Ontime" Table

The following is an example (subset) from the `ontime` table:

```
stats=> SELECT c.column_name AS "Column",
.          cc.compression_type AS "Actual compression",
.          AVG(cc.compressed_size) "Compressed",
.          AVG(cc.uncompressed_size) "Uncompressed",
.          AVG(cc.uncompressed_size::FLOAT/ cc.compressed_size) -1 AS "Compression␣
↪effectiveness",
.          MIN(c.compression_strategy) AS "Compression strategy"
.   FROM sqream_catalog.chunk_columns cc
.     INNER JOIN sqream_catalog.columns c
.             ON cc.table_id = c.table_id
.            AND cc.database_name = c.database_name
.            AND cc.column_id = c.column_id
.
.   WHERE c.table_name = 'ontime'
.
.   GROUP BY 1,
.            2;

Column                     | Actual compression | Compressed | Uncompressed |␣
↪Compression effectiveness | Compression strategy
---------------------------+--------------------+------------+--------------+----------
↪-----------------+---------------------
actualelapsedtime@null     | dict               |     129177 |      1032957 |         ␣
↪               7 | default
actualelapsedtime@val      | dict               |    1379797 |      4131831 |         ␣
↪               2 | default
airlineid                  | dict               |     578150 |      2065915 |         ␣
↪             2.7 | default
airtime@null               | dict               |     130011 |      1039625 |         ␣
↪               7 | default
airtime@null               | rle                |      93404 |      1019833 |         ␣
↪       116575.61 | default
airtime@val                | dict               |    1142045 |      4131831 |         ␣
↪            7.57 | default
arrdel15@null              | dict               |     129177 |      1032957 |         ␣
↪               7 | default
arrdel15@val               | dict               |     129183 |      4131831 |         ␣
↪           30.98 | default
arrdelay@null              | dict               |     129177 |      1032957 |         ␣
↪               7 | default
arrdelay@val               | dict               |    1389660 |      4131831 |         ␣
↪               2 | default
arrdelayminutes@null       | dict               |     129177 |      1032957 |         ␣
↪               7 | default
arrdelayminutes@val        | dict               |    1356034 |      4131831 |         ␣
↪            2.08 | default
arrivaldelaygroups@null    | dict               |     129177 |      1032957 |         ␣
↪               7 | default
arrivaldelaygroups@val     | p4d                |     516539 |      2065915 |         ␣
↪               3 | default
arrtime@null               | dict               |     129177 |      1032957 |         ␣
↪               7 | default
arrtime@val                | p4d                |    1652799 |      2065915 |         ␣
```

(continues on next page)

(continued from previous page)

```
↪            0.25 | default
arrtimeblk                | dict                |     688870 |      9296621 |          ␣
↪           12.49 | default
cancellationcode@null     | dict                |     129516 |      1035666 |          ␣
↪               7 | default
cancellationcode@null     | rle                 |      54392 |      1031646 |          ␣
↪       131944.62 | default
cancellationcode@val      | dict                |     263149 |      1032957 |          ␣
↪            4.12 | default
cancelled                 | dict                |     129183 |      4131831 |          ␣
↪           30.98 | default
carrier                   | dict                |     578150 |      2065915 |          ␣
↪             2.7 | default
carrierdelay@null         | dict                |     129516 |      1035666 |          ␣
↪               7 | default
carrierdelay@null         | flat                |    1041250 |      1041250 |          ␣
↪               0 | default
carrierdelay@null         | rle                 |       4869 |      1026493 |          ␣
↪        202740.2 | default
carrierdelay@val          | dict                |     834559 |      4131831 |          ␣
↪           14.57 | default
crsarrtime                | p4d                 |    1652799 |      2065915 |          ␣
↪            0.25 | default
crsdeptime                | p4d                 |    1652799 |      2065915 |          ␣
↪            0.25 | default
crselapsedtime@null       | dict                |     130449 |      1043140 |          ␣
↪               7 | default
crselapsedtime@null       | rle                 |       3200 |      1013388 |          ␣
↪       118975.75 | default
crselapsedtime@val        | dict                |    1182286 |      4131831 |          ␣
↪             2.5 | default
dayofmonth                | dict                |     688730 |      1032957 |          ␣
↪             0.5 | default
dayofweek                 | dict                |     393577 |      1032957 |          ␣
↪            1.62 | default
departuredelaygroups@null | dict                |     129177 |      1032957 |          ␣
↪               7 | default
departuredelaygroups@val  | p4d                 |     516539 |      2065915 |          ␣
↪               3 | default
depdel15@null             | dict                |     129177 |      1032957 |          ␣
↪               7 | default
depdel15@val              | dict                |     129183 |      4131831 |          ␣
↪           30.98 | default
depdelay@null             | dict                |     129177 |      1032957 |          ␣
↪               7 | default
depdelay@val              | dict                |    1384453 |      4131831 |          ␣
↪            2.01 | default
depdelayminutes@null      | dict                |     129177 |      1032957 |          ␣
↪               7 | default
depdelayminutes@val       | dict                |    1362893 |      4131831 |          ␣
↪            2.06 | default
deptime@null              | dict                |     129177 |      1032957 |          ␣
↪               7 | default
deptime@val               | p4d                 |    1652799 |      2065915 |          ␣
↪            0.25 | default
deptimeblk                | dict                |     688870 |      9296621 |          ␣
↪           12.49 | default
```

(continues on next page)

(continued from previous page)

```
month                     | dict                |     247852 |      1035246 |          ␣
↪            3.38 | default
month                     | rle                 |          5 |       607346 |          ␣
↪        121468.2 | default
origin                    | dict                |    1119457 |      3098873 |          ␣
↪            1.78 | default
quarter                   | rle                 |          8 |      1032957 |          ␣
↪       136498.61 | default
securitydelay@null        | dict                |     129516 |      1035666 |          ␣
↪               7 | default
securitydelay@null        | flat                |    1041250 |      1041250 |          ␣
↪               0 | default
securitydelay@null        | rle                 |       4869 |      1026493 |          ␣
↪        202740.2 | default
securitydelay@val         | dict                |     581893 |      4131831 |          ␣
↪           15.39 | default
tailnum@null              | dict                |     129516 |      1035666 |          ␣
↪               7 | default
tailnum@null              | rle                 |      38643 |      1031646 |          ␣
↪       121128.68 | default
tailnum@val               | dict                |    1659918 |     12395495 |          ␣
↪           22.46 | default
taxiin@null               | dict                |     130011 |      1039625 |          ␣
↪               7 | default
taxiin@null               | rle                 |      93404 |      1019833 |          ␣
↪       116575.61 | default
taxiin@val                | dict                |     839917 |      4131831 |          ␣
↪            8.49 | default
taxiout@null              | dict                |     130011 |      1039625 |          ␣
↪               7 | default
taxiout@null              | rle                 |      84327 |      1019833 |          ␣
↪       116575.86 | default
taxiout@val               | dict                |     891539 |      4131831 |          ␣
↪            8.28 | default
totaladdgtime@null        | dict                |     129516 |      1035666 |          ␣
↪               7 | default
totaladdgtime@null        | rle                 |       3308 |      1031646 |          ␣
↪       191894.18 | default
totaladdgtime@val         | dict                |     465839 |      4131831 |          ␣
↪           20.51 | default
uniquecarrier             | dict                |     578221 |      7230705 |          ␣
↪           11.96 | default
year                      | rle                 |          6 |      2065915 |          ␣
↪       317216.08 | default
```

11.3.2.4.3 Notes on Reading the "Ontime" Table

The following are some useful notes on reading the "Ontime" table shown above:

1. Higher numbers in the **Compression effectiveness** column represent better compressions. **0** represents a column that has **not been compressed**.

2. Column names are an internal representation. Names with `@null` and `@val` suffixes represent a nullable column's null (boolean) and values respectively, but are treated as one logical column.

3. The query lists all actual compressions for a column, so it may appear several times if the compression has changed mid-way through the loading (as with the `carrierdelay` column).

4. When your compression strategy is `default`, the system automatically selects the best compression, including no compression at all (`flat`).

11.3.3 Best Practices

This section describes the best compression practices:

- *Letting SQream Determine the Best Compression Strategy*
- *Maximizing the Advantage of Each Compression Scheme*
- *Choosing Data Types that Fit Your Data*

11.3.3.1 Letting SQream Determine the Best Compression Strategy

In general, SQream determines the best compression strategy for most cases. If you decide to override SQream's selected compression strategies, we recommend benchmarking your query and load performance **in addition to** your storage size.

11.3.3.2 Maximizing the Advantage of Each Compression Scheme

Some compression schemes perform better when data is organized in a specific way. For example, to take advantage of RLE, sorting a column may result in better performance and reduced disk-space and I/O usage. Sorting a column partially may also be beneficial. As a rule of thumb, aim for run-lengths of more than 10 consecutive values.

11.3.3.3 Choosing Data Types that Fit Your Data

Adapting to the narrowest data type improves query performance while reducing disk space usage. However, smaller data types may compress better than larger types.

For example, SQream recommends using the smallest numeric data type that will accommodate your data. Using `BIGINT` for data that fits in `INT` or `SMALLINT` can use more disk space and memory for query execution. Using `FLOAT` to store integers will reduce compression's effectiveness significantly.

11.4 Python User-Defined Functions

User-Defined Functions (UDFs) offer streamlined statements, enabling the creation of a function once, storing it in the database, and calling it multiple times within a statement. Additionally, UDFs can be shared among roles, created by a database administrator and utilized by others. Furthermore, they contribute to code simplicity by allowing independent modifications in SQream DB without altering program source code.

To enable UDFs, in your *legacy configuration file*, set the `enablePythonUdfs` configuration flag to `true`.

- *Before You Begin*
- *SQreamDB's UDF Support*
- *Working with Existing UDFs*
- *Permissions and Sharing*
- *Example*
- *Best Practices*

11.4.1 Before You Begin

- Ensure you have Python 3.9 or newer installed
- Enable UDFs by setting the `enablePythonUdfs` configuration flag to `true` in your *legacy configuration file*

11.4.2 SQreamDB's UDF Support

11.4.2.1 Scalar Functions

SQreamDB's UDFs are scalar functions. This means that the UDF returns a single data value of the type defined in the `RETURNS` clause. For an inline scalar function, the returned scalar value is the result of a single statement.

11.4.2.2 Python

Python is installed alongside SQreamDB, for use exclusively by SQreamDB. You may have a different version of Python installed on your server.

To find which version of Python is installed for use by SQreamDB, create and run this UDF:

```
master=> CREATE OR REPLACE FUNCTION py_version()
.  RETURNS text
.  AS $$
. import sys
. return ("Python version: " + sys.version + ". Path: " + sys.base_exec_prefix)
.  $$ LANGUAGE PYTHON;
executed
master=> SELECT py_version();
py_version
-------------------------------------------------------------------------------
Python version: 3.9.13 (default, Jul 22 2019, 11:03:54) [GCC 5.4.0].
Path: /opt/sqream/python-3.9.13
```

11.4.2.3 Using Modules

To import a Python module, use the standard `import` syntax in the first lines of the user-defined function.

11.4.3 Working with Existing UDFs

11.4.3.1 Finding Existing UDFs in the Catalog

The `user_defined_functions` catalog view contains function information.

Here's how you'd list all UDFs in the system:

```
master=> SELECT * FROM sqream_catalog.user_defined_functions;
database_name | function_id | function_name
--------------+-------------+--------------
master        |           1 | my_upper
```

11.4.3.2 Getting Function DDL

```
master=> SELECT GET_FUNCTION_DDL('my_upper');
ddl
----------------------------------------------------
create function "my_upper" (x1 text) returns text as
$$
   return x1.upper()
$$
language python volatile;
```

See get_function_ddl for more information.

11.4.3.3 Handling Errors

In UDFs, any error that occurs causes the execution of the function to stop. This in turn causes the statement that invoked the function to be canceled.

11.4.4 Permissions and Sharing

To create a UDF, the creator needs the `CREATE FUNCTION` permission at the database level.

For example, to grant `CREATE FUNCTION` to a non-superuser role:

```
GRANT CREATE FUNCTION ON DATABASE master TO role1;
```

To execute a UDF, the role needs the `EXECUTE FUNCTION` permission for every function.

For example, to grant the permission to the `r_bi_users` role group, run:

```
GRANT EXECUTE ON FUNCTION my_upper TO r_bi_users;
```

Note: Functions are stored for each database, outside of any schema.

See more information about permissions in the *Access control guide*.

11.4.5 Example

Most databases have an UPPER function, including SQream DB. However, assume that this function is missing for the sake of this example.

You can write a function in Python to uppercase a text value using the create_function syntax.

```
CREATE FUNCTION my_upper (x1 text)
  RETURNS text
  AS $$
return x1.upper()
$$ LANGUAGE PYTHON;
```

Let's break down this example:

- `CREATE FUNCTION my_upper` - Create a function called `my_upper`. This name must be unique in the current database
- `(x1 text)` - the function accepts one argument named `x1` which is of the SQL type `TEXT`. All data types are supported.
- `RETURNS text` - the function returns the same type - `TEXT`. All data types are supported.
- `AS $$` - what follows is some code that we don't want to quote, so we use dollar-quoting (`$$`) instead of single quotes (`'`).
- `return x1.upper()` - the Python function's body is the argument named `x1`, uppercased.
- `$$ LANGUAGE PYTHON` - this is the end of the function, and it's in the Python language.

Running this example

After creating the function, you can use it in any SQL query.

For example:

```
master=>CREATE TABLE jabberwocky(line text);
executed
master=> INSERT INTO jabberwocky VALUES
.   ('''Twas brillig, and the slithy toves '), ('      Did gyre and gimble in the␣
↪wabe: ')
.   ,('All mimsy were the borogoves, '), ('      And the mome raths outgrabe. ')
.   ,('"Beware the Jabberwock, my son! '), ('      The jaws that bite, the claws that␣
↪catch! ')
.   ,('Beware the Jubjub bird, and shun '), ('      The frumious Bandersnatch!" ');
executed
master=> SELECT line, my_upper(line) FROM jabberwocky;
line                                                | my_upper
----------------------------------------------------+-------------------------------------
↪-------------
'Twas brillig, and the slithy toves                 | 'TWAS BRILLIG, AND THE SLITHY TOVES
      Did gyre and gimble in the wabe:              |       DID GYRE AND GIMBLE IN THE␣
↪WABE:
All mimsy were the borogoves,                       | ALL MIMSY WERE THE BOROGOVES,
      And the mome raths outgrabe.                  |       AND THE MOME RATHS OUTGRABE.
"Beware the Jabberwock, my son!                     | "BEWARE THE JABBERWOCK, MY SON!
      The jaws that bite, the claws that catch!     |       THE JAWS THAT BITE, THE␣
↪CLAWS THAT CATCH!
Beware the Jubjub bird, and shun                    | BEWARE THE JUBJUB BIRD, AND SHUN
      The frumious Bandersnatch!"                   |       THE FRUMIOUS BANDERSNATCH!"
```

11.4.6 Best Practices

Although user-defined functions add flexibility, they may have some performance drawbacks. They are not usually a replacement for subqueries or views.

In some cases, the user-defined function provides benefits like sharing extended functionality which makes it very appealing.

Use user-defined functions sparingly in the `WHERE` clause. SQream DB can't optimize the function's usage, and it will be called once for every value. If possible, you should narrow down the number of results before the UDF is called by using a subquery.

11.5 Workload Manager

The Workload Manager enables SQream workers to identify their availability to clients with specific service names, allowing a system engineer or database administrator to allocate specific workers and compute resources for various tasks. The load balancer then uses this information to route statements to the designated workers.

For example:

1. Creating a service queue named `ETL` and allocating two workers exclusively to this service prevents non-`ETL` statements from using these compute resources.
2. Creating a service for the company's leadership during working hours for dedicated access, and disabling this service at night to allow maintenance operations to use the available compute.

11.5.1 Setting Up Service Queues

By default, every worker subscribes to the `sqream` service queue.

Additional service names are configured in the configuration file for every worker, but can also be set on a per-session basis.

11.5.2 Example - Allocating ETL Resources

Allocating ETL resources ensures high quality service without requiring management users to wait.

The configuration in this example allocates resources as shown below:

- 1 worker for ETL work
- 3 workers for general queries
- All workers assigned to queries from management

Service / Worker	Worker #1	Worker #2	Worker #3	Worker #4
ETL	✓	✗	✗	✗
Query service	✗	✓	✓	✓
Management	✓	✓	✓	✓

This configuration gives the ETL queue dedicated access to one worker, which cannot be used..

Queries from management uses any available worker.

11.5.2.1 Creating the Configuration

```
{
  "cluster": "/home/rhendricks/raviga_database",
  "cudaMemQuota": 25,
  "gpu": 0,
  "maxConnectionInactivitySeconds": 120,
  "legacyConfigFilePath": "tzah_legacy.json",
  "licensePath": "/home/sqream/.sqream/license.enc",
  "metadataServerIp": "192.168.0.103",
  "limitQueryMemoryGB": 250,
  "machineIP": "192.168.0.103",
  "metadataServerPort": 3105,
  "port": 5000,
  "useConfigIP": true
}
```

Listing 1: Legacy File

```
    {
      "debugNetworkSession": false,
      "diskSpaceMinFreePercent": 1,
      "maxNumAutoCompressedChunksThreshold" : 1,
      "insertMergeRowsThreshold":40000000,
      "insertCompressors": 8,
      "insertParsers": 8,
      "nodeInfoLoggingSec": 60,
      "reextentUse": true,
      "separatedGatherThreads": 16,
      "showFullExceptionInfo": true,
      "spoolMemoryGB":200,
      "useClientLog": true,
      "useMetadataServer":true
    }
```

Tip: You can create this configuration temporarily (for the current session only) by using the subscribe_service and unsubscribe_service statements.

11.5.2.2 Verifying the Configuration

Use show_subscribed_instances to view service subscriptions for each worker. Use SHOW_SERVER_STATUS to see the statement queues.

```
t=> SELECT SHOW_SUBSCRIBED_INSTANCES();
service     | servernode | serverip       | serverport
-----------+------------+---------------+-----------
management | node_9383  | 192.168.0.111 |       5000
etl        | node_9383  | 192.168.0.111 |       5000
query      | node_9384  | 192.168.0.111 |       5001
management | node_9384  | 192.168.0.111 |       5001
query      | node_9385  | 192.168.0.111 |       5002
management | node_9385  | 192.168.0.111 |       5002
query      | node_9551  | 192.168.1.91  |       5000
management | node_9551  | 192.168.1.91  |       5000
```

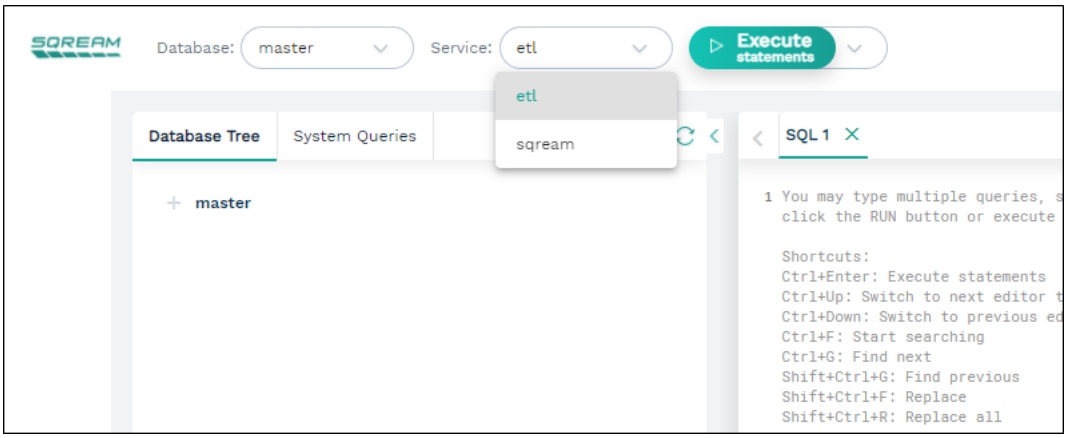
11.5.3 Configuring a Client Connection to a Specific Service

You can configure a client connection to a specific service in one of the following ways:

- *Using SQream Studio*
- *Using the SQream SQL CLI Reference*
- *Using a JDBC Client Driver*
- *Using an ODBC Client Driver*
- *Using a Python Client Driver*
- *Using a Node.js Client Driver*

11.5.3.1 Using SQream Studio

When using **SQream Studio**, you can configure a client connection to a specific service from the SQream Studio, as shown below:



For more information, in Studio, see *Executing Statements from the Toolbar*.

11.5.3.2 Using the SQream SQL CLI Reference

When using the **SQream SQL CLI Reference**, you can configure a client connection to a specific service by adding `--service=<service name>` to the command line, as shown below:

```
$ sqream sql --port=3108 --clustered --username=mjordan --databasename=master --
↪service=etl
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

For more information, see the *Sqream SQL CLI Reference*.

11.5.3.3 Using a JDBC Client Driver

When using a **JDBC client driver**, you can configure a client connection to a specific service by adding `--service=<service name>` to the command line, as shown below:

Listing 2: JDBC Connection String

```
jdbc:Sqream://127.0.0.1:3108/raviga;user=rhendricks;password=Tr0ub4dor&3;service=etl;
↪cluster=true;ssl=false;
```

For more information, see the JDBC Client Driver.

11.5.3.4 Using an ODBC Client Driver

When using an **ODBC client driver**, you can configure a client connection to a specific service on Linux by modifying the *DSN parameters* in `odbc.ini`.

For example, `Service="etl"`:

Listing 3: odbc.ini

```
[sqreamdb]
Description=64-bit Sqream ODBC
Driver=/home/rhendricks/sqream_odbc64/sqream_odbc64.so
Server="127.0.0.1"
Port="3108"
Database="raviga"
Service="etl"
User="rhendricks"
Password="Tr0ub4dor&3"
Cluster=true
Ssl=false
```

On Windows, change the parameter in the *DSN editing window*.

For more information, see the ODBC Client Driver.

11.5.3.5 Using a Python Client Driver

When using a **Python client driver**, you can configure a client connection to a specific service by setting the `service` parameter in the connection command, as shown below:

Listing 4: Python

```
con = pysqream.connect(host='127.0.0.1', port=3108, database='raviga'
                       , username='rhendricks', password='Tr0ub4dor&3'
                       , clustered=True, use_ssl = False, service='etl')
```

For more information, see the Python (pysqream) connector.

11.5.3.6 Using a Node.js Client Driver

When using a **Node.js client driver**, you can configure a client connection to a specific service by adding the service to the connection settings, as shown below:

Listing 5: Node.js

```
const Connection = require('sqreamdb');
const config = {
   host: '127.0.0.1',
   port: 3108,
   username: 'rhendricks',
   password: 'Tr0ub4dor&3',
   connectDatabase: 'raviga',
   cluster: 'true',
   service: 'etl'
};
```

For more information, see the Node.js Client Driver.

11.6 Concurrency and Locks

Locks are used in SQream DB to provide consistency when there are multiple concurrent transactions updating the database.

Read only transactions are never blocked, and never block anything. Even if you drop a database while concurrently running a query on it, both will succeed correctly (as long as the query starts running before the drop database commits).

11.6.1 Locking Modes

SQream DB has two kinds of locks:

- `exclusive` - this lock mode prevents the resource from being modified by other statements

  This lock tells other statements that they'll have to wait in order to change an object.

  DDL operations are always exclusive. They block other DDL operations, and update DML operations (insert and delete).

- `inclusive` - For insert operations, an inclusive lock is obtained on a specific object. This prevents other statements from obtaining an exclusive lock on the object.

  This lock allows other statements to insert or delete data from a table, but they'll have to wait in order to run DDL.

11.6.2 When are Locks Obtained?

Operation	select	insert	delete, truncate	DDL
select	Concurrent	Concurrent	Concurrent	Concurrent
insert	Concurrent	Concurrent	Concurrent	Wait
delete, truncate	Concurrent	Concurrent	Wait	Wait
DDL	Concurrent	Wait	Wait	Wait

Statements that wait will exit with an error if they hit the lock timeout. The default timeout is 3 seconds, see `statementLockTimeout`.

11.6.3 Monitoring Locks

Monitoring locks across the cluster can be useful when transaction contention takes place, and statements appear "stuck" while waiting for a previous statement to release locks.

The utility show_locks can be used to see the active locks.

In this example, we create a table based on results (create_table_as), but we are also effectively dropping the previous table (by using `OR REPLACE` which also drops the table). Thus, SQream DB applies locks during the table creation process to prevent the table from being altered during it's creation.

```
t=> SELECT SHOW_LOCKS();
statement_id | statement_string                                                                 ␣
↪                          | username | server       | port | locked_object                    ␣
↪        | lockmode  | statement_start_time | lock_start_time
-------------+-------------------------------------------------------------------------------
↪-------------------------+----------+--------------+------+-------------------------
↪--------+-----------+----------------------+--------------------
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | database$t                    ␣
↪           | Inclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | globalpermission$             ␣
↪           | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | schema$t$public               ␣
↪           | Inclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | table$t$public$nba2
↪$Insert      | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | table$t$public$nba2
↪$Update      | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
```

For more information on troubleshooting lock related issues, see *Lock Related Issues*.

CHAPTER

TWELVE

REFERENCES

The **Reference Guides** section provides reference for using SQream DB's interfaces and SQL features.

12.1 SQL Statements and Syntax

This section provides reference for using SQream DB's SQL statements - *DDL commands*, *DML commands* and *SQL query syntax*.

12.1.1 SQL Syntax Features

SQreamDB supports SQL from the ANSI 92 syntax.

Features	Description
keywords_and_identifiers	Keywords are reserved words with specific meanings, while identifiers are used to name database objects like tables and columns.
literals	Literals are fixed values representing specific data types, such as numbers or strings, used directly in SQL statements.
scalar_expressions	Scalar expressions are single-value computations that operate on one or more values to produce a single result.
cross_database_queries	Cross-database queries involve accessing and manipulating data from multiple databases within a single SQL statement or operation.
joins	Joins combine rows from two or more tables based on a related column to retrieve data from multiple sources in a single result set.
common_table_expressions	Common Table Expressions (CTEs) are named temporary result sets that simplify complex queries by allowing the definition of subqueries for better readability and reusability.
window_functions	Window Functions perform calculations across a specified range of rows related to the current row, offering advanced analytics and aggregation within result sets.
subqueries	Subqueries are nested queries that are embedded within a larger query to retrieve data, perform calculations, or filter results based on the outcome of the inner query.
null_handling	Null handling involves managing and evaluating the presence of null values, representing unknown or undefined data, to avoid unexpected results in queries and expressions.

12.1.2 SQL Statements

The **SQL Statements** page describes the following commands:

- *Data Definition Commands (DDL)*
- *Data Manipulation Commands (DML)*
- *Utility Commands*
- *Workload Management*
- *Access Control Commands*

SQream supports commands from ANSI SQL.

12.1.2.1 Data Definition Commands (DDL)

The following table shows the Data Definition commands:

Command	Usage
ADD COLUMN	Add a new column to a table
ALTER DEFAULT SCHEMA	Change the default schema for a role
ALTER TABLE	Change the schema of a table
CLUSTER BY	Change clustering keys in a table
CREATE DATABASE	Create a new database
CREATE FOREIGN TABLE	Create a new foreign table in the database
CREATE FUNCTION	Create a new user defined function in the database
CREATE SCHEMA	Create a new schema in the database
CREATE TABLE	Create a new table in the database
CREATE TABLE AS	Create a new table in the database using results from a select query
CREATE VIEW	Create a new view in the database
DROP CLUSTERING KEY	Drops all clustering keys in a table
DROP COLUMN	Drop a column from a table
DROP DATABASE	Drop a database and all of its objects
DROP FUNCTION	Drop a function
DROP SCHEMA	Drop a schema
DROP TABLE	Drop a table and its contents from a database
DROP VIEW	Drop a view
RENAME COLUMN	Rename a column
RENAME TABLE	Rename a table
RENAME SCHEMA	Rename a schema

12.1.2.2 Data Manipulation Commands (DML)

The following table shows the Data Manipulation commands:

Command	Usage
CREATE TABLE AS	Create a new table in the database using results from a select query
DELETE	Delete specific rows from a table
COPY FROM	Bulk load CSV data into an existing table
COPY TO	Export a select query or entire table to CSV files
INSERT	Insert rows into a table
SELECT	Select rows and column from a table
TRUNCATE	Delete all rows from a table
UPDATE	Modify the value of certain columns in existing rows without creating a table
VALUES	Return rows containing literal values

12.1.2.3 Utility Commands

The following table shows the Utility commands:

Command	Usage
GET TOTAL CHUNKS SIZE	Returns the total size of all data chunks saved in the system
DROP SAVED QUERY	Drops a saved query
EXECUTE SAVED QUERY	Executes a previously saved query
EXPLAIN	Returns a static query plan, which can be used to debug query plans
LIST SAVED QUERIES	Lists previously saved query names, one per row.
RECOMPILE SAVED QUERY	Recompiles a saved query that has been invalidated due to a schema change
SELECT GET LICENSE INFO	View a user's license information
SELECT GET DDL	View the `CREATE TABLE` statement for a table
SELECT GET FUNCTION DDL	View the `CREATE FUNCTION` statement for a UDF
SELECT GET TOTAL CHUNKS SIZE	Shows the total size of all data chunks saved in the system in both compressed and uncompressed formats
SELECT GET VIEW DDL	View the `CREATE VIEW` statement for a view
SELECT DATA READ METRICS	Monitor license quota usage by reviewing monthly or daily data read usage
SELECT GPU METRICS	Monitor license quota usage by reviewing monthly or daily GPU usage
SELECT HEALTH CHECK MONITORING	Returns system health monitoring logs
SELECT RECOMPILE VIEW	Recreate a view after schema changes
SELECT DUMP DATABASE DDL	View the `CREATE TABLE` statement for an current database
SHOW CONNECTIONS	Returns a list of active sessions on the current worker
SHOW LOCKS	Returns a list of locks from across the cluster
SHOW NODE INFO	Returns a snapshot of the current query plan, similar to `EXPLAIN ANALYZE` from other databases
SHOW SAVED QUERY	Returns a single row result containing the saved query string
SHOW SERVER STATUS	Returns a list of active sessions across the cluster
SHOW VERSION	Returns the system version for SQream DB
SHUTDOWN SERVER	Sets your server to finish compiling all active queries before shutting down according to a user-defined time value
STOP STATEMENT	Stops or aborts an active statement

12.1.2.4 Workload Management

The following table shows the Workload Management commands:

Command	Usage
subscribe_service	Add a SQream DB worker to a service queue
unsubscribe_service	Remove a SQream DB worker from a service queue
show_subscribed_instances	Return a list of service queues and workers

12.1.2.5 Access Control Commands

The following table shows the Access Control commands:

Command	Usage
alter_default_permissions	Applies a change to defaults in the current schema
alter_role	Applies a change to an existing role
create_role	Creates a roles, which lets a database administrator control permissions on tables and databases
drop_role	Removes roles
get_all_roles_database_ddl	Returns the definition of all role databases in DDL format
get_role_permissions	Returns all permissions granted to a role in table format
get_role_global_ddl	Returns the definition of a global role in DDL format
get_all_roles_global_ddl	Returns the definition of all global roles in DDL format
get_role_database_ddl	Returns the definition of a role's database in DDL format
get_statement_permissions	Returns a list of permissions required to run a statement or query
grant	Grant permissions to a role
GRANT USAGE ON SERVICE TO ALL ROLES	Grant service usage permissions
revoke	Revoke permissions from a role
rename_role	Rename a role

12.1.3 SQL Functions

SQream supports functions from ANSI SQL, as well as others for compatibility.

12.1.3.1 Summary of Functions

- *Built-In Scalar Functions*
  - *Bitwise Operations*
  - *Conditionals*
  - *Conversion*
  - *Date and Time*
  - *Numeric*

- *Strings*
- *User-Defined Scalar Functions*
- *Aggregate Functions*
- *Window Functions*
- *Workload Management Functions*

12.1.3.1.1 Built-In Scalar Functions

For more information about built-in scalar functions, see *Built-In Scalar Functions*.

12.1.3.1.1.1 Bitwise Operations

The following table shows the **bitwise operations** functions:

Function	Description
bitwise_and	Bitwise AND
bitwise_not	Bitwise NOT
bitwise_or	Bitwise OR
bitwise_shift_left	Bitwise shift left
bitwise_shift_right	Bitwise shift right
bitwise_xor	Bitwise XOR

12.1.3.1.1.2 Conditionals

The following table shows the **conditionals** functions:

Function	Description
between	Value is in [or not within] the range
case	Test a conditional expression, and depending on the result, evaluate additional expressions.
coalesce	Evaluate first non-NULL expression
in	Value is in [or not within] a set of values
isnull	Alias for coalesce with two expressions
is_ascii	Test a `TEXT` for ASCII-only characters
is_null	Check for `NULL` [or non-`NULL`] values

12.1.3.1.1.3 Conversion

The following table shows the **conversion** functions:

Function	Description
from_unixts	Converts a UNIX Timestamp to `DATE` or `DATETIME`
to_hex	Converts a number to a hexadecimal string representation
to_unixts	Converts a `DATE` or `DATETIME` to a UNIX Timestamp
chr	Returns the ASCII character representation of the supplied integer
is_castable	Checks whether a cast operation is possible or supported for a given column and data type and provides an alternative when there is an exception

12.1.3.1.1.4 Date and Time

The following table shows the **date and time** functions:

Function	Description
curdate	Special syntax, equivalent to current_date
current_date	Returns the current date as `DATE`
current_timestamp	Equivalent to getdate
datepart	Extracts a date or time element from a date expression
dateadd	Adds an interval to a date expression
datediff	Calculates the time difference between two date expressions
eomonth	Calculates the last day of the month of a given date expression
extract	ANSI syntax for extracting date or time element from a date expression
getdate	Returns the current timestamp as `DATETIME`
sysdate	Equivalent to getdate
date_trunc	Truncates a date element down to a specified date or time element

12.1.3.1.1.5 Numeric

The following table shows the **arithmetic operators**:

Table 1: Arithmetic Operators

Operator	Syntax	Description
+ (unary)	`+a`	Converts a string to a numeric value. Identical to `a :: double`
+	`a + b`	Adds two expressions together
− (unary)	`−a`	Negates a numeric expression
−	`a − b`	Subtracts `b` from `a`
*	`a * b`	Multiplies `a` by `b`
/	`a / b`	Divides `a` by `b`
%	`a % b`	Modulu of `a` by `b`. See also mod

For more information about arithmetic operators, see arithmetic_operators.

The following table shows the **arithmetic operator** functions:

Table 2: Arithemtic Operator Functions

Function	Description
abs	Calculates the absolute value of an argument
acos	Calculates the inverse cosine of an argument
asin	Calculates the inverse sine of an argument
atan	Calculates the inverse tangent of an argument
atn2	Calculates the inverse tangent for a point (y, x)
ceiling	Calculates the next integer for an argument
cos	Calculates the cosine of an argument
cot	Calculates the cotangent of an argument
degrees	Converts a value from radian values to degrees
exp	Calcalates the natural exponent for an argument (e^x)
floor	Calculates the largest integer smaller than the argument
log	Calculates the natural log for an argument
log10	Calculates the 10-based log for an argument
mod	Calculates the modulu (remainder) of two arguments
pi	Returns the constant value for π
power	Calculates x to the power of y (x^y)
radians	Converts a value from degree values to radians
round	Rounds an argument down to the nearest integer, or an arbitrary precision
sin	Calculates the sine of an argument
sqrt	Calculates the square root of an argument (√x)
square	Raises an argument to the power of 2 (x^y)
tan	Calculates the tangent of an argument
trunc	Rounds a number to its integer representation towards 0

12.1.3.1.1.6 Strings

The following table shows the **string** functions:

Function	Description
char_length	Calculates number of characters in an argument
charindex	Calculates the position where a string starts inside another string
concat	Concatenates two strings
crc64	Calculates a CRC-64 hash of an argument
decode	Decodes or extracts binary data from a textual input string
isprefixof	Matches if a string is the prefix of another string
left	Returns the first number of characters from an argument
len	Calculates the length of a string in characters
like	Tests if a string argument matches a pattern
lower	Converts an argument to a lower-case equivalent
ltrim	Trims whitespaces from the left side of an argument
octet_length	Calculates the length of a string in bytes
patindex	Calculates the position where a pattern matches a string
regexp_count	Calculates the number of matches of a regular expression match in an argument
regexp_instr	Returns the start position of a regular expression match in an argument
regexp_replace	Replaces and returns the text column substrings of a regular expression match in an argument
regexp_substr	Returns a substring of an argument that matches a regular expression
repeat	Repeats a string as many times as specified
replace	Replaces characters in a string
reverse	Reverses a string argument
right	Returns the last number of characters from an argument
rlike	Tests if a string argument matches a regular expression pattern
rtrim	Trims whitespace from the right side of an argument
substring	Returns a substring of an argument
trim	Trims whitespaces from an argument
upper	Converts an argument to an upper-case equivalent
select_ascii	Returns an `INT` value representing the ASCII code of the leftmost character in a string

12.1.3.1.2 User-Defined Scalar Functions

For more information about user-defined scalar functions, see scalar_sql_udf.

12.1.3.1.3 Aggregate Functions

The following table shows the **aggregate** functions:

Function	Aliases	Description
avg		Calculates the average of all of the values
corr		Calculates the Pearson correlation coefficient
count		Calculates the count of all of the values or only distinct values
covar_pop		Calculates population covariance of values
covar_samp		Calculates sample covariance of values
max		Returns maximum value of all values
min		Returns minimum value of all values
sum		Calculates the sum of all of the values or only distinct values
stddev_samp	`stdev`, `stddev`	Calculates sample standard deviation of values
stddev_pop	`stdevp`	Calculates population standard deviation of values
var_samp	`var`, `variance`	Calculates sample variance of values
var_pop	`varp`	Calculates population variance of values

For more information about aggregate functions, see *Aggregate Functions*.

12.1.3.1.4 Window Functions

The following table shows the **window** functions:

Function	Description
lag	Calculates the value evaluated at the row that is before the current row within the partition
lead	Calculates the value evaluated at the row that is after the current row within the partition
max	Calculates the maximum value
min	Calculates the minimum value
sum	Calculates the sum of all of the values
rank	Calculates the rank of a row
first_value	Returns the value in the first row of a window
last_value	Returns the value in the last row of a window
nth_value	Returns the value in a specified `(n)` row of a window
dense_rank	Returns the rank of the current row with no gaps
percent_rank	Returns the relative rank of the current row
cume_dist	Returns the cumulative distribution of rows
ntile	Returns an integer ranging between `1` and the argument value, dividing the partitions as equally as possible

For more information about window functions, see window_functions.

12.1.3.1.5 Workload Management Functions

The following table shows the **workload management** functions:

Function	Description
subscribe_service	Add a SQream DB worker to a service queue
unsubscribe_service	Remove a SQream DB worker to a service queue
show_subscribed_instances	Return a list of service queues and workers

12.1.3.1.5.1 Built-In Scalar Functions

The **Built-In Scalar Functions** page describes functions that return one value per call:

- bitwise_and
- bitwise_not
- bitwise_or
- bitwise_shift_left
- bitwise_shift_right
- bitwise_xor
- between
- case
- coalesce
- decode
- in
- is_ascii
- is_null
- isnull
- from_unixts
- to_hex
- to_unixts
- curdate
- current_date
- current_timestamp
- dateadd
- datediff
- datepart
- eomonth
- extract
- getdate
- sysdate
- trunc
- abs
- acos
- asin
- atan
- atn2
- ceiling
- cos
- cot
- crc64
- degrees
- exp
- floor
- log
- log10
- mod
- pi
- power
- radians
- round
- sin
- sqrt
- square
- tan
- trunc
- char_length
- charindex
- concat
- isprefixof
- left
- len
- like
- lower
- ltrim
- octet_length
- patindex
- regexp_count
- regexp_instr
- regexp_replace
- regexp_substr
- repeat
- replace
- reverse
- right
- rlike
- rtrim
- substring
- trim
- upper
- select_ascii
- sign
- chr

12.1.3.1.5.2 User-Defined Functions

The following user-defined functions are functions that can be defined and configured by users.

The **User-Defined Functions** page describes the following:

- *Python user-defined functions*
- Scalar SQL user-defined functions
- Simple Scalar SQL UDF's

12.1.3.1.5.3 Aggregate Functions

12.1.3.1.5.4 Overview

Aggregate functions perform calculations based on a set of values and return a single value. Most aggregate functions ignore null values. Aggregate functions are often used with the `GROUP BY` clause of the select statement.

12.1.3.1.5.5 Available Aggregate Functions

The following list shows the available aggregate functions:

- AVG
- CORR
- COUNT
- COVAR_POP
- COVAR_SAMP
- MAX
- MIN
- MODE
- PERCENTILE_CONT
- PERCENTILE_DISC
- STDDEV_POP
- STDDEV_SAMP
- SUM
- VAR_POP
- VAR_SAMP

12.1.3.1.5.6 Window Functions

Window functions are functions applied over a subset (known as a window) of the rows returned by a select query and describes the following:

- lag
- lead
- row_number
- rank
- first_value
- last_value
- nth_value
- dense_rank
- percent_rank
- cume_dist
- ntile

For more information, see window_functions in the *SQL Syntax Features* section.

12.2 Catalog Reference

The SQreamDB database uses a schema called `sqream_catalog` that contains information about database objects such as tables, columns, views, and permissions. Some additional catalog tables are used primarily for internal analysis and may differ across SQreamDB versions.

12.2.1 What Information Does the Schema Contain?

The schema contains data management tables with information about structure and management of database elements, including tables, schemas, queries, and permissions, and physical storage and organization of data tables of extents, chunk columns, chunks, and delete predicates.

12.2.2 How to Get Table Information?

To get the information stored on a table, use this syntax, as in this example of working with the `parameters` table:

```
SELECT * FROM sqream_catalog.parameters;
```

To get the table ddl, use this syntax, as in this example of working with the `parameters` table:

```
SELECT get_ddl('sqream_catalog.parameters');
```

12.2.2.1 Database Management Tables

Database Object	Table
Clustering Keys	`clustering_keys`
Columns	`columns`, `external_table_columns`
Databases	`databases`
Parameters	`parameters`
Permissions	`table_permissions`, `database_permissions`, `schema_permissions`, `permission_types`, `udf_permissions`, `sqream_catalog.table_default_permissions`
Queries	`savedqueries`
Roles	`roles`, `role_memberships`
Schemas	`schemas`
Tables	`tables`, `external_tables`
Views	`views`
User Defined Functions	`user_defined_functions`

12.2.2.2 Data Storage and Organization Tables

Database Object	Table
Extents	Shows `extents`
Chunk columns	Shows `chunks_columns`
Chunks	Shows `chunks`
Delete predicates	Shows `delete_predicates`. For more information, see *Deleting Data*

12.2.2.2.1 Catalog Tables

The `sqream_catalog` includes the following tables:

- *Clustering Keys*
- *Columns*
- *Databases*
- *Parameters*
- *Permissions*
- *Queries*
- *Roles*
- *Schemas*
- *Tables*
- *Views*

- *User Defined Functions*

12.2.2.2.1.1 Clustering Keys

The `clustering_keys` data object is used for explicit clustering keys for tables. If you define more than one clustering key, each key is listed in a separate row, and is described in the following table:

Column	Description
`database_name`	Shows the name of the database containing the table.
`table_id`	Shows the ID of the table containing the column.
`schema_name`	Shows the name of the schema containing the table.
`table_name`	Shows the name of the table containing the column.
`clustering_key`	Shows the name of the column used as a clustering key for this table.

12.2.2.2.1.2 Columns

The **Columns** database object shows the following tables:

- *Columns*
- *External Table Columns*

12.2.2.2.1.3 Columns

The `column` data object is used with standard tables and is described in the following table:

Column	Description
database_name	Shows the name of the database containing the table.
schema_name	Shows the name of the schema containing the table.
table_id	Shows the ID of the table containing the column.
table_name	Shows the name of the table containing the column.
column_id	Shows the ordinal number of the column in the table (begins at **0**).
column_name	Shows the column's name.
type_name	Shows the column's data type. For more information see *Supported Data Types*.
column_size	Shows the maximum length in bytes.
has_default	Shows `NULL` if the column has no default value, `1` if the default is a fixed value, or `2` if the default is an identity. For more information, see identity.
default_value	Shows the column's default value. For more information, see Default Value Constraints.
compression_strategy	Shows the compression strategy that a user has overridden.
created	Shows the timestamp displaying when the column was created.
altered	Shows the timestamp displaying when the column was last altered.

12.2.2.2.1.4 External Table Columns

The `external_table_columns` is used for viewing data from foreign tables.

For more information on foreign tables, see CREATE FOREIGN TABLE.

12.2.2.2.1.5 Databases

The `databases` data object is used for displaying database information, and is described in the following table:

Column	Description
database_id	Shows the database's unique ID.
database_name	Shows the database's name.
default_disk_chunk_size	Reserved for internal use.
default_process_chunk_size	Reserved for internal use.
rechunk_size	Reserved for internal use.
storage_subchunk_size	Reserved for internal use.
compression_chunk_size_threshold	Reserved for internal use.

12.2.2.2.1.6 Parameters

The `parameters` object is used for displaying all flags, providing the scope (default, cluster and session), description, default value and actual value.

Column	Description
`flag_name`	Shows the flag name
`value`	Shows the current flag configured value
`default_value`	Shows the flag default value
`scope`	Shows whether flag configuration is session-based or cluster-based
`description`	Describes the purpose of the flag

12.2.2.2.1.7 Permissions

The `permissions` data object is used for displaying permission information, such as roles (also known as **grantees**), and is described in the following tables:

- *Permission Types*
- *Default Permissions*
- *Table Permissions*
- *Database Permissions*
- *Schema Permissions*

12.2.2.2.1.8 Permission Types

The `permission_types` object identifies the permission names existing in the database.

Column	Description
`permission_type_id`	Shows the permission type's ID.
`name`	Shows the name of the permission type.

12.2.2.2.1.9 Default Permissions

The commands included in the **Default Permissions** section describe how to check the following default permissions:

- *Default Table Permissions*
- *Default Schema Permissions*

12.2.2.2.1.10 Default Table Permissions

The `sqream_catalog.table_default_permissions` command shows the columns described below:

Column	Description
`database_name`	Shows the database that the default permission rule applies to.
`schema_id`	Shows the schema that the rule applies to, or `NULL` if the `ALTER` statement does not specify a schema.
`modifier_role_id`	Shows the role to apply the rule to.
`getter_role_id`	Shows the role that the permission is granted to.
`permission_type`	Shows the type of permission granted.

12.2.2.2.1.11 Default Schema Permissions

The `sqream_catalog.schema_default_permissions` command shows the columns described below:

Column	Description
`database_name`	Shows the database that the default permission rule applies to.
`modifier_role_id`	Shows the role to apply the rule to.
`getter_role_id`	Shows the role that the permission is granted to.
`permission_type`	Shows the type of permission granted.
`getter_role_type`	Shows the type of role that is granted permissions.

For an example of using the `sqream_catalog.table_default_permissions` command, see Granting Default Table Permissions.

12.2.2.2.1.12 Table Permissions

The `table_permissions` data object identifies all permissions granted to tables. Each role-permission combination displays one row.

The following table describes the `table_permissions` data object:

Column	Description
`database_name`	Shows the name of the database containing the table.
`table_id`	Shows the ID of the table the permission applies to.
`role_id`	Shows the ID of the role granted permissions.
`permission_type`	Identifies the permission type.

12.2.2.2.1.13 Database Permissions

The `database_permissions` data object identifies all permissions granted to databases. Each role-permission combination displays one row.

The following table describes the `database_permissions` data object:

Column	Description
`database_name`	Shows the name of the database the permission applies to
`role_id`	Shows the ID of the role granted permissions.
`permission_type`	Identifies the permission type.

12.2.2.2.1.14 Schema Permissions

The `schema_permissions` data object identifies all permissions granted to schemas. Each role-permission combination displays one row.

The following table describes the `schema_permissions` data object:

Column	Description
`database_name`	Shows the name of the database containing the schema.
`schema_id`	Shows the ID of the schema the permission applies to.
`role_id`	Shows the ID of the role granted permissions.
`permission_type`	Identifies the permission type.

12.2.2.2.1.15 Queries

The `savedqueries` data object identifies the saved queries in the database, as shown in the following table:

Column	Description
`name`	Shows the saved query name.
`num_parameters`	Shows the number of parameters to be replaced at run-time.

For more information, see *Saved Queries*.

12.2.2.2.1.16 Roles

The `roles` data object is used for displaying role information, and is described in the following tables:

- *Roles*
- *Role Memberships*

12.2.2.2.1.17 Roles

The `roles` data object identifies the roles in the database, as shown in the following table:

Column	Description
`role_id`	Shows the role's database-unique ID.
`name`	Shows the role's name.
`superuser`	Identifies whether the role is a superuser (`1` - superuser, `0` - regular user).
`login`	Identifies whether the role can be used to log in to SQream (`1` - yes, `0` - no).
`has_password`	Identifies whether the role has a password (`1` - yes, `0` - no).

12.2.2.2.1.18 Role Memberships

The `roles_memberships` data object identifies the role memberships in the database, as shown below:

Column	Description
`role_id`	Shows the role ID.
`member_role_id`	Shows the ID of the parent role that this role inherits from.
`inherit`	Identifies whether permissions are inherited (`1` - yes, `0` - no).
`admin`	Identifies whether role is admin (`1` - yes, `0` - no).

12.2.2.2.1.19 Schemas

The `schemas` data object identifies all the database's schemas, as shown below:

Column	Description
`schema_id`	Shows the schema's unique ID.
`schema_name`	Shows the schema's name.
`schema_owner`	Shows the name of the role that owns the schema.
`rechunker_ignore`	Reserved for internal use.

12.2.2.2.1.20 Tables

The `tables` data object is used for displaying table information, and is described in the following tables:

- *Tables*
- *Foreign Tables*

12.2.2.2.1.21 Tables

The `tables` data object identifies proper (**Comment** - *What does "proper" mean?*) SQream tables in the database, as shown in the following table:

Column	Description
`database_name`	Shows the name of the database containing the table.
`table_id`	Shows the table's database-unique ID.
`schema_name`	Shows the name of the schema containing the table.
`table_name`	Shows the name of the table.
`row_count_is_valid`	Identifies whether the `row_count` can be used.
`row_count`	Shows the number of rows in the table.
`rechunker_ignore`	Relevant for internal use.

12.2.2.2.1.22 Foreign Tables

The `external_tables` data object identifies foreign tables in the database, as shown below:

Column	Description
`database_name`	Shows the name of the database containing the table.
`table_id`	Shows the table's database-unique ID.
`schema_name`	Shows the name of the schema containing the table.
`table_name`	Shows the name of the table.
`format`	Identifies the foreign data wrapper used. `0` for `csv_fdw`, `1` for `parquet_fdw`, `2` for `orc_fdw`.
`created`	Identifies the clause used to create the table.

12.2.2.2.1.23 Views

The `views` data object is used for displaying views in the database, as shown below:

Column	Description
`view_id`	Shows the view's database-unique ID.
`view_schema`	Shows the name of the schema containing the view.
`view_name`	Shows the name of the view.
`view_data`	Reserved for internal use.
`view_query_text`	Identifies the `AS` clause used to create the view.

12.2.2.2.1.24 User Defined Functions

The `udf` data object is used for displaying UDFs in the database, as shown below:

Column	Description
`database_name`	Shows the name of the database containing the view.
`function_id`	Shows the UDF's database-unique ID.
`function_name`	Shows the name of the UDF.

12.2.2.2.2 Additional Tables

The Reference Catalog includes additional tables that can be used for performance monitoring and inspection. The definition for these tables described on this page may change across SQream versions.

- *Extents*
- *Chunk Columns*
- *Chunks*
- *Delete Predicates*

12.2.2.2.2.1 Extents

The `extents` storage object identifies storage extents, and each storage extents can contain several chunks.

Note: This is an internal table designed for low-level performance troubleshooting.

Column	Description
`database_name`	Shows the name of the databse containing the extent.
`table_id`	Shows the ID of the table containing the extent.
`column_id`	Shows the ID of the column containing the extent.
`extent_id`	Shows the ID for the extent.
`size`	Shows the extent size in megabytes.
`path`	Shows the full path to the extent on the file system.

12.2.2.2.2.2 Chunk Columns

The `chunk_columns` storage object lists chunk information by column.

Column	Description
`database_name`	Shows the name of the databse containing the extent.
`table_id`	Shows the ID of the table containing the extent.
`column_id`	Shows the ID of the column containing the extent.
`chunk_id`	Shows the chunk ID.
`extent_id`	Shows the extent ID.
`compressed_size`	Shows the compressed chunk size in bytes.
`uncompressed_size`	Shows the uncompressed chunk size in bytes.
`compression_type`	Shows the chunk's actual compression scheme.
`long_min`	Shows the minimum numeric value in the chunk (if one exists).
`long_max`	Shows the maximum numeric value in the chunk (if one exists).
`string_min`	Shows the minimum text value in the chunk (if one exists).
`string_max`	Shows the maximum text value in the chunk (if one exists).
`offset_in_file`	Reserved for internal use.

Note: This is an internal table designed for low-level performance troubleshooting.

12.2.2.2.2.3 Chunks

The `chunks` storage object identifies storage chunks.

Column	Description
`database_name`	Shows the name of the databse containing the chunk.
`table_id`	Shows the ID of the table containing the chunk.
`column_id`	Shows the ID of the column containing the chunk.
`rows_num`	Shows the amount of rows in the chunk.
`deletion_status`	Determines what data to logically delete from the table first, and identifies how much data to delete from the chunk. The value `0` is ued for no data, `1` for some data, and `2` to delete the entire chunk.

Note: This is an internal table designed for low-level performance troubleshooting.

12.2.2.2.2.4 Delete Predicates

The `delete_predicates` storage object identifies the existing delete predicates that have not been cleaned up.

Each DELETE command may result in several entries in this table.

Column	Description
`database_name`	Shows the name of the databse containing the predicate.
`table_id`	Shows the ID of the table containing the predicate.
`max_chunk_id`	Reserved for internal use, this is a placeholder marker for the highest `chunk_id` logged during the `DELETE` operation.
`delete_predicate`	Identifies the DELETE predicate.

Note: This is an internal table designed for low-level performance troubleshooting.

12.2.2.2.3 Examples

- *Listing All Tables in a Database*
- *Listing All Schemas in a Database*
- *Listing Columns and Their Types for a Specific Table*
- *Listing Delete Predicates*
- *Listing Saved Queries*

12.2.2.2.3.1 Listing All Tables in a Database

```
master=> SELECT * FROM sqream_catalog.tables;
database_name | table_id | schema_name | table_name     | row_count_valid | row_count␣
↪| rechunker_ignore
--------------+----------+-------------+----------------+-----------------+-----------
↪+-----------------
master        |        1 | public      | nba            | true            |       457␣
↪|                0
master        |       12 | public      | cool_dates     | true            |         5␣
↪|                0
master        |       13 | public      | cool_numbers   | true            |         9␣
↪|                0
master        |       27 | public      | jabberwocky    | true            |         8␣
↪|                0
```

12.2.2.2.3.2 Listing All Schemas in a Database

```
master=> SELECT * FROM sqream_catalog.schemas;
schema_id | schema_name    | rechunker_ignore
----------+----------------+-----------------
        0 | public         | false
        1 | secret_schema  | false
```

12.2.2.2.3.3 Listing Columns and Their Types for a Specific Table

```
SELECT column_name, type_name
FROM sqream_catalog.columns
WHERE table_name='cool_animals';
```

12.2.2.2.3.4 Listing Delete Predicates

```
SELECT  t.table_name, d.*  FROM
sqream_catalog.delete_predicates AS d
INNER JOIN sqream_catalog.tables AS t
ON d.table_id=t.table_id;
```

12.2.2.2.3.5 Listing Saved Queries

```
SELECT * FROM sqream_catalog.savedqueries;
```

12.3 Command line programs

SQream contains several command line programs for using, starting, managing, and configuring SQream DB clusters.

This topic contains the reference for these programs, as well as flags and configuration settings.

Table 3: User CLIs

Command	Usage
sqream sql	Built-in SQL client

Table 4: SQream DB cluster components

Command	Usage
sqreamd	Start a SQream DB worker
metadata_server	The cluster manager/coordinator that enables scaling SQream DB.
server_picker	Load balancer end-point

Table 5: SQream DB utilities

Command	Usage
SqreamStorage	Initialize a cluster and set superusers
upgrade_storage	Upgrade metadata schemas when upgrading between major versions

12.3.1 metadata_server

SQream DB's cluster manager/coordinator is called `metadata_server`.

In general, you should not need to run `metadata_server` manually, but it is sometimes useful for testing.

12.3.1.1 Command Line Arguments

Argument	Default	Description
`--config`	`/home/omert/.sqream/metadata_server_config.json`	The configuration file to use
`--port`	`3105`	The metadata server listening port
`--log_path`	`./metadata_server_log`	The `metadata_server` log file output contains information about the activities and events related to the metadata server of a system.
`--log4_config`	None	Specifies the location of the configuration file for the `Log4cxx` logging library.
`--num_deleters`	1	Specifies the number of threads to use for the file reaper in a system or program.
`--metadata_path`	`<...sqreamd/leveldb>`	Specifies the path to the directory where metadata files are stored for a system or program.
`--help`	None	Used to display a help message or documentation for a particular program or command.

12.3.1.2 Starting metadata server

12.3.1.2.1 Starting temporarily

```
nohup metadata_server -config ~/.sqream/metadata_server_config.json &
MS_PID=$!
```

Using `nohup` and `&` sends metadata server to run in the background.

Note:

- Logs are saved to the current directory, under `metadata_server_logs`.
- The default listening port is 3105

12.3.1.2.2 Starting temporarily with non-default port

To use a non-default port, specify the logging path as well.

```
nohup metadata_server --log_path=/home/rhendricks/metadata_logs --port=9241 &
MS_PID=$!
```

Using `nohup` and `&` sends metadata server to run in the background.

Note:

- Logs are saved to the `/home/rhendricks/metadata_logs` directory.
- The listening port is 9241

12.3.1.2.3 Stopping metadata server

To stop metadata server:

```
kill -9 $MS_PID
```

Tip: It is safe to stop any SQream DB component at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

12.3.2 sqreamd

SQream DB's main worker is called `sqreamd`.

This page serves as a reference for the options and parameters.

12.3.2.1 Starting SQream DB

12.3.2.1.1 Start SQream DB temporarily

In general, you should not need to run `sqreamd` manually, but it is sometimes useful for testing.

```
$ nohup sqreamd -config ~/.sqream/sqream_config.json &
$ SQREAM_PID=$!
```

Using `nohup` and `&` sends SQream DB to run in the background.

To stop the active worker:

```
$ kill -9 $SQREAM_PID
```

Tip: It is safe to stop SQream DB at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

12.3.2.2 Command line arguments

`sqreamd` supports the following command line arguments:

Argument	Default	Description
`--version`	None	Outputs the version of SQream DB and immediately exits.
`-config`	`$HOME/.sqream/sqream_config.json`	Specifies the configuration file to use
`--port_ssl`	Don't use SSL	When specified, tells SQream DB to listen for SSL connections

12.3.2.2.1 Positional command arguments

`sqreamd` also supports positional arguments, when not using a configuration file.

This method can be used to temporarily start a SQream DB worker for testing.

```
$ sqreamd <Storage path> <GPU ordinal> <TCP listen port (unsecured)> <License path>
```

Argument	Required	Description
Storage path	✓	Full path to a valid SQream DB persistant storage
GPU Ordinal	✓	Number representing the GPU to use. Check GPU ordinals with *nvidia-smi -L*
TCP listen port (unsecured)	✓	TCP port SQream DB should listen on. Recommended: 5000
License path	✓	Full path to a SQream DB license file

12.3.3 sqream Linux console

`sqream-console` is an interactive shell designed to help manage a dockerized SQream DB installation.

The console itself is a dockerized application.

- *Starting the console*
- *Operations and flag reference*
- *Using the console to start SQream DB*

12.3.3.1 Starting the console

`sqream-console` can be found in your SQream DB installation, under the name `sqream-console`.

Start the console by executing it from the shell

```
$ ./sqream-console
```

(continues on next page)

(continued from previous page)

```
↪.............................


Welcome to SQream Console ver 1.7.6, type exit to log-out

usage: sqream [-h] [--settings] {master,worker,client,editor} ...

Run SQream Cluster

optional arguments:
  -h, --help            show this help message and exit
  --settings            sqream environment variables settings

subcommands:
  sqream services

  {master,worker,client,editor}
                        sub-command help
    master              start sqream master
    worker              start sqream worker
    client              operating sqream client
    editor              operating sqream statement editor
sqream-console>
```

The console is now waiting for commands.

The console is a wrapper around a standard linux shell. It supports commands like `ls`, `cp`, etc.

All SQream DB-specific commands start with the keyword `sqream`.

12.3.3.2 Operations and flag reference

12.3.3.2.1 Commands

Command	Description
`sqream --help`	Shows the initial usage information
`sqream master`	Controls the master node's operations
`sqream worker`	Controls workers' operations
`sqream client`	Access to *sqream sql*
`sqream editor`	Controls the statement editor's operations (web UI)

12.3.3.2.2 Master

The master node contains the *metadata server* and the *load balancer*.

12.3.3.2.2.1 Syntax

```
sqream master <flags>
```

Flag/command	Description
`--start [ --single-host ]`	Starts the master node. The `--single-host` modifier sets the mode to allow all containers to run on the same server.
`--stop [ --all ]`	Stops the master node and all connected *workers*. The `--all` modifier instructs the `--stop` command to stop all running services related to SQream DB
`--list`	Shows a list of all active master nodes and their workers
`-p <port>`	Sets the port for the load balancer. Defaults to `3108`
`-m <port>`	Sets the port for the metadata server. Defaults to `3105`

12.3.3.2.2.2 Common usage

12.3.3.2.2.3 Start master node

```
sqream-console> sqream master --start
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   3105,3108
```

12.3.3.2.2.4 Start master node on different ports

```
sqream-console> sqream master --start -p 4105 -m 4108
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   4105,4108
```

12.3.3.2.2.5 Listing active master nodes and workers

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

12.3.3.2.2.6 Stopping all SQream DB workers and master

```
sqream-console> sqream master --stop --all
  shutting down 2 sqream services ...
 sqream_editor    stopped
 sqream_single_host_worker_1    stopped
 sqream_single_host_worker_0    stopped
 sqream_single_host_master    stopped
```

12.3.3.2.3 Workers

Workers are *SQream DB daemons*, that connect to the master node.

12.3.3.2.3.1 Syntax

```
sqream worker <flags>
```

Flag/command	Description
`--start [ options [ ... ] ]`	Starts worker nodes. See options table below.
`--stop [ <worker name> | --all ]`	Stops the specified worker name. The `--all` modifier instructs the `--stop` command to stop all running workers.

Start options are specified consecutively, separated by spaces.

Table 6: Start options

Option	Description
`<n>`	Specifies the number of workers to start
`-j <config file> [ ...]`	Specifies configuration files to apply to each worker. When launching multiple workers, specify one file per worker, separated by spaces.
`-p <port> [ ...]`	Sets the ports to listen on. When launching multiple workers, specify one port per worker, separated by spaces. Defaults to 5000 - 5000+n.
`-g <gpu id> [ ...]`	Sets the GPU ordinal to assign to each worker. When launching multiple workers, specify one GPU ordinal per worker, separated by spaces. Defaults to automatic allocation.
`-m <spool memory>`	Sets the spool memory per node in gigabytes.
`--master-host`	Sets the hostname for the master node. Defaults to `localhost`.
`--master-port`	Sets the port for the master node. Defaults to `3105`.
`--stand-alone`	For testing only: Starts a worker without connecting to the master node.

12.3.3.2.3.2 Common usage

12.3.3.2.3.3 Start 2 workers

After starting the master node, start workers:

```
sqream-console> sqream worker --start 2
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 1
```

12.3.3.2.3.4 Stop a single worker

To stop a single worker, find its name first:

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

Then, issue a stop command:

```
sqream-console> sqream worker --stop sqream_single_host_worker_1
stopped sqream_single_host_worker_1
```

12.3.3.2.3.5 Start workers with a different spool size

If no spool size is specified, the RAM is equally distributed among workers. Sometimes a system engineer may wish to specify the spool size manually.

This example starts two workers, with a spool size of 50GB per node:

```
sqream-console> sqream worker --start 2 -m 50
```

12.3.3.2.3.6 Starting multiple workers on non-dedicated GPUs

By default, SQream DB workers assign one worker per GPU. However, a system engineer may wish to assign multiple workers per GPU, if the workload permits it.

This example starts 4 workers on 2 GPUs, with 50GB spool each:

```
sqream-console> sqream worker --start 2 -g 0 -m 50
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 0
sqream-console> sqream worker --start 2 -g 1 -m 50
started sqream_single_host_worker_2 on port 5002, allocated gpu: 1
started sqream_single_host_worker_3 on port 5003, allocated gpu: 1
```

12.3.3.2.3.7 Overriding default configuration files

It is possible to override default configuration settings by listing a configuration file for every worker.

This example starts 2 workers on the same GPU, with modified configuration files:

```
sqream-console> sqream worker --start 2 -g 0 -j /etc/sqream/configfile.json /etc/
↪sqream/configfile2.json
```

12.3.3.2.4 Client

The client operation runs *sqream sql* in interactive mode.

Note: The dockerized client is useful for testing and experimentation. It is not the recommended method for executing analytic queries. See more about connecting a third party tool to SQream DB for data analysis.

12.3.3.2.4.1 Syntax

```
sqream client <flags>
```

Flag/command	Description
`--master`	Connects to the master node via the load balancer
`--worker`	Connects to a worker directly
`--host <hostname>`	Specifies the hostname to connect to. Defaults to `localhost`.
`--port <port>`, `-p <port>`	Specifies the port to connect to. Defaults to `3108` when used with `-master`.
`--user <username>`, `-u <username>`	Specifies the role's username to use
`--password <password>`, `-w <password>`	Specifies the password to use for the role
`--database <database>`, `-d <database>`	Specifies the database name for the connection. Defaults to `master`.

12.3.3.2.4.2 Common usage

12.3.3.2.4.3 Start a client

Connect to default `master` database through the load balancer:

```
sqream-console> sqream client --master -u sqream -w sqream
Interactive client mode
To quit, use ^D or \q.

master=> _
```

12.3.3.2.4.4 Start a client to a specific worker

Connect to database `raviga` directly to a worker on port 5000:

```
sqream-console> sqream client --worker -u sqream -w sqream -p 5000 -d raviga
Interactive client mode
To quit, use ^D or \q.

raviga=> _
```

12.3.3.2.4.5 Start master node on different ports

```
sqream-console> sqream master --start -p 4105 -m 4108
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   4105,4108
```

12.3.3.2.4.6 Listing active master nodes and worker nodes

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

12.3.3.2.5 Editor

The editor operation runs the web UI for the SQream DB Statement Editor.

The editor can be used to run queries from a browser.

12.3.3.2.5.1 Syntax

```
sqream editor <flags>
```

Flag/command	Description
`--start`	Start the statement editor
`--stop`	Shut down the statement editor
`--port <port>`, `-p <port>`	Specify a different port for the editor. Defaults to `3000`.

12.3.3.2.5.2 Common usage

12.3.3.2.5.3 Start the editor UI

```
sqream-console> sqream editor --start
access sqream statement editor through Chrome http://192.168.0.100:3000
```

12.3.3.2.5.4 Stop the editor UI

```
sqream-console> sqream editor --stop
 sqream_editor    stopped
```

12.3.3.3 Using the console to start SQream DB

The console is used to start and stop SQream DB components in a dockerized environment.

12.3.3.3.1 Starting a SQream DB cluster for the first time

To start a SQream DB cluster, start the master node, followed by workers.

The example below starts 2 workers, running on 2 dedicated GPUs.

```
sqream-console> sqream master --start
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   3105,3108

sqream-console> sqream worker --start 2
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 1

sqream-console> sqream editor --start
access sqream statement editor through Chrome http://192.168.0.100:3000
```

SQream DB is now listening on port 3108 for any incoming statements.

A user can also access the web editor (running on port `3000` on the SQream DB machine) to connect and run queries.

12.3.4 Server Picker

SQreamDB's load balancer is called `server_picker`.

12.3.4.1 Command Line Arguments

12.3.4.1.1 Parameters

Argument	Default	Description
`--metadata_server_port`	`3105`	The metadata server listening port
`--metadata_server_ip`	`127.0.0.1`	The metadata server IP
`--port`	`3108`	The server picker port
`--ssl_port`	`3109`	The server picker ssl port
`--log4_config`	`/home/sqream/sqream3/etc/server_picker_log_properties`	The server picker log4 configuration file to use
`--refresh_interval`	`15`	Refresh interval time to check available nodes
`--services`	None	Lists services separated by comma
`--help`	None	Used to display a help message or documentation for a particular program or command

12.3.4.1.2 Example

```
server_picker --metadata_ip=127.0.0.1 --metadata_server_port=3105 --port=3118 --ssl_
↪port=3119 --services=sqream23,sqream0 --log4_config=/home/sqream/metadata_log_
↪properties --refresh_interval=10
```

12.3.4.2 Starting server picker

12.3.4.2.1 Starting temporarily

In general, you should not need to run `server_picker` manually, but it is sometimes useful for testing.

Assuming we have a *metadata server* listening on the localhost, on port 3105:

```
nohup server_picker --metadata_server_ip=127.0.0.1 metadata_server_port=3105 &
SP_PID=$!
```

Using `nohup` and `&` sends server picker to run in the background.

12.3.4.2.2 Starting temporarily with non-default port

Tell server picker to listen on port 2255 for unsecured connections, and port 2266 for SSL connections.

```
nohup server_picker --metadata_server_ip=127.0.0.1 --metadata_server_port=3105 --
↪port=2255 --ssl_port=2266 &
SP_PID=$!
```

Using `nohup` and `&` sends server picker to run in the background.

12.3.4.2.3 Stopping server picker

```
kill -9 $SP_PID
```

Tip: It is safe to stop any SQream DB component at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

12.3.5 SqreamStorage

You can use the **SqreamStorage** program to create a new *storage cluster*.

The **SqreamStorage** page serves as a reference for the options and parameters.

12.3.5.1 Running SqreamStorage

The **SqreamStorage** program is located in the **bin** directory of your SQream installation..

12.3.5.2 Command Line Arguments

The **SqreamStorage** program supports the following command line arguments:

Argument	Shorthand	Description
`--create-cluster`	`-C`	Creates a storage cluster at a specified path
`--cluster-root`	`-r`	Specifies the cluster path. The path must **not** already exist.

12.3.5.3 Example

The **Examples** section describes how to create a new storage cluster at `/home/rhendricks/raviga_database`:

```
$ SqreamStorage --create-cluster --cluster-root /home/rhendricks/raviga_database
Setting cluster version to: 26
```

Alternatively, you can write this in shorthand as `SqreamStorage -C -r /home/rhendricks/raviga_database`. A message is displayed confirming that your cluster has been created.

12.3.6 Sqream SQL CLI Reference

SQream DB comes with a built-in client for executing SQL statements either interactively or from the command-line.

This page serves as a reference for the options and parameters. Learn more about using SQream DB SQL with the CLI by visiting the first_steps tutorial.

In this topic:

- *Installing Sqream SQL*
  - *Troubleshooting Sqream SQL Installation*
- *Using Sqream SQL*
  - *Running Commands Interactively (SQL shell)*
  - *Executing Batch Scripts (`-f`)*
  - *Executing Commands Immediately (`-c`)*
- *Examples*
  - *Starting a Regular Interactive Shell*
  - *Executing Statements in an Interactive Shell*
  - *Executing SQL Statements from the Command Line*
  - *Controlling the Client Output*
    - *Exporting SQL Query Results to CSV*
    - *Changing a CSV to a TSV*

- *Executing a Series of Statements From a File*
- *Connecting Using Environment Variables*
- *Connecting to a Specific Queue*

- *Operations and Flag References*
  - *Command Line Arguments*
    - *Supported Record Delimiters*
  - *Meta-Commands*
  - *Basic Commands*
  - *Moving Around the Command Line*
  - *Searching*

12.3.6.1 Installing Sqream SQL

If you have a SQream DB installation on your server, `sqream sql` can be found in the `bin` directory of your SQream DB installation, under the name `sqream`.

Changed in version 2020.1: As of version 2020.1, `ClientCmd` has been renamed to `sqream sql`.

To run `sqream sql` on any other Linux host:

1. Download the `sqream sql` tarball package from the *Client Drivers* page.
2. Untar the package: `tar xf sqream-sql-v2020.1.1_stable.x86_64.tar.gz`
3. Start the client:

```
$ cd sqream-sql-v2020.1.1_stable.x86_64
$ ./sqream sql --port=5000 --username=jdoe --databasename=master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> _
```

12.3.6.1.1 Troubleshooting Sqream SQL Installation

Upon running sqream sql for the first time, you may get an error `error while loading shared libraries: libtinfo.so.5: cannot open shared object file: No such file or directory`.

Solving this error requires installing the ncruses or libtinfo libraries, depending on your operating system.

- Ubuntu:
  1. Install `libtinfo`:

     ```
     $ sudo apt-get install -y libtinfo
     ```

  2. Depending on your Ubuntu version, you may need to create a symbolic link to the newer libtinfo that was installed.

     For example, if `libtinfo` was installed as `/lib/x86_64-linux-gnu/libtinfo.so.6.2`:

```
$ sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6.2 /lib/x86_64-linux-gnu/libtinfo.so.5
```

- CentOS / RHEL:
  1. Install `ncurses`:

     ```
     $ sudo yum install -y ncurses-libs
     ```

  2. Depending on your RHEL version, you may need to create a symbolic link to the newer libtinfo that was installed.

     For example, if `libtinfo` was installed as `/usr/lib64/libtinfo.so.6`:

     ```
     $ sudo ln -s /usr/lib64/libtinfo.so.6 /usr/lib64/libtinfo.so.5
     ```

12.3.6.2 Using Sqream SQL

By default, sqream sql runs in interactive mode. You can issue commands or SQL statements.

12.3.6.2.1 Running Commands Interactively (SQL shell)

When starting sqream sql, after entering your password, you are presented with the SQL shell.

To exit the shell, type `\q` or `Ctrl-d`.

```
$ sqream sql --port=5000 --username=jdoe --databasename=master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> _
```

The database name shown means you are now ready to run statements and queries.

Statements and queries are standard SQL, followed by a semicolon (`;`). Statement results are usually formatted as a valid CSV, followed by the number of rows and the elapsed time for that statement.

```
master=> SELECT TOP 5 * FROM nba;
Avery Bradley           ,Boston Celtics        ,0,PG,25,6-2 ,180,Texas              ␣
↪,7730337
Jae Crowder             ,Boston Celtics        ,99,SF,25,6-6 ,235,Marquette         ␣
↪ ,6796117
John Holland            ,Boston Celtics        ,30,SG,27,6-5 ,205,Boston University  ␣
↪ ,\N
R.J. Hunter             ,Boston Celtics        ,28,SG,22,6-5 ,185,Georgia State      ␣
↪ ,1148640
Jonas Jerebko           ,Boston Celtics        ,8,PF,29,6-10,231,\N,5000000
5 rows
time: 0.001185s
```

Note: Null values are represented as \N.

When writing long statements and queries, it may be beneficial to use line-breaks. The prompt for a multi-line statement will change from => to `.`, to alert users to the change. The statement will not execute until a semicolon is used.

```
$ sqream sql --port=5000 --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> SELECT "Age",
. AVG("Salary")
. FROM NBA
. GROUP BY 1
. ORDER BY 2 ASC
. LIMIT 5
. ;
38,1840041
19,1930440
23,2034746
21,2067379
36,2238119
5 rows
time: 0.009320s
```

12.3.6.2.2 Executing Batch Scripts (-f)

To run an SQL script, use the `-f <filename>` argument.

For example,

```
$ sqream sql --port=5000 --username=jdoe -d master -f sql_script.sql --results-only
```

Tip: Output can be saved to a file by using redirection (>).

12.3.6.2.3 Executing Commands Immediately (-c)

To run a statement from the console, use the `-c <statement>` argument.

For example,

```
$ sqream sql --port=5000 --username=jdoe -d nba -c "SELECT TOP 5 * FROM nba"
Avery Bradley           ,Boston Celtics        ,0,PG,25,6-2 ,180,Texas              ␣
↪,7730337
Jae Crowder             ,Boston Celtics        ,99,SF,25,6-6 ,235,Marquette         ␣
↪ ,6796117
John Holland            ,Boston Celtics        ,30,SG,27,6-5 ,205,Boston University ␣
↪ ,\N
R.J. Hunter             ,Boston Celtics        ,28,SG,22,6-5 ,185,Georgia State      ␣
↪ ,1148640
Jonas Jerebko           ,Boston Celtics        ,8,PF,29,6-10,231,\N,5000000
5 rows
time: 0.202618s
```

Tip: Remove the timing and row count by passing the `--results-only` parameter

12.3.6.3 Examples

12.3.6.3.1 Starting a Regular Interactive Shell

Connect to local server 127.0.0.1 on port 5000, to the default built-in database, *master*:

```
$ sqream sql --port=5000 --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

Connect to local server 127.0.0.1 via the built-in load balancer on port 3108, to the default built-in database, *master*:

```
$ sqream sql --port=3105 --clustered --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

12.3.6.3.2 Executing Statements in an Interactive Shell

Note that all SQL commands end with a semicolon.

Creating a new database and switching over to it without reconnecting:

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> create database farm;
executed
time: 0.003811s
master=> \c farm
farm=>
```

```
farm=> create table animals(id int not null, name text(30) not null, is_angry bool␣
↪not null);
executed
time: 0.011940s

farm=> insert into animals values(1,'goat',false);
executed
time: 0.000405s

farm=> insert into animals values(4,'bull',true) ;
executed
time: 0.049338s
```

(continues on next page)

(continued from previous page)

```
farm=> select * from animals;
1,goat                          ,0
4,bull                          ,1
2 rows
time: 0.029299s
```

12.3.6.3.3 Executing SQL Statements from the Command Line

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals WHERE is_angry = true"
4,bull                          ,1
1 row
time: 0.095941s
```

12.3.6.3.4 Controlling the Client Output

Two parameters control the dispay of results from the client:

- `--results-only` - removes row counts and timing information
- `--delimiter` - changes the record delimiter

12.3.6.3.4.1 Exporting SQL Query Results to CSV

Using the `--results-only` flag removes the row counts and timing.

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals" --results-only > file.csv
$ cat file.csv
1,goat                          ,0
2,sow                           ,0
3,chicken                       ,0
4,bull                          ,1
```

12.3.6.3.4.2 Changing a CSV to a TSV

The `--delimiter` parameter accepts any printable character.

Tip: To insert a tab, use `Ctrl-V` followed by `Tab` ⇆ in Bash.

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals" --delimiter '	' > file.tsv
$ cat file.tsv
1	goat                            0
2	sow                             0
3	chicken                         0
4	bull                            1
```

12.3.6.3.5 Executing a Series of Statements From a File

Assuming a file containing SQL statements (separated by semicolons):

```
$ cat some_queries.sql
   CREATE TABLE calm_farm_animals
  ( id INT IDENTITY(0, 1), name TEXT(30)
  );

INSERT INTO calm_farm_animals (name)
  SELECT name FROM   animals WHERE  is_angry = false;
```

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -f some_queries.sql
executed
time: 0.018289s
executed
time: 0.090697s
```

12.3.6.3.6 Connecting Using Environment Variables

You can save connection parameters as environment variables:

```
$ export SQREAM_USER=sqream;
$ export SQREAM_DATABASE=farm;
$ sqream sql --port=3105 --clustered --username=$SQREAM_USER -d $SQREAM_DATABASE
```

12.3.6.3.7 Connecting to a Specific Queue

When using the *dynamic workload manager* - connect to `etl` queue instead of using the default `sqream` queue.

```
$ sqream sql --port=3105 --clustered --username=mjordan -d master --service=etl
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

12.3.6.4 Operations and Flag References

12.3.6.4.1 Command Line Arguments

Sqream SQL supports the following command line arguments:

Argument	Default	Description
`-c` or `--command`	None	Changes the mode of operation to single-command, non-interactive. Use this argument to run a statement and immediately exit.
`-f` or `--file`	None	Changes the mode of operation to multi-command, non-interactive. Use this argument to run a sequence of statements from an external file and immediately exit.
`--host`	`127.0.0.1`	Address of the SQream DB worker.
`--port`	`5000`	Sets the connection port.
`--databasename` or `-d`	None	Specifies the database name for queries and statements in this session.
`--username`	None	Username to connect to the specified database.
`--password`	None	Specify the password using the command line argument. If not specified, the client will prompt the user for the password.
`--cluster`	False	When used, the client connects to the load balancer, usually on port `3108`. If not set, the client assumes the connection is to a standalone SQream DB worker.
`--service`	`sqream`	*Service name (queue)* that statements will file into.
`--results-only`	False	Outputs results only, without timing information and row counts
`--no-history`	False	When set, prevents command history from being saved in `~/.sqream/clientcmdhist`
`--delimiter`	`,`	Specifies the field separator. By default, `sqream sql` outputs valid CSVs. Change the delimiter to modify the output to another delimited format (e.g. TSV, PSV). See the section *supported record delimiters* below for more information.

Tip: Run `$ sqream sql --help` to see a full list of arguments

12.3.6.4.1.1 Supported Record Delimiters

The supported record delimiters are printable ASCII values (32-126).

- Recommended delimiters for use are: `,`, `|`, tab character.
- The following characters are **not supported**: `\`, `N`, `-`, `:`, `"`, `\n`, `\r`, `.`, lower-case latin letters, digits (0-9)

12.3.6.4.2 Meta-Commands

- Meta-commands in Sqream SQL start with a backslash (`\`)

Note: Meta commands do not end with a semicolon

Command	Example	Description
`\q` or `\quit`	`master=> \q`	Quit the client. (Same as `Ctrl-d`)
`\c <database>` or `\connect <database>`	`master=> \c fox` `fox=>`	Changes the current connection to an alternate database

12.3.6.4.3 Basic Commands

12.3.6.4.4 Moving Around the Command Line

Command	Description
`Ctrl-a`	Goes to the beginning of the command line.
`Ctrl-e`	Goes to the end of the command line.
`Ctrl-u`	Deletes from cursor to the beginning of the command line.
`Ctrl-k`	Deletes from the cursor to the end of the command line.
`Ctrl-w`	Delete from cursor to beginning of a word.
`Ctrl-y`	Pastes a word or text that was cut using one of the deletion shortcuts (such as the one above) after the cursor.
`Alt-b`	Moves back one word (or goes to the beginning of the word where the cursor is).
`Alt-f`	Moves forward one word (or goes to the end of word the cursor is).
`Alt-d`	Deletes to the end of a word starting at the cursor. Deletes the whole word if the cursor is at the beginning of that word.
`Alt-c`	Capitalizes letters in a word starting at the cursor. Capitalizes the whole word if the cursor is at the beginning of that word.
`Alt-u`	Capitalizes from the cursor to the end of the word.
`Alt-l`	Makes lowercase from the cursor to the end of the word.
`Ctrl-f`	Moves forward one character.
`Ctrl-b`	Moves backward one character.
`Ctrl-h`	Deletes characters located before the cursor.
`Ctrl-t`	Swaps a character at the cursor with the previous character.

12.3.6.4.5 Searching

Command	Description
`Ctrl-r`	Searches the history backward.
`Ctrl-g`	Escapes from history-searching mode.
`Ctrl-p`	Searches the previous command in history.
`Ctrl-n`	Searches the next command in history.

12.3.7 upgrade_storage

`upgrade_storage` is used to upgrade metadata schemas, when upgrading between major versions.

This page serves as a reference for the options and parameters.

12.3.7.1 Running upgrade_storage

`upgrade_storage` can be found in the `bin` directory of your SQream DB installation.

12.3.7.1.1 Command line arguments and options

Parameter	Parameter Type	Description
`storage_path`	Argument	Full path to a valid storage cluster.
`--storage_version`	Option	Displays your current storage version.
`--check_predicates=0`	Option	Allows the upgrade process to proceed even if there are predicates marked for deletion.

12.3.7.1.2 Syntax

```
$ upgrade_storage <storage path> [--check_predicates=0]
```

```
$ upgrade_storage <storage path> [--storage_version]
```

12.3.7.2 Results and error codes

Result	Message	Description
Success	`storage has been upgraded successfully to version 26`	Storage has been successfully upgraded
Success	`no need to upgrade`	Storage doesn't need an upgrade
Failure: can't read storage	`RocksDB is in use by another application`	Check permissions, and ensure no SQream DB workers or *metadata_server* are running when performing this operation.

12.3.7.3 Examples

12.3.7.3.1 Upgrade SQream DB's storage cluster

```
$ ./upgrade_storage /home/rhendricks/raviga_database
get_rocksdb_version path{/home/rhendricks/raviga_database}
current storage version 23
upgrade_v24
upgrade_storage to 24
upgrade_storage to 24 - Done
upgrade_v25
```

(continues on next page)

(continued from previous page)

```
upgrade_storage to 25
upgrade_storage to 25 - Done
upgrade_v26
upgrade_storage to 26
upgrade_storage to 26 - Done
validate_rocksdb
storage has been upgraded successfully to version 26
```

This message confirms that the cluster has already been upgraded correctly.

12.4 SQL Feature Checklist

To understand which ANSI SQL and other SQL features SQreamDB supports, use the tables below.

In this topic:

- *Data Types and Values*
- *Constraints*
- *Transactions*
- *Indexes*
- *Schema Changes*
- *Statements*
- *Clauses*
- *Table Expressions*
- *Scalar Expressions*
- *Permissions*
- *Extra Functionality*

12.4.1 Data Types and Values

Table 7: Data Types and Values

Item	Supported	Further information
`BOOL`	Yes	Boolean values
`TINTINT`	Yes	Unsigned 1 byte integer (0 - 255)
`SMALLINT`	Yes	2 byte integer (-32,768 - 32,767)
`INT`	Yes	4 byte integer (-2,147,483,648 - 2,147,483,647)
`BIGINT`	Yes	8 byte integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807)
`REAL`	Yes	4 byte floating point
`DOUBLE`, `FLOAT`	Yes	8 byte floating point
`DECIMAL`, `NUMERIC`	Yes	Fixed-point numbers.
`TEXT`	Yes	Variable length string - UTF-8 encoded
`DATE`	Yes	Date
`DATETIME`, `TIMESTAMP`	Yes	Date and time
`NULL`	Yes	`NULL` values
`TIME`	No	Can be stored as a text string or as part of a `DATETIME`

12.4.2 Constraints

Table 8: Constraints

Item	Supported	Further information
`Not null`	Yes	`NOT NULL`
`Default values`	Yes	`DEFAULT`
`AUTO INCREMENT`	Yes (different name)	`IDENTITY`

12.4.3 Transactions

SQreamDB treats each statement as an auto-commit transaction. Each transaction is isolated from other transactions with serializable isolation.

If a statement fails, the entire transaction is canceled and rolled back. The database is unchanged.

12.4.4 Indexes

SQreamDB has a range-index collected on all columns as part of the metadata collection process.

SQreamDB does not support explicit indexing, but does support clustering keys.

Read more about *clustering keys*.

12.4.5 Schema Changes

Table 9: Schema Changes

Item	Supported	Further information
`ALTER TABLE`	Yes	alter_table - Add column, alter column, drop column, rename column, rename table, modify clustering keys
Rename database	No	
Rename table	Yes	rename_table
Rename column	Yes	rename_column
Add column	Yes	add_column
Remove column	Yes	drop_column
Alter column data type	No	
Add / modify clustering keys	Yes	cluster_by
Drop clustering keys	Yes	drop_clustering_key
Add / Remove constraints	No	
Rename schema	Yes	rename_schema
Drop schema	Yes	drop_schema
Alter default schema per user	Yes	alter_default_schema

12.4.6 Statements

Table 10: Statements

Item	Supported	Further information
SELECT	Yes	select
CREATE TABLE	Yes	create_table
CREATE FOREIGN / EXTERNAL TABLE	Yes	create_foreign_table
DELETE	Yes	*Deleting Data*
INSERT	Yes	insert, copy_from
TRUNCATE	Yes	truncate
UPDATE	Yes	
VALUES	Yes	values

12.4.7 Clauses

Table 11: Clauses

Item	Supported	Further information
`LIMIT` / `TOP`	Yes	
`LIMIT` with `OFFSET`	No	
`WHERE`	Yes	
`HAVING`	Yes	
`OVER`	Yes	

12.4.8 Table Expressions

Table 12: Table Expressions

Item	Supported	Further information
Tables, Views	Yes	
Aliases, `AS`	Yes	
`JOIN` - `INNER`, `LEFT [ OUTER ]`, `RIGHT [ OUTER ]`, `CROSS`	Yes	
Table expression subqueries	Yes	
Scalar subqueries	No	

12.4.9 Scalar Expressions

Read more about scalar_expressions.

Table 13: Scalar Expressions

Item	Supported	Further information
Common functions	Yes	`CURRENT_TIMESTAMP`, `SUBSTRING`, `TRIM`, `EXTRACT`, etc.
Comparison operators	Yes	`<`, `<=`, `>`, `>=`, `=`, `<>`, `!=`, `IS`, `IS NOT`
Boolean operators	Yes	`AND`, `NOT`, `OR`
Conditional expressions	Yes	`CASE .. WHEN`
Conditional functions	Yes	`COALESCE`
Pattern matching	Yes	`LIKE`, `RLIKE`, `ISPREFIXOF`, `CHARINDEX`, `PATINDEX`
REGEX POSIX pattern matching	Yes	`RLIKE`, `REGEXP_COUNT`, `REGEXP_INSTR`, `REGEXP_SUBSTR`,
`EXISTS`	No	
`IN`, `NOT IN`	Partial	Literal values only
Bitwise arithmetic	Yes	`&`, `|`, `XOR`, `~`, `>>`, `<<`

12.4.10 Permissions

Read more about *Access Control* in SQreamDB.

Table 14: Permissions

Item	Supported	Further information
Roles as users and groups	Yes	
Object default permissions	Yes	
Column / Row based permissions	No	
Object ownership	No	

12.4.11 Extra Functionality

Table 15: Extra Functionality

Item	Supported	Further information
Information schema	Yes	*Catalog Reference*
Views	Yes	create_view
Window functions	Yes	window_functions
CTEs	Yes	common_table_expressions
Saved queries, Saved queries with parameters	Yes	*Saved Queries*
Sequences	Yes	identity

CHAPTER

THIRTEEN

DATA TYPES

This section describes the following:

13.1 Supported Data Types

Data types define the type of data that a column can hold in a table. They ensure that data is handled accurately and efficiently. Common data types include integers, decimals, characters, and dates. For example, an `INT` data type is used for whole numbers, `TEXT` for variable-length character strings, and `DATE` for date values.

13.1.1 Primitive Data Types

SQreamDB compresses all columns and types. The data size noted is the maximum data size allocation for uncompressed data.

Name	Description	Data Size (Not Null, Uncompressed)	Example	Alias
BOOL	Boolean values (true, false)	1 byte	true	BIT
TINYINT	Unsigned integer (0 - 255)	1 byte	5	NA
SMALLINT	Integer (-32,768 - 32,767)	2 bytes	-155	NA
INT	Integer (-2,147,483,648 - 2,147,483,647)	4 bytes	1648813	INTEGER
BIGINT	Integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807)	8 bytes	36124441255243	NUMBER
REAL	Floating point (inexact)	4 bytes	3.141	NA
DOUBLE	Floating point (inexact)	8 bytes	0.000003	FLOAT/DOUBLE PRECISION
TEXT (n)	Variable length string - UTF-8 unicode	Up to 4 bytes	'Kiwis have tiny wings, but cannot fly.'	CHAR VARYING, CHAR, CHARACTER VARYING, CHARACTER, NATIONAL CHARACTER VARYING, NATIONAL CHARACTER, NCHAR VARYING, NCHAR, NATIONAL CHAR, NATIONAL CHAR VARYING
NUMERIC	38 digits	16 bytes	0.12324567890123456789012345678901234567	DECIMAL
DATE	Date	4 bytes	'1955-11-05'	NA
DATETIME	Date and time pairing in UTC	8 bytes	'1955-11-05 01:24:00.000'	TIMESTAMP, DATETIME2

13.1.2 Array

The ARRAY data type offers a convenient way to store ordered collections of elements in a single column. It provides storage efficiency by allowing multiple values of the same data type to be compactly stored, optimizing space utilization and enhancing database performance. Working with ARRAY simplifies queries as operations and manipulations can be performed on the entire ARRAY, resulting in more concise and readable code.

An ARRAY represents a sequence of zero or more elements of the same data type. Arrays in the same column can contain varying numbers of elements across different rows. Arrays can include null values, eliminating the need for separate SQL declarations.

Each data type has its companion ARRAY type, such as INT[] for integers and TEXT[] for text values.

You may use the ARRAY data type with all *SQreamDB connectors*, except for ODBC since the ODBC protocol does not support ARRAY.

The maximum size of an ARRAY, indicating the number of elements it can hold, is 65535. You have the option to specify the size of an ARRAY, providing a maximum allowed size, while each row can have a different number of elements up to the specified maximum. If the ARRAY size is not specified, the maximum size is assumed.

See also:

A full list of *data types* supported by SQreamDB.

13.1.2.1 Limitation

Please note that SQLAlchemy does not support the ARRAY data type.

13.1.2.1.1 Syntax

Defining an ARRAY is done by appending the [] notation to a supported data type, for example, INT[] for an array of integers.

```
CREATE TABLE <table_name> (<column1> TEXT[], <column2> INT[])

INSERT INTO TABLE <table_name> VALUES (ARRAY['a','b','c'], ARRAY[1,2,NULL])
```

13.1.2.1.2 Supported Operators

Operator	Description	Example
Literals `ARRAY []`	Literals are created using the `ARRAY` operator	`ARRAY[1,2,3]`
Mapping	Parquet, ORC, JSON, and AVRO `ARRAY` types may be mapped into SQreamDB `ARRAY`	See extended section under **Examples**
Indexing	Access to specific elements within the array by using a **zero-based index**	`SELECT (<column_name>[2]) FROM <table_name>` returns the third element of the specified column
`UNNEST`	Converts the arrayed elements within a single row into a set of rows	`SELECT UNNEST(<column_name>) FROM <table_name>`
Concatenate `||`	Converts arrayed elements into one string	• `SELECT (<column_name>) || (<column2_name>) FROM <table_name>` • `SELECT (<column_name>) || ARRAY[1,2,3] FROM <table_name>`
`array_length`	Returns the number of arrayed elements within the specified column	`SELECT array_length(<column_name>) FROM <table_name>`
`array_position`	Locates the position of the specified value within the specified array. Returns `NULL` if the value is not found	`SELECT array_position(<column_name>, <value>) FROM <table_name>;`
`array_remove`	Returns the specified `ARRAY` column with the specified value deducted	`SELECT array_remove(<column_name>, <value>) FROM <table_name>;`
`array_replace`	Enables replacing values within an `ARRAY` column	`SELECT array_replace(<column_name>, <value_to_replace>, <replacing_value>) FROM <table_name>;`
Limiting number of arrayed elements	You may limit the number of arrayed elements within an `ARRAY`	Limiting the number of arrayed elements to 4: `CREATE TABLE <table_name> (<column1> TEXT[4]);`
Compression	You may follow SQreamDB *compression guide* for compression types and methods	`CREATE TABLE t (comp_dict INT[] CHECK('CS "dict"');`
Aggregation	The `array_agg()` function arrays groups created using the `GROUP BY` clause	`CREATE TABLE t2 (x INT, y INT);` `SELECT x, array_agg(y) FROM t2 GROUP BY x;`
Sorting	`TEXT[]` elements are considered together as a single text, and comparisons are made based on their lexicographic order. In contrast, for arrays of non-TEXT data types, com-	`CREATE TABLE t (x TEXT[]);` `INSERT INTO t VALUES (ARRAY['1']),`

13.1.2.1.3 Examples

13.1.2.2 ARRAY Statements

Creating a table with arrayed columns:

```
CREATE TABLE my_array (clmn1 TEXT[], clmn2 TEXT[], clmn3 INT[], clmn4 NUMERIC(38,
↪20)[]);
```

Inserting arrayed values into a table:

```
INSERT INTO my_array VALUES (ARRAY['1','2','3'], ARRAY['4','5','6'], ARRAY[7,8,9,10]);
```

Converting arrayed elements into a set of rows:

```
SELECT UNNEST(ARRAY['1','2','3'], ARRAY['4','5','6']);
```

```
 clmn1     | clmn2
----------+----------
 "1"       | "4"
 "2"       | "5"
 "3"       | "6"
```

Updating table values:

```
UPDATE my_array SET clmn1[0] = 'A';

SELECT * FROM my_array;
```

```
clmn1                 | clmn2             | clmn3
----------------------+-------------------+-----------
["A","1","2","3"]     | ["4","5","6"]     | [7,8,9,10]
```

13.1.2.3 Ingesting Arrayed Data from External Files

Consider the following JSON file named `t`, located under `/tmp/`:

```
[{
                "name": "Avery Bradley",
                "age": 25,
                "position": "PG",
                "years_in_nba": [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
↪ 2019, 2020, 2021]
        },
        {
                "name": "Jae Crowder",
                "age": 25,
                "position": "PG",
                "years_in_nba": [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020,
↪ 2021]
        },
        {
                "name": "John Holland",
                "age": 27,
                "position": "SG",
```

(continues on next page)

(continued from previous page)

```
                "years_in_nba": [2017, 2018]
        }
]
```

Execute the following statement:

```
CREATE FOREIGN TABLE nba
           (
           name text,
           age int,
           position text,
           years_in_nba int[]
           )

  WRAPPER json_fdw
  OPTIONS
  (
        location ='/tmp/t.json'
  )
;

SELECT * FROM nba;
```

Result:

```
name            | age     | position     | years_in_nba
----------------+---------+--------------+---------------------------------------------
↪--------------------------
Avery Bradley   | 25      | PG           | [2010, 2011, 2012, 2013, 2014, 2015, 2016,␣
↪2017, 2018, 2019, 2020, 2021]
Jae Crowder     | 25      | PG           | [2012, 2013, 2014, 2015, 2016, 2017, 2018,␣
↪2019, 2020, 2021]
John Holland    | 27      | SG           | [2017, 2018]
```

13.1.2.3.1 Limitations

13.1.2.4 Connectors

13.1.2.4.1 Limitations

Please note that the SQreamDB ODBC and .NET connectors do not support the use of ARRAY data types. If your database schema includes ARRAY columns, you may encounter compatibility issues when using these connectors.

13.1.2.5 Casting

Numeric data types smaller than `INT`, such as `TINYINT`, `SMALLINT`, and `BOOL`, must explicitly be cast.

```
CREATE OR REPLACE TABLE my_array (clmn1 tinyint []);
SELECT array_replace(clmn1 , 4::tinyint, 5::tinyint) FROM my_array;

CREATE OR REPLACE TABLE my_array (clmn1 bool []);
SELECT array_replace(clmn1 , 0::bool, 1::bool) FROM my_array;
```

Casting `TEXT` to non-`TEXT` and non-`TEXT` to `TEXT` data types is not supported.

```
CREATE TABLE t_text (xtext TEXT[]);
CREATE TABLE t_int (xint INT[]);
INSERT INTO t_int VALUES (array[1,2,3]);
INSERT INTO t_text SELECT xint::TEXT[] FROM t_int;
```

13.1.2.6 Function

Using the `||` (Concatenate) function with two different data types requires explicit casting.

```
SELECT (clmn1, 4::tinyint) || (clmn2, 5::tinyint) FROM my_array;
```

It is possible to use the `UNNEST` operator within a statement only once.

Window functions are not supported.

13.2 Casts and Conversions

SQream supports explicit and implicit casting and type conversion. The system may automatically add implicit casts when combining different data types in the same expression. In many cases, while the details related to this are not important, they can affect the results of a query. When necessary, an explicit cast can be used to override the automatic cast added by SQream DB.

For example, the ANSI standard defines a `SUM()` aggregation over an `INT` column as an `INT`. However, when dealing with large amounts of data this could cause an overflow.

You can rectify this by casting the value to a larger data type, as shown below:

```
SUM(some_int_column :: BIGINT)
```

SQream supports the following three data conversion types:

- `CAST(<value> AS <data type>)`, to convert a value from one type to another. For example, `CAST('1997-01-01' AS DATE)`,`CAST(3.45 AS SMALLINT)`,`CAST(some_column AS TEXT)`.

- `<value> :: <data type>`, a shorthand for the `CAST` syntax. For example, `'1997-01-01' :: DATE`, `3.45 :: SMALLINT`, `(3+5) :: BIGINT`.

- See the *SQL functions reference* for additional functions that convert from a specific value which is not an SQL type, such as from_unixts, etc.

13.2.1 Supported Casts

	BOOL	TINYINT/SMALLINT/INT/BIGINT	REAL/FLOAT	NUMERIC	DATE/DATETIME	VARCHAR/TEXT
BOOL	N/A	✓	✗	✗	✗	✓
TINYINT/SMALLINT/INT/BIGINT	✓	N/A	✓	✓	✗	✓
REAL/FLOAT	✗	✓	N/A	✓	✗	✓
NUMERIC	✗	✓	✓	✓	✗	✓
DATE/DATETIME	✗	✗	✗	✗	N/A	✓
VARCHAR/TEXT	✓	✓	✓	✓	✓	N/A

13.3 Supported Casts

The **Supported Casts** section describes supported casts for the following types:

13.3.1 Numeric

The **Numeric** data type (also known as **Decimal**) is recommended for values that tend to occur as exact decimals, such as in Finance. While Numeric has a fixed precision of `38`, higher than `REAL(9)` or `DOUBLE(17)`, it runs calculations more slowly. For operations that require faster performance, using Floating Point is recommended.

The correct syntax for Numeric is `numeric(p, s)`, where `p` is the total number of digits (`38` maximum), and `s` is the total number of decimal digits. If no parameters are specified, Numeric defaults to `numeric(38, 0)`.

13.3.1.1 Numeric Examples

The following is an example of the Numeric syntax:

```
CREATE OR REPLACE table t(x numeric(20, 10), y numeric(38, 38));
INSERT INTO t VALUES(1234567890.1234567890, 0.12324567890123456789012345678901234567);
SELECT x + y FROM t;
```

The following table shows information relevant to the Numeric data type:

Description	Data Size (Not Null, Uncompressed)	Example
38 digits	16 bytes	`0.12324567890123456789012345678901234`

Numeric supports the following operations:

- All join types.
- All aggregation types (not including Window functions).
- Scalar functions (not including some trigonometric and logarithmic functions).

13.3.2 Boolean

The following table describes the Boolean data type.

Values	Syntax	Data Size (Not Null, Uncompressed)
`true`, `false` (case sensitive)	When loading from CSV, `BOOL` columns can accept `0` as `false` and `1` as `true`.	1 byte, but resulting average data sizes may be lower after compression.

13.3.2.1 Boolean Examples

The following is an example of the Boolean syntax:

```
CREATE TABLE animals (name TEXT, is_angry BOOL);

INSERT INTO animals VALUES ('fox',true), ('cat',true), ('kiwi',false);

SELECT name, CASE WHEN is_angry THEN 'Is really angry!' else 'Is not angry' END FROM␣
↪animals;
```

The following is an example of the correct output:

```
"fox","Is really angry!"
"cat","Is really angry!"
"kiwi","Is not angry"
```

13.3.2.2 Boolean Casts and Conversions

The following table shows the possible Boolean value conversions:

Type	Details
`TINYINT`, `SMALLINT`, `INT`, `BIGINT`	`true` → `1`, `false` → `0`
`REAL`, `DOUBLE`	`true` → `1.0`, `false` → `0.0`

13.3.3 Integer

Integer data types are designed to store whole numbers.

For more information about identity sequences (sometimes called auto-increment or auto-numbers), see identity.

13.3.3.1 Integer Types

The following table describes the Integer types.

Name	Details	Data Size (Not Null, Uncompressed)	Example
`TINYINT`	Unsigned integer (0 - 255)	1 byte	`5`
`SMALLINT`	Integer (-32,768 - 32,767)	2 bytes	`-155`
`INT`	Integer (-2,147,483,648 - 2,147,483,647)	4 bytes	`1648813`
`BIGINT`	Integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807)	8 bytes	`36124441255243`

The following table describes the Integer data type.

Syntax	Data Size (Not Null, Uncompressed)
An integer can be entered as a regular literal, such as `12`, `-365`.	Integer types range between 1, 2, 4, and 8 bytes - but resulting average data sizes could be lower after compression.

13.3.3.2 Integer Examples

The following is an example of the Integer syntax:

```
CREATE TABLE cool_numbers (a INT NOT NULL, b TINYINT, c SMALLINT, d BIGINT);

INSERT INTO cool_numbers VALUES (1,2,3,4), (-5, 127, 32000, 45000000000);

SELECT * FROM cool_numbers;
```

The following is an example of the correct output:

```
1,2,3,4
-5,127,32000,45000000000
```

13.3.3.3 Integer Casts and Conversions

The following table shows the possible Integer value conversions:

Type	Details
`REAL`, `DOUBLE`	`1` → `1.0`, `-32` → `-32.0`
`TEXT` (All numberic values must fit in the string length)	`1` → `'1'`, `2451` → `'2451'`

13.3.4 Floating Point

The **Floating Point** data types (`REAL` and `DOUBLE`) store extremely close value approximations, and are therefore recommended for values that tend to be inexact, such as Scientific Notation. While Floating Point generally runs faster than Numeric, it has a lower precision of `9` (`REAL`) or `17` (`DOUBLE`) compared to Numeric's `38`. For operations that require a higher level of precision, using Numeric is recommended.

The floating point representation is based on IEEE 754.

13.3.4.1 Floating Point Types

The following table describes the Floating Point data types.

Name	Details	Data Size (Not Null, Uncompressed)	Example
`REAL`	Single precision floating point (inexact)	4 bytes	`3.141`
`DOUBLE`	Double precision floating point (inexact)	8 bytes	`0.000003`

The following table shows information relevant to the Floating Point data types.

Aliases	Syntax	Data Size (Not Null, Uncompressed)
`DOUBLE` is also known as `FLOAT`.	A double precision floating point can be entered as a regular literal, such as `3.14`, `2.718`, `.34`, or `2.71e-45`. To enter a `REAL` floating point number, cast the value. For example, `(3.14 :: REAL)`.	Floating point types are either 4 or 8 bytes, but size could be lower after compression.

13.3.4.2 Floating Point Examples

The following are examples of the Floating Point syntax:

```
CREATE TABLE cool_numbers (a REAL NOT NULL, b DOUBLE);

INSERT INTO cool_numbers VALUES (1,2), (3.14159265358979, 2.718281828459);

SELECT * FROM cool_numbers;
```

```
1.0,2.0
3.1415927,2.718281828459
```

Note: Most SQL clients control display precision of floating point numbers, and values may appear differently in some clients.

13.3.4.3 Floating Point Casts and Conversions

The following table shows the possible Floating Point value conversions:

Type	Details
`BOOL`	`1.0` → `true`, `0.0` → `false`
`TINYINT`, `SMALLINT`, `INT`, `BIGINT`	`2.0` → `2`, `3.14159265358979` → `3`, `2.718281828459` → `2`, `0.5` → `0`, `1.5` → `1`

Note: As shown in the above examples, casting `real` to `int` rounds down.

13.3.5 String

`TEXT` is designed for storing text or strings of characters. SQreamDB blocks non-UTF8 string inputs.

13.3.5.1 Length

When using `TEXT`, specifying a size is optional. If not specified, the text field carries no constraints. To limit the size of the input, use `TEXT(n)`, where `n` is the permitted number of characters.

The following apply to setting the String type length:

- If the data exceeds the column length limit on `INSERT` or `COPY` operations, SQreamDB will return an error.
- When casting or converting, the string has to fit in the target. For example, `'Kiwis are weird birds' :: TEXT(5)` will return an error. Use `SUBSTRING` to truncate the length of the string.

13.3.5.2 Syntax

String types can be written with standard SQL string literals, which are enclosed with single quotes, such as `'Kiwi bird'`. To include a single quote in the string, use double quotations, such as `'Kiwi bird''s wings are tiny'`. String literals can also be dollar-quoted with the dollar sign `$`, such as `$$Kiwi bird's wings are tiny$$` is the same as `'Kiwi bird''s wings are tiny'`.

13.3.5.3 Size

`TEXT(n)` can occupy up to *4*n* bytes. However, the size of strings is variable and is compressed by SQreamDB.

13.3.5.4 String Examples

The following is an example of the String syntax:

```
CREATE TABLE cool_strings (a TEXT NOT NULL, b TEXT);

INSERT INTO cool_strings VALUES ('hello world', 'Hello to kiwi birds specifically');

INSERT INTO cool_strings VALUES ('This is ASCII only', 'But this column can contain␣
↪⍰⍰⍰⍰');
```

(continues on next page)

(continued from previous page)

```
SELECT * FROM cool_strings;
```

The following is an example of the correct output:

```
hello world  ,Hello to kiwi birds specifically
This is ASCII only,But this column can contain ????
```

Note: Most clients control the display precision of floating point numbers, and values may appear differently in some clients.

13.3.5.5 String Casts and Conversions

The following table shows the possible String value conversions:

Type	Details
BOOL	'true' → true, 'false' → false
TINYINT, SMALLINT, INT, BIGINT	'2' → 2, '-128' → -128
REAL, DOUBLE	'2.0' → 2.0, '3.141592' → 3.141592
DATE, DATETIME	Requires a supported format, such as '1955-11-05 → date '1955-11-05', '1955-11-05 01:24:00.000' → '1955-11-05 01:24:00.000'

13.3.6 Date

DATE is a type designed for storing year, month, and day. DATETIME is a type designed for storing year, month, day, hour, minute, seconds, and milliseconds in UTC with 1 millisecond precision.

13.3.6.1 Date Types

The following table describes the Date types:

Table 1: Date Types

Name	Details	Data Size (Not Null, Uncompressed)	Example
DATE	Date	4 bytes	'1955-11-05'
DATETIME	Date and time pairing in UTC	8 bytes	'1955-11-05 01:24:00.000'

13.3.6.2 Aliases

`DATETIME` is also known as `TIMESTAMP` or `DATETIME2`.

13.3.6.3 Syntax

`DATE` values are formatted as string literals.

The following is an example of the DATETIME syntax:

```
'1955-11-05'
```

```
date '1955-11-05'
```

`DATETIME` values are formatted as string literals conforming to ISO 8601.

The following is an example of the DATETIME syntax:

```
'1955-11-05 01:26:00'
```

SQream attempts to guess if the string literal is a date or datetime based on context, for example when used in date-specific functions.

13.3.6.4 Size

A `DATE` column is 4 bytes in length, while a `DATETIME` column is 8 bytes in length.

However, the size of these values is compressed by SQream DB.

13.3.6.5 Date Examples

The following is an example of the Date syntax:

```
CREATE TABLE important_dates (a DATE, b DATETIME);

INSERT INTO important_dates VALUES ('1997-01-01', '1955-11-05 01:24');

SELECT * FROM important_dates;
```

The following is an example of the correct output:

```
1997-01-01,1955-11-05 01:24:00.0
```

The following is an example of the Datetime syntax:

```
SELECT a :: DATETIME, b :: DATE FROM important_dates;
```

The following is an example of the correct output:

```
1997-01-01 00:00:00.0,1955-11-05
```

13.3.6.6 Date Casts and Conversions

The following table shows the possible `DATE` and `DATETIME` value conversions:

Type	Details
`TEXT`	`'1997-01-01'` → `'1997-01-01'`, `'1955-11-05 01:24'` → `'1955-11-05 01:24:00.000'`

CHAPTER

FOURTEEN

RELEASE NOTES

Version 4.6 - February 19, 2024

- Execute a single SQL statement across your SQreamDB cluster using the new Cross-Database syntax.
- Safely cast data types with the new IsCastable function.

Version 4.5 - December 5, 2023

- Introducing a new Health-Check Monitor utility command empowers administrators to oversee the database's health. This command serves as a valuable tool for monitoring, enabling administrators to assess and ensure the optimal health and performance of the database
- A new Query Timeout session flag designed to identify queries that have exceeded a specified time limit. Once the flag value is reached, the query automatically stops

Version 4.4 - September 28, 2023

- Enhancing storage efficiency and performance with the newly supported ARRAY data type
- New integration with Denodo Platform

Version 4.3 - June 11, 2023

- Access Control Permission Expansion
- New AWS S3 Access Configurations

Version 4.2 - April 23, 2023

- New Apache Spark Connector
- Physical Deletion Performance Enhancement

Version 4.1 - March 01, 2032

- LDAP Management Enhancements
- New Trino Connector
- Brute-Force Attack Protection

Version 4.0 - January 25, 2023

- SQreamDB License Storage Capacity
- LDAP Authentication
- Physical Deletion Performance Enhancement

14.1 4.0 Release Notes

14.1.1 Release Notes 4.0

SQream is introducing a new version release system that follows the more commonly used Major.Minor versioning schema. The newly released **4.0 version** is a minor version upgrade and does not require considerable preparation.

The 4.0 release notes were released on 01/25/2023 and describe the following:

- *New Features*
- *Storage Version*
- *SQream Studio Updates and Improvements*
- *Known Issues*
- *Version 4.0 resolved Issues*
- *Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to version 4.0*

14.1.1.1 New Features

- Re-enabling an enhanced version of the License Storage Capacity feature
- *Lightweight Directory Access Protocol(LDAP)* may be used to authenticate SQream roles
- Physical deletion performance enhancement by supporting file systems with parallelism capabilities

14.1.1.2 Storage Version

The storage version presently in effect is version 45.

14.1.1.3 SQream Studio Updates and Improvements

- When creating a **New Role**, you may now create a group role by selecting **Set as a group role**.
- When editing an **Existing Role**, you are no longer obligated to update the role's password.

14.1.1.4 Known Issues

Percentile is not supported for Window functions.

14.1.1.5 Version 4.0 resolved Issues

SQ No.	Description
SQ-10544	SQream Studio dashboard periodic update enhancement
SQ-11296	Slow catalog queries
SQ-11772	Slow query performance when using `JOIN` clause
SQ-12318	JDBC `insertBuffer` parameter issue
SQ-12364	`GET DDL` foreign table output issue
SQ-12446	SQream Studio group role modification issue
SQ-12468	Internal compiler error
SQ-12580	Server Picker GPU dependency
SQ-12598	Executing `SELECT` on a foreign table with no valid path produces no error message
SQ-12652	SQream Studio result panel adjustment
SQ-13055	NULL issue when executing query with pysqream

14.1.1.6 Configuration Changes

No configuration changes were made.

14.1.1.7 Naming Changes

No relevant naming changes were made.

14.1.1.8 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective from Version 2022.1.3 (September 2022).

TEXT data type is replacing VARCHAR and NVARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

14.1.1.9 End of Support

No End of Support changes were made.

14.1.1.10 Upgrading to version 4.0

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

Tip: SQream recommends storing the generated back-up locally in case needed.

SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.1.2 Release Notes 4.1

SQream is introducing a new version release system that follows the more commonly used Major.Minor.Patch versioning schema. The newly released **4.0 version** is a minor version upgrade and does not require considerable preparation.

The 4.1 release notes were released on 03/01/2023 and describe the following:

- *New Features*
- *Newly Released Connector Drivers*
- *Storage Version*
- *SQream Studio Updates and Improvements*
- *Known Issues*
- *Version 4.1 resolved Issues*
- *Configuration Changes*
- *Naming Changes*
- *Deprecated Features*

- *End of Support*
- *Upgrading to v4.1*

14.1.2.1 New Features

- *Lightweight Directory Access Protocol (LDAP)* management enhancement

- A new brute-force attack protection mechanism locks out user accounts for 15 minutes following 5 consecutive failed login attempts

14.1.2.2 Newly Released Connector Drivers

JDBC 4.5.7 .jar file

14.1.2.3 Storage Version

The storage version presently in effect is version 45.

14.1.2.4 SQream Studio Updates and Improvements

SQream Studio v5.5.4 has been released.

14.1.2.5 Known Issues

Percentile is not supported for Window functions.

14.1.2.6 Version 4.1 resolved Issues

SQ No.	Description
SQ-11287	Function definition SQL UDF parenthesis issue
SQ-11296	Slow catalog queries
SQ-12255	Text column additional characters when using `COPY TO` statement
SQ-12510	Encryption memory issues
SQ-13219	JDBC `supportsSchemasInDataManipulation()` method issue

14.1.2.7 Configuration Changes

No configuration changes

14.1.2.8 Naming Changes

No naming changes

14.1.2.9 Deprecated Features

► Square Brackets `[]`

The `[]`, which are frequently used to delimit identifiers such as column names, table names, and other database objects, will soon be deprecated to facilitate the use of the `ARRAY` data type.

- Support in `[]` for delimiting database object identifiers ends on June 1st, 2023.
- To delimit database object identifiers, you will be able to use double quotes `""`.

► `VARCHAR`

The `VARCHAR` data type is deprecated to improve the core functionalities of the platform and to align with the constantly evolving ecosystem requirements.

- Support in the `VARCHAR` data type ends at September 30th, 2023.
- `VARCHAR` is no longer supported for new customers, effective from Version 2022.1.3.
- The `TEXT` data type is replacing the `VARCHAR` and `NVARCHAR` data types.

14.1.2.10 End of Support

No End of Support changes were made.

14.1.2.11 Upgrading to v4.1

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Copy the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.1.3 Release Notes 4.2

SQream is introducing a new version release system that follows the more commonly used Major.Minor.Patch versioning schema. The newly released **4.0 version** is a minor version upgrade and does not require considerable preparation.

The 4.2 release notes were released on 04/23/2023 and describe the following:

- *New Features*
- *Newly Released Connector Drivers*
- *Compatibility Matrix*
- *SQream Studio Updates and Improvements*
- *Known Issues*
- *Version 4.2 Resolved Issues*
- *Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v4.2*

14.1.3.1 New Features

Apache Spark may now be used for large-scale data processing.

Physical deletion performance enhancement by supporting file systems with parallelism capabilities

14.1.3.2 Newly Released Connector Drivers

Pysqream 3.2.5

- Supports Python version 3.9 and newer
- .tar file
- *Documentation*

ODBC 4.4.4

- *Getting the ODBC Driver*

JDBC 4.5.8

- .jar file
- *Documentation*

Spark 5.0.0

- .jar file
- Documentation

14.1.3.3 Compatibility Matrix

System Requirement	Details
Supported OS	• CentOS / REHL - 7.6 - 7.9 • IBM RedHat 7.6
supported Nvidia driver	CUDA version from 10.1 up to 11.4.3
Storage version	46
Driver compatibility	• JDBC 4.5.8 • ODBC 4.4.4 • NodeJS • .NET 3.0.2 • Pysqream 3.2.5 • Spark

14.1.3.4 SQream Studio Updates and Improvements

SQream Studio v5.5.4 has been released.

14.1.3.5 Known Issues

- Percentile is not supported for Window Functions.
- Performance degradation when using `VARCHAR` partition key in a Window Functions expression

14.1.3.6 Version 4.2 Resolved Issues

SQ No.	Description
SQ-12598	Foreign table `SELECT` statement issue
SQ-13018	*cleanup_extent* operation buffer issue
SQ-13055	Pysqream `NULL` value issue
SQ-13322	Clean up process is case sensitive
SQ-13450	Storage upgrade issue

14.1.3.7 Configuration Changes

No configuration changes

14.1.3.8 Naming Changes

No naming changes

14.1.3.9 Deprecated Features

► `INT96`

Due to Parquet's lack of support of the `INT96` data type, SQream has decided to deprecate this data type.

► Square Brackets `[]`

The `[]`, which are frequently used to delimit identifiers such as column names, table names, and other database objects, will soon be deprecated to facilitate the use of the `ARRAY` data type.

- Support in `[]` for delimiting database object identifiers ends on June 1st, 2023.
- To delimit database object identifiers, you will be able to use double quotes `""`.

► `VARCHAR`

The `VARCHAR` data type is deprecated to improve the core functionalities of the platform and to align with the constantly evolving ecosystem requirements.

- Support in the `VARCHAR` data type ends at September 30th, 2023.
- `VARCHAR` is no longer supported for new customers, effective from Version 2022.1.3.
- The `TEXT` data type is replacing the `VARCHAR` and `NVARCHAR` data types.

14.1.3.10 End of Support

No End of Support changes were made.

14.1.3.11 Upgrading to v4.2

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

Tip: SQream recommends storing the generated back-up locally in case needed.

SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Copy the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.1.4 Release Notes 4.3

The 4.3 release notes were released on 11/06/2023 and describe the following:

- *Compatibility Matrix*
- *New Features and Enhancements*
- *SQreamDB Studio Updates and Improvements*
- *Known Issues*
- *Version 4.3 resolved Issues*
- *Configuration Adjustments*
- *Deprecations*
- *Upgrading to v4.3*

14.1.4.1 Compatibility Matrix

System Requirement	Details
Supported OS	• CentOS - 7.x • RHEL - 7.x / 8.x
Supported Nvidia driver	CUDA version from 10.1 up to 11.4.3
Storage version	49
Driver compatibility	• JDBC 4.5.8 • ODBC 4.4.4 • NodeJS • .NET 3.0.2 • Pysqream 3.2.5 • Spark
SQream Acceleration Studio	Version 5.6.0

14.1.4.2 New Features and Enhancements

► A new *SQLoader* will enable you to load data into SQreamDB from other databases.

► Access control permissions in SQreamDB have been expanded, allowing roles to now grant and revoke access to other roles for the following:

- VIEWS
- FOREIGN TABLE
- COLUMN
- CATALOG
- SERVICE

To learn more about how and when you should use this new capability, visit *Permissions*.

► RocksDB's metadata scale-up improvements have been implemented.

14.1.4.3 SQreamDB Studio Updates and Improvements

SQream Studio version 5.6.0 has been released.

14.1.4.4 Known Issues

- Percentile is not supported for Window Functions.
- Performance degradation when using `VARCHAR` partition key in a Window Functions expression

14.1.4.5 Version 4.3 resolved Issues

SQ No.	Description
SQ-11108	Slow `COPY FROM` statements using ORC files
SQ-11804	Slow metadata optimization
SQ-12721	`maxConnectionInactivitySeconds` flag issue when executing Batch Shell Program ETLs
SQ-12799	Catalog queries may not be terminated
SQ-13112	`GRANT` query queue issue
SQ-13201	`INSERT INTO` statement error while copying data from non-clustered table to clustered table
SQ-13210, SQ-13426	Slow query execution time
SQ-13225	LoopJoin performance enhancement supports =, >, <, and <= operators
SQ-13322	Cleanup operation case-sensitivity issue
SQ-13401	The JDBC driver causes the log summery of `INSERT` statements to fail
SQ-13453	Metadata performance issue
SQ-13460	`GRANT ALL ON ALL TABLES` statement slow compilation time
SQ-13461	`WHERE` clause filter issue
SQ-13467	Snapshot issue causes metadata failure
SQ-13529	Pysqream concurrency issue
SQ-13566, SQ-13694	S3 access to bucket failure when using custom endpoint
SQ-13587	Large number of worker connections failure
SQ-13947	Unicode character issue when using Tableau
SQ-14094	Metadata server error stops workers and query queue
SQ-14268	Internal runtime memory issue
SQ-14724	Alias issue when executing `DELETE` statement
SQ-13387	Simple query slow compilation time due to metadata size

14.1.4.6 Configuration Adjustments

► You may now configure the object access style and your endpoint URL with Virtual Private Cloud (VPC) when using AWS S3.

Visit *Inserting Data Using Amazon S3* to learn more about how and when you should use these two new parameters:

- `AwsEndpointOverride`
- `AwsObjectAccessStyle`

14.1.4.7 Deprecations

► **CentOS Linux 7.x**

- As of June 2024, CentOS Linux 7.x will reach its End of Life and will not be supported by SQreamDB. This announcement provides a one-year advance notice for our users to plan for this change. We recommend users to explore migration or upgrade options to maintain ongoing support and security beyond this date.
- **REHL 8.x** is now officially supported.

► `INT96`

Due to Parquet's lack of support of the `INT96` data type, SQream has decided to deprecate this data type.

► Square Brackets `[]`

The `[]`, which are frequently used to delimit identifiers such as column names, table names, and other database objects, are officially deprecated to facilitate the use of the `ARRAY` data type. To delimit database object identifiers, use double quotes `""`.

► `VARCHAR`

The `VARCHAR` data type is deprecated to improve the core functionalities of the platform and to align with the constantly evolving ecosystem requirements.

- Support in the `VARCHAR` data type ends at September 30th, 2023.
- `VARCHAR` is no longer supported for new customers, effective from version 2022.1.3.
- The `TEXT` data type is replacing the `VARCHAR` and `NVARCHAR` data types.

14.1.4.8 Upgrading to v4.3

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.
3. Copy the recently created back-up file.
4. Replace your current metadata with the metadata you stored in the back-up file.
5. Navigate to the new SQream package bin folder.
6. Run the following command:

   ```
   $ ./upgrade_storage <levelDB path>
   ```

7. Version 4.3 introduces a service permission feature that enables superusers to grant and revoke role access to services. However, when upgrading from version 4.2 or earlier to version 4.3 or later, this feature initializes access to services and to catalog tables, causing existing roles to lose their access to services, catalog tables and consequently also to the UI (Catalog tables may also be used to determine user access rights and privileges. The UI can integrate with these permissions to control what actions users are allowed to perform in the database.).

There are two methods of granting back access to services:

- Grant access to all services for all roles using the grant_usage_on_service_to_all_roles utility function
- Selectively grant or revoke access to services by following the *access permission guide*

To grant back access to catalog tables and the UI, you may either grant access to all system roles, using your `public` role:

```
GRANT ALL PERMISSIONS ON CATALOG <catalog_name> TO public;
```

Or individually grant access to selected roles:

```
GRANT ALL PERMISSIONS ON CATALOG <catalog_name> TO <role_name>;
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.1.5 Release Notes 4.4

The 4.4 release notes were released on September 28th, 2023

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Newly Released Connector Drivers*
- *SQreamDB Studio Updates and Improvements*
- *Known Issues*
- *Version 4.4 resolved Issues*
- *Configuration Adjustments*
- *Deprecations*
- *Upgrading to Version 4.4*

14.1.5.1 Compatibility Matrix

System Requirement	Details
Supported OS	• CentOS - 7.x • RHEL - 7.x / 8.x
supported Nvidia driver	CUDA version from 10.1 up to 11.4.3
Storage version	50
Driver compatibility	• JDBC 5.0.0 • ODBC 4.4.4 • NodeJS • .NET 3.0.2 • Pysqream 5.0.0 • Spark
SQream Acceleration Studio	Version 5.7.0

14.1.5.2 New Features and Enhancements

► The newly supported *Array* data type enables you to simplify queries and optimize space utilization.

► *Denodo Platform* may now be used for real-time data visualization of various sources.

► Introducing select_data_read_metrics utility function, facilitating easy monitoring of cluster data read within specified timeframes, ensuring adherence to license limits.

► New select_gpu_metrics utility function now available, providing insights into cluster GPU usage over defined periods, crucial for maintaining compliance with license limits.

14.1.5.3 Newly Released Connector Drivers

► **Pysqream 5.0.0**

- tar.file
- *Documentation*

► **JDBC 5.0.0**

- jar.file
- *Documentation*

14.1.5.4 SQreamDB Studio Updates and Improvements

SQream Studio version 5.7.0 has been released.

14.1.5.5 Known Issues

- Percentile is not supported for Window Functions.

14.1.5.6 Version 4.4 resolved Issues

SQ No.	Description
SQ-12965	`ReadParquet` chunk producer issue
SQ-13461	`LEFT JOIN` in the `WHERE` clause with different date values results in missing filters
SQ-13772	Foreign table `JOIN` operation issue
SQ-13805	Different table structures provide different query times when using Parquet files
SQ-13947	Unicode character issue when using Tableau
SQ-13954	Runtime error when granting role multiple permissions using the web interface
SQ-13971	Parquet file data loading issue when columns contain over 100,000 digits
SQ-14136	Query deceleration due to metadata server issue
SQ-14268	`TEXT` column length calculation CUDA memory issue
SQ-14399	Figment snapshot recognition issue
SQ-14400	Healer configuration flag unavailability
SQ-14556	Object store path issue when using S3 API
SQ-14724	Aliases error when using `DELETE` statement
SQ-15074	Web interface login issue for non-`SUPERUSER` roles

14.1.5.7 Configuration Adjustments

► You may now configure the object access style and your endpoint URL with Virtual Private Cloud (VPC) when using AWS S3.

Visit *Inserting Data Using Amazon S3* to learn more about how and when you should use these two new parameters:

- `AwsEndpointOverride`
- `AwsObjectAccessStyle`

► New *Server Picker* parameters enable you to direct services to specific Workers and examine Worker availability.

14.1.5.8 Deprecations

► **CentOS Linux 7.x**

- As of June 2024, CentOS Linux 7.x will reach its End of Life and will not be supported by SQreamDB. This announcement provides a one-year advance notice for our users to plan for this change. We recommend users to explore migration or upgrade options to maintain ongoing support and security beyond this date.
- **REHL 8.x** is now officially supported.

► `INT96`

Due to Parquet's lack of support of the `INT96` data type, SQreamDB has decided to deprecate this data type.

► Square Brackets `[]`

The `[]`, which are frequently used to delimit identifiers such as column names, table names, and other database objects, are officially deprecated to facilitate the use of the `ARRAY` data type. To delimit database object identifiers, use double quotes `""`.

► `VARCHAR`

With the improvement of the core functionalities of the platform and to align with the constantly evolving ecosystem requirements, the `VARCHAR` data type is deprecated and may not be used. The `TEXT` data type is replacing the `VARCHAR` and `NVARCHAR` data types.

14.1.5.9 Upgrading to Version 4.4

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQreamDB recommends storing the generated back-up locally in case needed.

   SQreamDB runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQreamDB services.
3. Copy the recently created back-up file.
4. Replace your current metadata with the metadata you stored in the back-up file.
5. Navigate to the new SQreamDB package bin folder.
6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

7. Version 4.4 introduces a service permission feature that enables superusers to grant and revoke role access to services. However, when upgrading from version 4.2 or earlier to version 4.4 or later, this feature initializes access to services, causing existing roles to lose their access to services.

There are two methods of granting back access to services:

- Grant access to all services for all roles using the grant_usage_on_service_to_all_roles utility function
- Selectively grant or revoke access to services by following the *access permission guide*

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQreamDB Version procedure.

14.1.6 Release Notes 4.5

The 4.5 release notes were released on December 5th, 2023

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.5 resolved Issues*
- *Deprecations*
- *Upgrading to Version 4.5*

14.1.6.1 Compatibility Matrix

System Requirement	Details
Supported OS	• CentOS 7.x • RHEL 7.x / 8.x
supported Nvidia driver	CUDA version from 10.1 up to 11.4.3
Storage version	50
Driver compatibility	• JDBC 5.0.0 • ODBC 4.4.4 • NodeJS • .NET 5.0.0 • Pysqream 5.0.0 • Spark
SQream Acceleration Studio	Version 5.7.0

14.1.6.2 New Features and Enhancements

► Introducing a new Health-Check Monitor utility command empowers administrators to oversee the database's health. This command serves as a valuable tool for monitoring, enabling administrators to assess and ensure the optimal health and performance of the database

► A new Query Timeout session flag designed to identify queries that have exceeded a specified time limit. Once the flag value is reached, the query automatically stops

► Optimized `JOIN` operation for improved performance with large tables

14.1.6.3 Known Issues

- Percentile is not supported for Window Functions

14.1.6.4 Version 4.5 resolved Issues

SQ No.	Description
SQ-11523	Resolved internal runtime issue affecting the `datetime` saved query
SQ-14292	Resolved `maxConnections` Worker allocation issue
SQ-14869	Optimized compilation time for improved performance with large metadata
SQ-15074	Addressed UI access issue for non-`SUPERUSER` roles

14.1.6.5 Deprecations

► **CentOS Linux 7.x**

- As of June 2024, CentOS Linux 7.x will reach its End of Life and will not be supported by SQreamDB. This announcement provides a one-year advance notice for our users to plan for this change. We recommend users to explore migration or upgrade options to maintain ongoing support and security beyond this date.
- **REHL 8.x** is now officially supported.

14.1.6.6 Upgrading to Version 4.5

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

> **Tip:** SQreamDB recommends storing the generated back-up locally in case needed.

   SQreamDB runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQreamDB services.
3. Copy the recently created back-up file.
4. Replace your current metadata with the metadata you stored in the back-up file.
5. Navigate to the new SQreamDB package bin folder.
6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

7. Version 4.4 introduces a service permission feature that enables superusers to grant and revoke role access to services. However, when upgrading from version 4.2 or earlier to version 4.4 or later, this feature initializes access to services, causing existing roles to lose their access to services.

There are two methods of granting back access to services:

- Grant access to all services for all roles using the grant_usage_on_service_to_all_roles utility function
- Selectively grant or revoke access to services by following the *access permission guide*

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQreamDB Version procedure.

14.1.7 Release Notes 4.6

The 4.6 release notes were released on February 19, 2024

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.6 resolved Issues*
- *Deprecations*
- *Upgrading to Version 4.6*

14.1.7.1 Compatibility Matrix

System Requirement	Details
Supported OS	• CentOS - 7.x • RHEL - 7.x / 8.x
supported Nvidia driver	CUDA version from 10.1 up to 11.4.3
Storage version	51
Driver compatibility	• JDBC 5.2.0 • ODBC 4.4.4 • NodeJS • .NET 3.0.2 • Pysqream 5.0.0 • Spark 5.0.0 • SQLoader 7.12 (compatible with v4.6.1 or later)

14.1.7.2 New Features and Enhancements

► You may now retrieve and manipulate data from different databases within a single SQreamDB cluster through the execution of a single SQL statement using the Cross-Database syntax.

► Enhance your COPY FROM operations with the new `DELETE_SOURCE_ON_SUCCESS` parameter, which automatically deletes the source file being copied into SQreamDB. This not only saves time and effort in cleaning cloud storage but also helps conserve storage space.

► Safely cast data types with the new IsCastable function. This function allows you to check whether a cast operation is possible or supported for a given column and data type and provides an alternative when an exception occurs when used within a `CASE` statement.

► Enable automatic termination of queries that exceed a pre-defined time limit in the queue. The introduction of the *queueTimeoutMinutes flag* flag empowers you to set time constraints for queries in the queue, ranging from a few minutes to a maximum of 72 hours.

► You may now set your *Server Picker* more easily using keyword arguments.

► We enhanced our *Saved Query* permissions, ensuring that your saved queries are accessible and can be executed and reviewed exclusively by authorized users.

► Our *Pysqream* connector now support SQLAlchemy version 2.0.27.

14.1.7.3 Known Issues

- Percentile is not supported for Window Functions

14.1.7.4 Version 4.6 resolved Issues

SQ No.	Description
SQ-12872	Fixed unexpected Worker behavior caused by `DROP TABLE` statement
SQ-12873	Improved the time it takes to delete metadata keys
SQ-12965	Fixed `ReadParquet` chunk producer output
SQ-13057	Fixed `DOUBLE` casting into `TEXT` issue
SQ-15691	Fixed `TEXT` casting into `DOUBLE` and `NUMERIC` issue when using scientific notation
SQ-15828	Fixed slow query runtime due to `VIEW` unexpected behavior

14.1.7.5 Deprecations

► **CentOS Linux 7.x**

- As of June 2024, CentOS Linux 7.x will reach its End of Life and will not be supported by SQreamDB. This announcement provides a one-year advance notice for our users to plan for this change. We recommend users to explore migration or upgrade options to maintain ongoing support and security beyond this date.

14.1.7.6 Upgrading to Version 4.6

1. Generate a back-up of the metadata by running the following command:

```
select backup_metadata('out_path');
```

Tip: SQreamDB recommends storing the generated back-up locally in case needed.

SQreamDB runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQreamDB services.
3. Copy the recently created back-up file.
4. Replace your current metadata with the metadata you stored in the back-up file.
5. Navigate to the new SQreamDB package bin folder.
6. Run the following command:

```
./upgrade_storage <levelDB path>
```

7. Version 4.4 introduces a service permission feature that enables superusers to grant and revoke role access to services. However, when upgrading from version 4.2 or earlier to version 4.4 or later, this feature initializes access to services, causing existing roles to lose their access to services.

There are two methods of granting back access to services:

- Grant access to all services for all roles using the grant_usage_on_service_to_all_roles utility function
- Selectively grant or revoke access to services by following the *access permission guide*

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQreamDB Version procedure.

14.2 Release Notes 2022.1

The 2022.1 Release Notes describe the following releases:

14.2.1 Release Notes 2022.1.7

The 2022.1.7 release notes were released on 12/15/2022 and describe the following:

- *New Features*
- *Storage Version*
- *Known Issues*
- *Version 2022.1.7 resolved Issues*
- *Configuration Changes*
- *Naming Changes*

- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.7*

14.2.1.1 New Features

- Ingesting data from *JSON* files.

- ZLIB compression performance enhancements.

14.2.1.2 Storage Version

The storage version presently in effect is version 43.

14.2.1.3 Known Issues

Percentile is not supported for Window functions.

14.2.1.4 Version 2022.1.7 resolved Issues

SQ No.	Description
SQ-11523	`SAVED QUERY` execution internal error
SQ-11811	Missing metadata optimization when joining `TEXT` columns
SQ-12178	SQreamNet does not support the `ExecuteNonQuery` ADO.NET command

14.2.1.5 Configuration Changes

No configuration changes were made.

14.2.1.6 Naming Changes

No relevant naming changes were made.

14.2.1.7 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective from Version 2022.1.3 (September 2022).

TEXT data type is replacing VARCHAR and NVARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

14.2.1.8 End of Support

No End of Support changes were made.

14.2.1.9 Upgrading to v2022.1.7

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

   Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.2.2 Release Notes 2022.1.6

The 2022.1.6 release notes were released on 12/11/2022 and describe the following:

- *New Features*
- *Storage Version*
- *Known Issues*
- *Version 2022.1.6 resolved Issues*
- *Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*

- *Upgrading to v2022.1.6*

14.2.2.1 New Features

- *.Net Driver* now supports .NET version 6 or newer.

14.2.2.2 Storage Version

The storage version presently in effect is version 42.

14.2.2.3 Known Issues

Percentile is not supported for Window functions.

14.2.2.4 Version 2022.1.6 resolved Issues

SQ No.	Description
SQ-10160	Spotfire casting issues when reading SQream data
SQ-11295	`max_file_size` when executing `COPY_TO` is imprecise
SQ-11940, SQ-11926, SQ-11874	Known encryption issues
SQ-11975	Internal runtime error
SQ-12019	Using `PERCENTILE_DISC` function with `PARTITION BY` function causes internal error
SQ-12089	`COUNT (*)` execution fails when using foreign table
SQ-12117	Running TCPH-21 results in out of memory
SQ-12204	Possible issue when trying to INSERT Unicode data using .Net client

14.2.2.5 Configuration Changes

No configuration changes were made.

14.2.2.6 Naming Changes

No relevant naming changes were made.

14.2.2.7 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective from Version 2022.1.3 (September 2022).

TEXT data type is replacing VARCHAR and NVARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

14.2.2.8 End of Support

No End of Support changes were made.

14.2.2.9 Upgrading to v2022.1.6

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

   ```
   $ ./upgrade_storage <levelDB path>
   ```

   Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.2.3 Release Notes 2022.1.5

The 2022.1.5 release notes were released on 11/02/2022 and describe the following:

- *New Features*
- *Storage Version*
- *Known Issues*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.5*

14.2.3.1 New Features

The 2022.1.5 Release Notes include the following new features:

- `keys_evaluate` utility function enhancement - add problematic chunk ID to the function's output report
- Automatically close database client connections that have been open for 24 hours without any active statements
- `release_defunct_locks` utility function enhancement to receive new optional input parameter to specify timeout - for more details see *Lock Related Issues*.
- Metadata scale up process improvement through RocksDB configuration improvements

14.2.3.2 Storage Version

The storage version presently in effect is version 42.

14.2.3.3 Known Issues

Recently discovered issue with the encryption feature, at this time SQream recommends to avoid using this feature - a fix will be introduced in the near future.

14.2.3.4 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.5:

SQ No.	Description
SQ-11081	Tableau connection are not getting closed
SQ-11473	SQream Command Line Interface connectivity issues
SQ-11551	SQream Studio Logs pages filtering issues
SQ-11631	Log related configuration flags are not working as expected
SQ-11745	Missing validation of sufficient GPU memory
SQ-11792	CUME_DIST function causes query execution errors
SQ-11905	GetDate casting to as text returns DATE with 0s in the time part or no time part at all
SQ-12580	Server Picker and Meta Data server may not be deployed on servers without GPU
SQ-12690	Worker thread increase
SQ-13775	Worker down issue
SQ-13947	Non-Unicode character query execution error

14.2.3.5 Operations and Configuration Changes

No configuration changes were made.

14.2.3.6 Naming Changes

No relevant naming changes were made.

14.2.3.7 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective from Version 2022.1.3 (September 2022).

TEXT data type is replacing VARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

14.2.3.8 End of Support

No End of Support changes were made.

14.2.3.9 Upgrading to v2022.1.5

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.2.4 Release Notes 2022.1.4

The 2022.1.4 release notes were released on 10/11/2022 and describe the following:

- *Version Content*
- *Storage Version*
- *Known Issues*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.4*

14.2.4.1 Version Content

The 2022.1.4 Release Notes describes the following:

- Security enhancement - Disable Python UDFs by default.

14.2.4.2 Storage Version

The storage version presently in effect is version 42.

14.2.4.3 Known Issues

No relevant Known Issues.

14.2.4.4 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.4:

SQ No.	Description
SQ-11782	Alter default permissions to grant update results in error
SQ-11740	A correlated subquery is blocked when having 'not exist' where clause in update query
SQ-11686, SQ-11584	CUDA malloc error
SQ-10602	Group by clause error
SQ-9813	When executing copy from a parquet file that contain date values earlier than 1970, values are changed to 1970.

14.2.4.5 Operations and Configuration Changes

No configuration changes were made.

14.2.4.6 Naming Changes

No relevant naming changes were made.

14.2.4.7 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective from Version 2022.1.3 (September 2022).

TEXT data type is replacing VARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

14.2.4.8 End of Support

No End of Support changes were made.

14.2.4.9 Upgrading to v2022.1.4

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

Tip: SQream recommends storing the generated back-up locally in case needed.

SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.2.5 Release Notes 2022.1.3

The 2022.1.3 release notes were released on 9/20/2022 and describe the following:

- *Version Content*
- *Storage Version*
- *Known Issues*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.3*

14.2.5.1 Version Content

The 2022.1.3 Release Notes describes the following:

- Optimize the delete operation by removing redundant calls.
- Support LIKE condition for filtering metadata.
- Migration tool for converting VARCHAR columns into TEXT columns.
- Support sub-queries in the UPDATE condition.

14.2.5.2 Storage Version

The storage version presently in effect is version 42.

14.2.5.3 Known Issues

The following table lists the issues that are known limitations in Version 2022.1.3:

SQ No.	Description
SQ-11677	UPADTE or DELETE using a sub-query that includes '%' (modulo) is crashing SQreamDB worker

14.2.5.4 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.3:

SQ No.	Description
SQ-11487	COPY FROM with offset = 0 (which is an unsupported option) is stuck up to the query timeout.
SQ-11373	SQL statement fails after changing the foreign table the statement tries to query.
SQ-11320	Locked users are not being released on system reset.
SQ-11310	Using "create table like" on foreign tables results in flat compression of the created table.
SQ-11287	SQL User Defined Function fails when function definition contain parenthesis
SQ-11187	FLAT compression is wrongly chosen when dealing with data sets starting with all-nulls
SQ-10892	Update - enhanced error message when trying to run update on foreign table.

14.2.5.5 Operations and Configuration Changes

No configuration changes were made.

14.2.5.6 Naming Changes

No relevant naming changes were made.

14.2.5.7 Deprecated Features

SQream is declaring end of support of VARCHAR data type, the decision resulted by SQream's effort to enhance its core functionalities and with respect to ever changing echo system requirements.

VARCHAR is no longer supported for new customers - effective immediately.

TEXT data type is replacing VARCHAR - SQream will maintain VARCHAR data type support until 09/30/2023.

As part of this release 2022.1.3, SQream provides an automated and secured migration tool to help customers with the conversion phase from VARCHAR to TEXT data type, please address delivery for further information.

14.2.5.8 End of Support

No End of Support changes were made.

14.2.5.9 Upgrading to v2022.1.3

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

   ```
   $ ./upgrade_storage <levelDB path>
   ```

   Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.2.6 Release Notes 2022.1.2

The 2022.1.2 release notes were released on 8/24/2022 and describe the following:

- *Version Content*
- *Storage Version*
- *New Features*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.2*

14.2.6.1 Version Content

The 2022.1.2 Release Notes describes the following:

- Automatic schema identification.

- Optimized queries on external Parquet tables.

14.2.6.2 Storage Version

The storage version presently in effect is version 41.

14.2.6.3 New Features

The 2022.1.2 Release Notes include the following new features:

- *Parquet Read Optimization*

14.2.6.3.1 Parquet Read Optimization

Querying Parquet foreign tables has been optimized and is now up to 20x faster than in previous versions.

14.2.6.4 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.2:

SQ No.	Description
SQ-10892	An incorrect error message was displayed when users ran the `UPDATE` command on foreign tables.
SQ-11273	Clustering optimization only occurs when copying data from CSV files.

14.2.6.5 Operations and Configuration Changes

No configuration changes were made.

14.2.6.6 Naming Changes

No relevant naming changes were made.

14.2.6.7 Deprecated Features

No features were deprecated for Version 2022.1.2.

14.2.6.8 End of Support

The End of Support section is not relevant to Version 2022.1.2.

14.2.6.9 Upgrading to v2022.1.2

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   Tip: SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the *Upgrading SQream Version* procedure.

14.2.7 Release Notes 2022.1.1

The 2022.1.1 release notes were released on 7/19/2022 and describe the following:

- *Version Content*
- *Storage Version*
- *New Features*
- *Known Issues*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.1*

14.2.7.1 Version Content

The 2022.1.1 Release Notes describes the following:

- Enhanced security features
- For more information, see *SQream Acceleration Studio 5.4.7*.

14.2.7.2 Storage Version

The storage version presently in effect is version 40.

14.2.7.3 New Features

The 2022.1.1 Release Notes include the following new features:

- *Password Security Compliance*

14.2.7.3.1 Password Security Compliance

In compliance with GDPR standards, SQream now requires a strong password policy when accessing the CLI or Studio.

For more information, see *Password Policy*.

14.2.7.4 Known Issues

There were no known issues in Version 2022.1.1.

14.2.7.5 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.1:

SQ No.	Description
SQ-6419	An internal compiler error occurred when casting Numeric literals in an aggregation function.
SQ-10873	Inserting 100K bytes into a text column resulted in an unclear error message.
SQ-10955	Unneeded reads were occurring when filtering by date.

14.2.7.6 Operations and Configuration Changes

The `login_max_retries` configuration flag is required for adjusting the permitted log-in attempts.

For more information, see Adjusting the Permitted Log-In Attempts.

14.2.7.7 Naming Changes

No relevant naming changes were made.

14.2.7.8 Deprecated Features

In *SQream Acceleration Studio 5.4.7*, the **Configuration** section has been temporarily disabled and will be enabled at a later date. In addition, the **Log Lines** tab in the **Log** section has been removed.

14.2.7.9 End of Support

The End of Support section is not relevant to Version 2022.1.1.

14.2.7.10 Upgrading to v2022.1.1

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

> **Tip:** SQream recommends storing the generated back-up locally in case needed.

SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the *Upgrading SQream Version* procedure.

14.2.8 Release Notes 2022.1

The 2022.1 release notes were released on 7/19/2022 and describe the following:

- *Version Content*
- *Storage Version*
- *New Features*
- *Known Issues*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1*

14.2.8.1 Version Content

The 2022.1 Release Notes describe the following:

- Enhanced security features.
- New data manipulation command.
- Additional data ingestion format.

14.2.8.2 Storage Version

The storage version presently in effect is version 40.

14.2.8.3 New Features

The 2022.1 Release Notes include the following new features:

- *Data Encryption*
- *Update Feature*
- *Avro Ingestion*

14.2.8.3.1 Data Encryption

SQream now supports data encryption mechanisms in accordance with **General Data Protection Regulation (GDPR)** standards.

Using the data encryption feature may lead to a maximum of a 10% increase in performance degradation.

For more information, see Data Encryption.

14.2.8.3.2 Update Feature

SQream now supports the DML **Update** feature, which is used for modifying the value of certain columns in existing rows.

For more information, see UPDATE.

14.2.8.3.3 Avro Ingestion

SQream now supports ingesting data from Avro files.

For more information, see Inserting Data from Avro.

14.2.8.4 Known Issues

The following table lists the known issues for Version 2022.1:

SQ No.	Description
SQ-7732	Reading numeric columns from an external Parquet file generated an error.
SQ-9889	Running a query including Thai characters generated an internal runtime error.
SQ-10071	Error on existing subqueries with TEXT and VARCHAR equality condition
SQ-10191	The `ALTER DEFAULT SCHEMA` command was not functioning correctly.
SQ-10629	Inserting data into a table significantly slowed down running queries.
SQ-10659	Using a comment generated a compile error.

14.2.8.5 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1:

SQ No.	Description
SQ-10111	Reading numeric columns from an external Parquet file generated an error.

14.2.8.6 Operations and Configuration Changes

No relevant operations and configuration changes were made.

14.2.8.7 Naming Changes

No relevant naming changes were made.

14.2.8.8 Deprecated Features

In SQream version 2022.1 the `VARCHAR` data type has been deprecated and replaced with `TEXT`. SQream will maintain `VARCHAR` in all previous versions until completing the migration to `TEXT`, at which point it will be deprecated in all earlier versions. SQream also provides an automated and secure tool to facilitate and simplify migration from `VARCHAR` to `TEXT`.

If you are using an earlier version of SQream, see the Using Legacy String Literals configuration flag.

14.2.8.9 End of Support

The End of Support section is not relevant to Version 2022.1.

14.2.8.10 Upgrading to v2022.1

1. Generate a backup of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path', 'single_file');
   ```

   Tip: SQream recommends storing the generated backup locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created backup file.

4. Replace your current metadata with the metadata you stored in the backup file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

Note: Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

14.3 Release Notes 2021.2

The 2021.2 Release Notes describe the following releases:

14.3.1 Release Notes 2021.2.1.24

The 2021.2.1.24 release notes were released on 7/28/2022 and describe the following:

- *Version Content*
- *New Features*
- *Resolved Issues*
- *Known Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*

14.3.1.1 Version Content

The 2021.2.1.24 Release Notes includes a query maintenance feature.

14.3.1.2 New Features

The 2021.2.1.24 Release Notes include the following new features:

- *Query Healer*

14.3.1.2.1 Query Healer

The new **Query Healer** feature periodically examines the progress of running statements, and is used for query maintenance.

For more information, see Query Healer.

14.3.1.3 Resolved Issues

The following table lists the resolved issues for Version 2021.2.1.24:

SQ No.	Description
SQ-10606	Queries were getting stuck in the queue for a prolonged time.
SQ-10691	The DB schema identifier was causing an error when running queries from joins suite.
SQ-10918	The Workload Manager was only assigning jobs sequentially, delaying user SQLs assigned to workers running very large jobs.
SQ-10955	Metadata filters were not being applied when users filtered by nullable dates using `dateadd`

14.3.1.4 Known Issues

The following table lists the known issues for Version 2021.2.1.24:

SQ No.	Description
SQ-10071	An error occurred on existing subqueries with `TEXT` and `VARCHAR` equality conditions.
SQ-10902	Inserting a null value into non-null column was causing SQream to crash.
SQ-11088	Specific workers caused low performance during compilation.

14.3.1.5 Operations and Configuration Changes

The following worker level configuration flags were added:

- is_healer_on
- healer_max_statement_inactivity_seconds
- healer_detection_frequency_seconds

14.3.1.6 Naming Changes

No relevant naming changes were made.

14.3.1.7 Deprecated Features

Version 2021.2.1.24 includes no deprecated features.

14.3.1.8 End of Support

The End of Support section is not relevant to Version 2021.2.1.24.

14.3.2 Release Notes 2021.2.1

The 2021.2.1 release notes were released on 15/12/2021 and describes the following:

- *New Features*
- *Performance Enhancements*
- *Resolved Issues*
- *Known Issues*
- *Naming Convention Modifications*
- *End of Support*
- *Deprecated Features*

14.3.2.1 New Features

The 2021.2.1 Release Notes include the following new features:

- *CREATE TABLE*
- *PERCENTILE FUNCTIONS*
- *REGEX REPLACE*
- *Delete Optimization*

14.3.2.1.1 CREATE TABLE

SQream now supports duplicating the column structure of an existing table using the `LIKE` clause.

For more information, see Duplicating the Column Structure of an Existing Table.

14.3.2.1.2 PERCENTILE FUNCTIONS

SQream now supports the following aggregation functions:

- PERCENTILE_CONT
- PERCENTILE_DISC
- MODE

14.3.2.1.3 REGEX REPLACE

SQream now supports the `REGEXP_REPLACE` function for finding and replacing text column substrings.

For more information, see REGEX_REPLACE.

14.3.2.1.4 Delete Optimization

The `DELETE` statement can now delete values that contain multi-table conditions.

For more information, see Deleting Values that Contain Multi-Table Conditions.

For more information, see REGEX_REPLACE.

14.3.2.2 Performance Enhancements

The **Performance Enhancements** section is not relevant to Version 2021.2.1.

14.3.2.3 Resolved Issues

The following table lists the issues that were resolved in Version 2021.2.1:

SQ No.	Description
SQ-8267	A method has been provided for including the `GROUP BY` and `DISTINCT COUNT` statements.

14.3.2.4 Known Issues

The **Known Issues** section is not relevant to 2021.2.1.

14.3.2.5 Naming Convention Modifications

The **Naming Convention Modifications** section is not relevant to Version 2021.2.1.

14.3.2.6 End of Support

The **End of Support** section is not relevant to Version 2021.2.1.

14.3.2.7 Deprecated Features

The **Deprecated Components** section is not relevant to Version 2021.2.1.

14.3.3 Release Notes 2021.2

The 2021.2 release notes were released on 13/9/2021.

- *New Features*
- *Performance Enhancements*
- *Resolved Issues*
- *Known Issues*
- *Naming Convention Modifications*
- *End of Support*
- *Deprecated Features*
- *Upgrading Your SQream Version*

14.3.3.1 New Features

The 2021.2 Release Notes include the following new features:

- *New Driver Compatibility*
- *Centralized Configuration System*
- *Qualifying Schemas Without Providing an Alias*
- *Double-Quotations Supported When Importing and Exporting CSVs*

14.3.3.1.1 New Driver Compatibility

The 2021.2 release supports the following drivers:

- **JDBC** - new driver version (JDBC 4.5) with important bug fixes.
- **ODBC** - ODBC 4.1.1. available on request.
- **NodeJS** - all versions starting with NodeJS 4.0. SQream recommends the latest version (NodeJS 4.2.4).
- **Dot Net** - SQream recommends version version 3.02 (compatible with DotNet version 48).
- **Pysqream** - pysqream 3.1.2

14.3.3.1.2 Centralized Configuration System

SQream now uses a new configuration system based on centralized configuration accessible from SQream Studio.

For more information, see the following:

- *Configuration <https://docs.sqream.com/en/v2021.2/guides/operations/configuration.html>* - describes how to configure your instance of SQream from a centralized location.
- SQream Studio 5.4.3 - configure your instance of SQream from Studio.

14.3.3.1.3 Qualifying Schemas Without Providing an Alias

When running queries, SQream now supports qualifying schemas without providing an alias.

For more information, see SQream Studio 5.4.3.

14.3.3.1.4 Double-Quotations Supported When Importing and Exporting CSVs

When importing and exporting CSVs, SQream now supports using quotation characters other than double quotation marks (`"`).

For more information, see the following:

- COPY_FROM
- COPY_TO

Note the following:

- Leaving *<x>* unspecified uses the default value of standard double quotations `"`.
- The quotation character must be a single, 1-byte printable ASCII character. The same octal syntax of the copy command can be used.
- The quote character cannot be contained in the field delimiter, record delimiter, or null marker.
- Double-quotations can be customized when the `csv_fdw` value is used with the `COPY FROM` and `CREATE FOREIGN TABLE` statements.

- The default escape character always matches the quote character, and can be overridden by using the `ESCAPE = {'\\' | E'\XXX'}` syntax as shown in the following examples:

```
copy t from wrapper csv_fdw options (location = '/tmp/file.csv', escape='\\');
```

```
copy t from wrapper csv_fdw options (location = '/tmp/file.csv', escape=E'\017');
```

```
copy t to wrapper csv_fdw options (location = '/tmp/file.csv', escape='\\');
```

For more information, see the following statements:

- COPY_FROM
- CREATE_FOREIGN_TABLE

14.3.3.2 Performance Enhancements

In Version 2021.2, an advanced smart spooling mechanism splits spool memory based on required CP usage.

14.3.3.3 Resolved Issues

The following table lists the issues that were resolved in Version 2021.2:

SQ No.	Description
SQ-8294	Quote qualifiers were not present in exported file, preventing it from being reloaded.
SQ-8288	Saved `TEXT` query parameters were not supported.
SQ-8266	A data loading issue occurred related to column order.

14.3.3.4 Known Issues

The **Known Issues** section is not relevant to Version 2021.2.

14.3.3.5 Naming Convention Modifications

The **Naming Convention Modifications** describes SQream features, such as data types or statements, that have been renamed.

14.3.3.5.1 NVARCHAR Data Type Renamed TEXT

The `NVARCHAR` data type has been renamed `TEXT`.

For more information on the `TEXT` data type, see String (TEXT)

14.3.3.6 End of Support

The **End of Support** section is not relevant to Version 2021.2.

14.3.3.7 Deprecated Features

The **Deprecated Components** section is not relevant to Version 2021.2.

14.3.3.8 Upgrading Your SQream Version

The **Upgrading Your SQream Version** section describes the following:

- *Upgrading Your Storage Version*
- *Upgrading Your Client Drivers*
- *Configuring Your Instance of SQream*

14.3.3.8.1 Upgrading Your Storage Version

When upgrading from a SQream version earlier than 2021.2 you must upgrade your storage version, as shown in the following example:

```
$ cat /etc/sqream/sqream1_config.json |grep cluster
$ ./upgrade_storage <cluster path>
```

For more information on upgrading your SQream version, see Upgrading SQream Version.

14.3.3.8.2 Upgrading Your Client Drivers

For more information on the client drivers for version 2021.2, see Client Drivers for 2021.2.

14.3.3.8.3 Configuring Your Instance of SQream

A new configuration method is used starting with Version 2021.2.

For more information about configuring your instance of SQream, see Client Drivers for 2021.2.

CHAPTER

FIFTEEN

TROUBLESHOOTING

The **Troubleshooting** page describes solutions to the following issues:

15.1 Remedying Slow Queries

The **Remedying Slow Queries** page describes how to troubleshoot the causes of slow queries.

The following table is a checklist you can use to identify the cause of your slow queries:

Step	Description	Results
1	A single query is slow	If a query isn't performing as you expect, follow the *Query best practices* part of the *Optimization and Best Practices* guide. If all queries are slow, continue to step 2.
2	All queries on a specific table are slow	1. If all queries on a specific table aren't performing as you expect, follow the *Table design best practices* part of the *Optimization and Best Practices* guide. 2. Check for active delete predicates in the table. Consult the *Deleting Data* guide for more information. If the problem spans all tables, continue to step 3.
3	Check that all workers are up	Use `SELECT show_cluster_nodes();` to list the active cluster workers. If the worker list is incomplete, locate and start the missing worker(s). If all workers are up, continue to step 4.
4	Check that all workers are performing well	1. Identify if a specific worker is slower than others by running the same query on different workers. (e.g. by connecting directly to the worker or through a service queue) 2. If a specific worker is slower than others, investigate performance issues on the host using standard monitoring tools (e.g. `top`). 3. Restart SQream DB workers on the problematic host. If all workers are performing well, continue to step 5.
5	Check if the workload is balanced across all workers	1. Run the same query several times and check that it appears across multiple workers (use `SELECT show_server_status()` to monitor) 2. If some workers have a heavier workload, check the service queue usage. Refer to the *Workload Manager* guide. If the workload is balanced, continue to step 6.
6	Check if there are long running statements	1. Identify any currently running statements (use `SELECT show_server_status()` to monitor)

15.2 Resolving Common Issues

The **Resolving Common Issues** page describes how to resolve the following common issues:

15.2.1 Troubleshooting Cluster Setup and Configuration

1. Note any errors - Make a note of any error you see, or check the *logs* for errors you might have missed.
2. If SQream DB can't start, start SQream DB on a new storage cluster, with default settings. If it still can't start, there could be a driver or hardware issue. *Contact SQream support*.
3. Reproduce the issue with a standalone SQream DB - starting up a temporary, standalone SQream DB can isolate the issue to a configuration issue, network issue, or similar.
4. Reproduce on a minimal example - Start a standalone SQream DB on a clean storage cluster and try to replicate the issue if possible.

15.2.2 Troubleshooting Connectivity Issues

1. Verify the correct login credentials - username, password, and database name.
2. Verify the host name and port
3. Try connecting directly to a SQream DB worker, rather than via the load balancer
4. Verify that the driver version you're using is supported by the SQream DB version. Driver versions often get updated together with major SQream DB releases.
5. Try connecting directly with *the built in SQL client*. If you can connect with the local SQL client, check network availability and firewall settings.

15.2.3 Troubleshooting Query Performance

1. Use show_node_info to examine which building blocks consume time in a statement. If the query has finished, but the results are not yet materialized in the client, it could point to a problem in the application's data buffering or a network throughput issue. Alternatively, you may also *retrieve the query execution plan output* using SQreamDB Studio.
2. If a problem occurs through a 3rd party client, try reproducing it directly with *the built in SQL client*. If the performance is better in the local client, it could point to a problem in the application or network connection.
3. Consult the *Optimization and Best Practices* guide to learn how to optimize queries and table structures.

15.2.4 Troubleshooting Query Behavior

1. Consult the *SQL Statements and Syntax* reference to verify if a statement or syntax behaves correctly. SQream DB may have some differences in behavior when compared to other databases.
2. If a problem occurs through a 3rd party client, try reproducing it directly with *the built in SQL client*. If the problem still occurs, file an issue with SQream support.

15.2.5 File an issue with SQream support

To file an issue, follow our *Gathering Information for SQream Support* guide.

15.3 Identifying Configuration Issues

The **Troubleshooting Common Issues** page describes how to troubleshoot the following common issues:

Starting a SQream DB temporarily (not as part of a cluster, with default settings) can be helpful in identifying configuration issues.

Example:

```
$ sqreamd /home/rhendricks/raviga_database 0 5000 /home/sqream/.sqream/license.enc
```

Tip:

- Using `nohup` and `&` sends SQream DB to run in the background.
- It is safe to stop SQream DB at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

```
$ kill -9 $SQREAM_PID
```

15.4 Lock Related Issues

Sometimes, a rare situation can occur where a lock is never freed.

The workflow for troubleshooting locks is:

1. Identify which statement has obtained locks.
2. Understand if the statement is itself stuck, or waiting for another statement.
3. Try to stop the offending statement, as in the following example:

```
SELECT STOP_STATEMENT(2484923);
```

15.5 Log Related Issues

The **Log Related Issues** page describes how to resolve the following common issues:

15.5.1 Loading Logs with Foreign Tables

Assuming logs are stored at `/home/rhendricks/sqream_storage/logs/`, a database administrator can access the logs using the external_tables concept through SQream DB.

```
CREATE FOREIGN TABLE logs
(
  start_marker      TEXT(4),
  row_id            BIGINT,
  timestamp         DATETIME,
  message_level     TEXT,
  thread_id         TEXT,
  worker_hostname   TEXT,
  worker_port       INT,
  connection_id     INT,
  database_name     TEXT,
  user_name         TEXT,
  statement_id      INT,
  service_name      TEXT,
  message_type_id   INT,
  message           TEXT,
  end_message       TEXT(5)
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
     DELIMITER = '|',
     CONTINUE_ON_ERROR = true
  )
;
```

For more information, see Loading Logs with Foreign Tables.

15.5.2 Counting Message Types

```
t=> SELECT message_type_id, COUNT(*) FROM logs GROUP BY 1;
message_type_id | count
----------------+----------
              0 |         9
              1 |      5578
              4 |      2319
             10 |      2788
             20 |       549
             30 |       411
             31 |      1720
             32 |      1720
            100 |      2592
            101 |      2598
            110 |      2571
            200 |        11
            500 |       136
           1000 |        19
           1003 |        19
           1004 |        19
           1010 |         5
```

15.5.3 Finding Fatal Errors

```
t=> SELECT message FROM logs WHERE message_type_id=1010;
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/rocksdb/LOCK: Resource temporarily unavailable
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/rocksdb/LOCK: Resource temporarily unavailable
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/LOCK: Resource temporarily unavailable
```

15.5.4 Countng Error Events Within a Certain Timeframe

```
t=> SELECT message_type_id,
.          COUNT(*)
.   FROM logs
.   WHERE message_type_id IN (1010,500)
.   AND timestamp BETWEEN '2019-12-20' AND '2020-01-01'
.   GROUP BY 1;
message_type_id | count
----------------+------
            500 |    18
           1010 |     3
```

15.5.5 Tracing Errors to Find Offending Statements

If we know an error occured, but don't know which statement caused it, we can find it using the connection ID and statement ID.

```
t=> SELECT connection_id, statement_id, message
.     FROM logs
.     WHERE message_level = 'ERROR'
.     AND timestamp BETWEEN '2020-01-01' AND '2020-01-06';
connection_id | statement_id | message
--------------+--------------+-------------------------------------------------------
↪-----------------------------------------------------------------------------------
↪----------------------
           79 |           67 | Column type mismatch, expected UByte, got INT64 on␣
↪column Number, file name: /home/sqream/nba.parquet
```

Use the `connection_id` and `statement_id` to narrow down the results.

```
t=>   SELECT database_name, message FROM logs
.       WHERE connection_id=79 AND statement_id=67 AND message_type_id=1;
database_name | message
--------------+--------------------------
master        | Query before parsing
master        | SELECT * FROM nba_parquet
```

15.6 Core Dumping Related Issues

The **Core Dumping Related Issues** page describes the troubleshooting procedure to be followed if all parameters have been configured correctly, but the cores have not been created.

To troubleshoot core dumping:

1. Reboot the server.
2. Verify that you have folder permissions:

```
$ sudo chmod -R 777 /tmp/core_dumps
```

3. Verify that the limits have been set correctly:

```
$ ulimit -c
```

   If all parameters have been configured correctly, the correct output is:

```
$ unlimited
```

4. If all parameters have been configured correctly, but running **ulimit -c** outputs **0**, run the following:

```
$ sudo vim /etc/profile
```

5. Search for line and tag it with the **hash** symbol:

```
$ ulimit -S -c 0 > /dev/null 2>&1
```

6. If the line is not found in **/etc/profile** directory, do the following:
   a. Run the following command:

```
$ sudo vim /etc/init.d/functions
```

   b. Search for the following:

```
$ ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>&1
```

   c. If the line is found, tag it with the **hash** symbol and reboot the server.

15.7 Retrieving Execution Plan Output Using SQreamDB Studio

You may use SQreamDB Studio to create a query plan snapshot to be used for monitoring and troubleshooting slow running statements and for identifying long-running execution Workers (components that process data), that may cause performance issues.

15.7.1 Retrieving Execution Plan Output

You can retrieve the execution plan output either after the query execution has completed, in the case of a hanging query, or if you suspect no progress is being made.

1. In the **Result Panel**, select **Execution Details View**.

   The **Execution Tree** window opens.

2. From the upper-right corner, select the ↓ to download a CSV execution plan table.
3. Save the execution plan on your local machine.

You can analyze this information using *Monitoring Query Performance* or with assistance from SQreamDB Support.

15.8 Gathering Information for SQream Support

- *Getting Support and Reporting Bugs*
- *How SQream Debugs Issues*
- *Collecting a Reproducible Example of a Problematic Statement*
- *Collecting Logs and Metadata Database*
- *Using the Command Line Utility:*

15.8.1 Getting Support and Reporting Bugs

When contacting SQreamDB Support, we recommend reporting the following information:

- What is the problem encountered?
- What was the expected outcome?
- How can SQream reproduce the issue?

When possible, please attach as many of the following:

- Error messages or result outputs
- DDL and queries that reproduce the issue
- *Log files*
- Screen captures if relevant
- *Execution plan output*

15.8.2 How SQream Debugs Issues

15.8.2.1 Reproduce

If we are able to easily reproduce your issue in our testing lab, this greatly improves the speed at which we can fix it.

Reproducing an issue consists of understanding:

1. What was SQream DB doing at the time?
2. How is the SQream DB cluster configured?
3. How does the schema look?
4. What is the query or statement that exposed the problem?
5. Were there any external factors? (e.g. Network disconnection, hardware failure, etc.)

See the *Collecting a Reproducible Example of a Problematic Statement* section ahead for information about collecting a full reproducible example.

15.8.2.2 Logs

The logs produced by SQream DB contain a lot of information that may be useful for debugging.

Look for *error messages in the log and the offending statements*. SQream's support staff are experienced in correlating logs to workloads, and finding possible problems.

See the *Collecting Logs and Metadata Database* section ahead for information about collecting a set of logs that can be analyzed by SQream support.

15.8.2.3 Fix

Once we have a fix, this can be issued as a hotfix to an existing version, or as part of a bigger major release.

Your SQream account manager will keep you up-to-date about the status of the issue.

15.8.3 Collecting a Reproducible Example of a Problematic Statement

SQream DB contains an SQL utility that can help SQream support reproduce a problem with a query or statement.

This utility compiles and executes a statement, and collects the relevant data in a small database which can be used to recreate and investigate the issue.

15.8.3.1 SQL Syntax

```
SELECT EXPORT_REPRODUCIBLE_SAMPLE(output_path, query_stmt [, ... ])
;

output_path ::=
   filepath
```

15.8.3.2 Parameters

Parameter	Description
`output_path`	Path for the output archive. The output file will be a tarball.
`query_stmt [, ...]`	Statements to analyze.

15.8.3.3 Example

```
SELECT EXPORT_REPRODUCIBLE_SAMPLE('/home/rhendricks', 'SELECT * FROM t', $$SELECT
↪"Name", "Team" FROM nba$$);
```

15.8.4 Collecting Logs and Metadata Database

SQream DB comes bundled with a data collection utility and an SQL utility intended for collecting logs and additional information that can help SQream support drill down into possible issues.

See more information in the *Collect logs from your cluster* section of the *Logging* guide.

15.8.4.1 Examples

Write an archive to `/home/rhendricks`, containing log files:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'log')
;
```

Write an archive to `/home/rhendricks`, containing log files and metadata database:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'db_and_log')
;
```

15.8.5 Using the Command Line Utility:

```
$ ./bin/report_collection /home/rhendricks/sqream_storage /home/rhendricks db_and_log
```

CHAPTER

SIXTEEN

GLOSSARY

The following table shows the **Glossary** descriptions:

Term	Description
Authentication	The process of verifying identity by validating a user or role identity using a username and password.
Authorization	Defines the set of actions that an authenticaed role can perform after gaining access to the system.
Catalog	A set of views containing metadata information about objects in a database.
Cluster	A SQream deployment containing several workers running on one or more nodes.
Custom connector	When SQream is integrated with Power BI, used for running direct queries.
Direct query	A Power BI data extraction method that retrieves data from a remote source instead of from a local repository.
Import	A Power BI data extraction method that retrieves data to local repository to be visualized at a later point.
Metadata	SQream's internal storage containing details about database objects.
Node	A machine used to run SQream workers.
Role	A group or a user. For more information see *SQream Studio*.
Storage cluster	The directory where SQream stores data.
Worker	A SQream application that responds to statements. Several workers running on one or more nodes form a cluster.