SQreamDB

Release 4.11

SQreamDB Documentation

Aug 24, 2025

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 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 . . . 21
2.1.1.3.3.4 Installing Recommended Tools . . . 22
2.1.1.3.3.5 Installing NodeJS . . . 22
2.1.1.3.4 Configuring Chrony for RHEL8 Only . . . 24
2.1.1.3.5 Configuring the Server to Boot Without Linux GUI . . . 24
2.1.1.3.6 Configuring the Security Limits . . . 24
2.1.1.3.7 Configuring the Kernel Parameters . . . 24
2.1.1.3.8 Configuring the Firewall . . . 25
2.1.1.3.9 Disabling SELinux . . . 25
2.1.1.3.10 Configuring the `/etc/hosts` File . . . 26
2.1.1.4 Installing the NVIDIA CUDA Driver . . . 26
2.1.1.4.1 Before You Begin . . . 26
2.1.1.4.2 Updating the Kernel Headers . . . 27
2.1.1.4.3 Disabling Nouveau . . . 27
2.1.1.4.4 Installing the CUDA Driver . . . 27
2.1.1.4.4.1 Installing the CUDA Driver from the Repository . . . 28
2.1.1.4.4.2 Tuning Up NVIDIA Performance . . . 28
2.1.1.4.4.3 Tune Up NVIDIA Performance when Driver Installed from the Repository . . . 28
2.1.1.4.4.4 Tune Up NVIDIA Performance when Driver Installed from the Runfile 29
2.1.1.5 Enabling Core Dumps . . . 29
2.1.1.5.1 Checking the `abrtd` Status . . . 30
2.1.1.5.2 Setting the Limits . . . 30
2.1.1.5.3 Creating the Core Dump Directory . . . 30
2.1.1.5.4 Setting the Output Directory on the `/etc/sysctl.conf` File . . . 30
2.1.1.5.5 Verifying that the Core Dumps Work . . . 31
2.1.1.5.6 Verify Your SQreamDB Installation . . . 31
2.1.1.5.7 Troubleshooting Core Dumping . . . 32
2.1.2 Installing SQream Using Binary Packages . . . 32
2.1.3 Installing Monit . . . 35
2.1.3.1 Getting Started . . . 35
2.1.3.2 Installing Monit on RHEL: . . . 35
2.1.3.3 Building Monit . . . 35
2.1.3.3.1 Building Monit from Source Code . . . 35
2.1.3.3.2 Building Monit from Pre-Built Binaries . . . 36
2.1.3.4 Configuring Monit . . . 36
2.1.3.5 Starting Monit . . . 37
2.1.4 Launching SQream with Monit . . . 38
2.1.4.1 Launching SQream . . . 38
2.1.4.2 Monit Usage Examples . . . 40
2.1.4.2.1 Stopping Monit and SQream Separately . . . 40
2.1.4.2.2 Stopping SQream Using a Monit Command . . . 41
2.1.4.2.3 Monit Command Line Options . . . 41
2.1.4.3 Using Monit While Upgrading Your Version of SQream . . . 41
2.2 Installing SQream Studio . . . 42
2.2.1 Installing Studio on a Stand-Alone Server . . . 42
2.2.1.1 Before You Begin . . . 43
2.2.1.2 Installing Studio . . . 43
2.2.1.2.1 Starting Studio . . . 44
2.2.1.2.2 Accessing Studio . . . 45
2.2.1.2.3 Maintaining Studio with the Process Manager (PM2) . . . 45
2.2.1.2.4 Upgrading Studio . . . 45

2.2.2 Installing an NGINX Proxy Over a Secure Connection . . . 46
2.2.2.1 Prerequisites . . . 47
2.2.2.2 Installing NGINX and Adjusting the Firewall . . . 47
2.2.2.3 Creating Your SSL Certificate . . . 48
2.2.2.4 Configuring NGINX to use SSL . . . 49
2.2.2.5 Redirecting Studio Access from HTTP to HTTPS . . . 51
2.2.2.6 Activating Your NGINX Configuration . . . 52
2.2.2.7 Verifying that NGINX is Running . . . 52
2.3 Upgrade Guides . . . 53
2.3.1 Version Upgrade . . . 53
2.3.2 Upgrade-Related Configuration Changes . . . 56

3 SQreamDB on AWS **57**
3.1 Before You Begin . . . 57
3.2 Usage Notes . . . 57
3.3 Configuration on AWS . . . 58
3.4 License . . . 58
3.5 Connecting to the Machine . . . 59
3.5.1 Connecting Using the CLI . . . 59
3.6 Connecting to SQreamDB . . . 59
3.6.1 Connection Troubleshooting . . . 59
3.7 Adding a Signed Certificate to the Cluster . . . 59

4 Operational Guides **61**
4.1 Access Control . . . 61
4.1.1 Overview . . . 61
4.1.2 Password Policy . . . 62
4.1.2.1 Password Strength Requirements . . . 62
4.1.2.2 Brute Force Prevention . . . 63
4.1.3 Managing Roles . . . 63
4.1.3.1 Creating New Roles (Users) . . . 63
4.1.3.2 Dropping a User . . . 64
4.1.3.3 Altering a User Name . . . 64
4.1.3.4 Changing a User Password . . . 64
4.1.3.5 Altering Public Role Permissions . . . 64
4.1.3.6 Altering Role Membership (Groups) . . . 65
4.1.4 Permissions . . . 65
4.1.4.1 Syntax . . . 67
4.1.4.1.1 GRANT . . . 67
4.1.4.1.2 REVOKE . . . 69
4.1.4.1.3 Altering Default Permissions . . . 71
4.1.4.2 Examples . . . 72
4.1.4.2.1 GRANT . . . 72
4.1.4.2.2 REVOKE . . . 72
4.1.5 Departmental Example . . . 73
4.1.5.1 Setting up the department permissions . . . 74
4.1.5.2 Creating new users in the departments . . . 75
4.2 Creating or Cloning Storage Clusters . . . 76
4.2.1 Creating a new storage cluster . . . 76
4.2.2 Tell SQream DB to use this storage cluster . . . 77
4.2.2.1 Permanently setting the storage cluster setting . . . 77
4.2.2.2 Start a temporary SQream DB worker with a storage cluster . . . 78
4.2.2.2.1 Using a configuration file (recommended) . . . 78
4.2.2.2.2 Using the command line parameters . . . 78

4.2.3 Copying an existing storage cluster . . . 78
4.3 Working with External Data . . . 78
4.4 Foreign Tables . . . 79
4.4.1 Supported Data Formats . . . 79
4.4.2 Supported Data Staging . . . 79
4.4.3 Using Foreign Tables . . . 80
4.4.3.1 Planning for Data Staging . . . 80
4.4.3.2 Creating a Foreign Table . . . 80
4.4.3.3 Querying Foreign Tables . . . 80
4.4.3.4 Modifying Data from Staging . . . 81
4.4.3.5 Converting a Foreign Table to a Standard Database Table . . . 82
4.4.4 Error Handling and Limitations . . . 82
4.5 Health Monitoring . . . 83
4.5.1 Before You Begin . . . 83
4.5.2 Installation . . . 83
4.5.2.1 Grafana . . . 83
4.5.2.1.1 Disabling SELinux . . . 83
4.5.2.1.2 Installing Grafana via YUM Repository . . . 84
4.5.2.1.3 Enabling the Grafana Service . . . 84
4.5.2.1.4 Modifying your Firewall . . . 85
4.5.2.2 Prometheus . . . 85
4.5.2.3 Loki and Promtail . . . 87
4.5.2.4 Exporters . . . 88
4.5.2.4.1 CPU Exporter . . . 88
4.5.2.4.2 Process Exporter . . . 89
4.5.3 Deployment . . . 89
4.5.3.1 Grafana . . . 89
4.5.4 Using the Monitor Service . . . 90
4.5.4.1 Configuring the Monitor Service Worker . . . 90
4.5.4.2 Execution Arguments . . . 90
4.5.4.2.1 Example . . . 91
4.5.4.3 Monitor Service Output Example . . . 91
4.6 Deleting Data . . . 92
4.6.1 The Deletion Process . . . 92
4.6.2 Usage Notes . . . 92
4.6.2.1 General Notes . . . 92
4.6.2.2 Clean-Up Operations Are I/O Intensive . . . 93
4.6.3 Examples . . . 93
4.6.3.1 Deleting Rows from a Table . . . 94
4.6.3.2 Deleting Values Based on Complex Predicates . . . 94
4.6.3.3 Identifying and Cleaning Up Tables . . . 95
4.6.3.3.1 Triggering a Clean-Up . . . 95
4.6.4 Best Practice . . . 96
4.7 Logging . . . 96
4.7.1 Locating the Log Files . . . 96
4.7.1.1 Log Structure and Contents . . . 97
4.7.1.2 Log-Naming . . . 99
4.7.2 Log Control and Maintenance . . . 99
4.7.2.1 Changing Log Verbosity . . . 99
4.7.2.2 Changing Log Rotation . . . 99
4.7.3 Collecting Logs from Your Cluster . . . 100
4.7.3.1 SQL Syntax . . . 100
4.7.3.2 Command Line Utility . . . 100
4.7.3.3 Parameters . . . 100

4.7.3.4 Example . . . 100
4.7.4 Troubleshooting with Logs . . . 101
4.7.4.1 Loading Logs with Foreign Tables . . . 101
4.7.4.2 Counting Message Types . . . 101
4.7.4.3 Finding Fatal Errors . . . 102
4.7.4.4 Countng Error Events Within a Certain Timeframe . . . 102
4.7.4.5 Tracing Errors to Find Offending Statements . . . 102
4.8 Query Split . . . 103
4.8.1 Syntax . . . 103
4.8.2 Example . . . 105
4.8.2.1 Creating a Sample Table and Query . . . 105
4.8.2.2 Splitting the Query . . . 107
4.8.3 Best Practices . . . 108
4.8.3.1 General . . . 108
4.8.3.2 Choosing a Column to Split by . . . 109
4.8.3.3 Aggregation Best Practices . . . 109
4.8.3.4 Date as Number best practices . . . 109
4.9 Monitoring Query Performance . . . 109
4.9.1 Setting Up System Monitoring Preferences . . . 110
4.9.1.1 Adjusting the Logging Frequency . . . 110
4.9.1.2 Creating a Dedicated Foreign Table to Store Log Details . . . 110
4.9.2 Using the SHOW_NODE_INFO Command . . . 111
4.9.3 Understanding the Query Execution Plan Output . . . 112
4.9.3.1 Information Presented in the Execution Plan . . . 114
4.9.3.2 Commonly Seen Nodes . . . 114
4.9.4 Examples . . . 115
4.9.4.1 Spooling to Disk . . . 115
4.9.4.1.1 Identifying the Offending Nodes . . . 115
4.9.4.1.2 Common Solutions for Reducing Spool . . . 117
4.9.4.2 Queries with Large Result Sets . . . 117
4.9.4.2.1 Identifying the Offending Nodes . . . 117
4.9.4.2.2 Common Solutions for Reducing Gather Time . . . 119
4.9.4.3 Inefficient Filtering . . . 119
4.9.4.3.1 Identifying the Situation . . . 119
4.9.4.3.2 Common Solutions for Improving Filtering . . . 123
4.9.4.4 Joins with TEXT Keys . . . 123
4.9.4.4.1 Identifying the Situation . . . 123
4.9.4.4.2 Common Solutions for Improving Query Performance . . . 125
4.9.4.5 Sorting on Big TEXT Fields . . . 127
4.9.4.5.1 Identifying the Situation . . . 127
4.9.4.5.2 Common Solutions for Improving Sort Performance on TEXT Keys . . . 129
4.9.4.6 High Selectivity Data . . . 129
4.9.4.6.1 Identifying the Situation . . . 129
4.9.4.6.2 Common Solutions for Improving Performance with High Selectivity Hints . . . 130
4.9.4.7 Performance of Unsorted Data in Joins . . . 130
4.9.4.7.1 Identifying the Situation . . . 130
4.9.4.7.2 Improving Join Performance when Data is Sparse . . . 131
4.9.4.8 Manual Join Reordering . . . 131
4.9.4.8.1 Identifying the situation . . . 131
4.9.4.8.2 Changing the Join Order . . . 132
4.9.5 Further Reading . . . 133
4.10 Security . . . 133
4.10.1 Overview . . . 133
4.10.2 Security best practices for SQream DB . . . 133

4.10.2.1 Secure OS access . . . 133
4.10.2.2 Change the default SUPERUSER . . . 134
4.10.2.3 Create distinct user roles . . . 134
4.10.2.4 Limit SUPERUSER access . . . 134
4.10.2.5 Password strength guidelines . . . 134
4.10.2.6 Use TLS/SSL when possible . . . 134
4.11 Saved Queries . . . 135
4.11.1 Syntax . . . 135
4.11.1.1 Parameter Support . . . 135
4.11.2 Permissions . . . 136
4.11.3 Parameterized Query . . . 136
4.12 Optimization and Best Practices . . . 136
4.12.1 Table design . . . 136
4.12.1.1 Using DATE and DATETIME Data Types . . . 136
4.12.1.2 Avoiding Data flattening and Denormalization . . . 137
4.12.1.3 Converting Foreign Tables to Native Tables . . . 137
4.12.1.4 Leveraging Column Data Information . . . 137
4.12.1.4.1 Appropriately Using NULL and NOT NULL . . . 137
4.12.2 Sorting . . . 137
4.12.3 Query Best Practices . . . 138
4.12.3.1 Reducing Datasets Before Joining Tables . . . 138
4.12.3.2 Using ANSI JOIN . . . 138
4.12.3.3 Using High-Selectivity hint . . . 139
4.12.3.4 Avoiding Aggregation Overflow . . . 139
4.12.3.5 Prefer COUNT(*) and COUNT to Non-nullable Columns . . . 140
4.12.3.6 Returning Only Required Columns . . . 140
4.12.3.7 Reducing Recurring Compilation Time . . . 140
4.12.3.8 Reducing JOIN Complexity . . . 140
4.12.4 Data Loading Considerations . . . 141
4.12.4.1 Using Natural Data Sorting . . . 141
4.13 Oracle Migration Guide . . . 141
4.13.1 Using SQream Commands, Statements, and UDFs . . . 142
4.13.1.1 Operation Functions . . . 142
4.13.1.2 Conditional Functions . . . 142
4.13.1.3 Conversion Functions . . . 142
4.13.1.4 Numeric Functions . . . 143
4.13.1.5 Character Functions Returning Character Values . . . 146
4.13.1.6 Character Functions Returning Number Values . . . 147
4.13.1.7 Datetime Functions . . . 147
4.13.1.8 General Comparison Functions . . . 148
4.13.1.9 NULL-Related Functions . . . 148
4.13.1.10 Aggregate Functions . . . 148
4.13.1.11 Analytic Functions . . . 149

5 Configuration Guides **151**
5.1 Configuring SQream . . . 151
5.1.1 Cluster and Session . . . 151
5.1.1.1 Setting the flags . . . 151
5.1.1.1.1 Syntax . . . 151
5.1.1.1.2 Configuration file . . . 152
5.1.1.2 Flag List . . . 152
5.1.2 Workers . . . 153
5.1.3 Modification Methods . . . 155
5.1.3.1 Worker Configuration File . . . 155

5.1.3.2 Cluster and Session Configuration File . . . 155
5.1.3.3 Metadata Configuration File . . . 156
5.1.4 Parameter Values . . . 157
5.1.5 Showing All Flags in the Catalog Table . . . 158
5.2 LDAP . . . 158
5.2.1 Before You Begin . . . 158
5.2.2 Setting LDAP Authentication Management . . . 158
5.2.2.1 Basic Method . . . 159
5.2.2.1.1 Flags . . . 159
5.2.2.1.2 Basic Method Configuration . . . 159
5.2.2.1.3 Example . . . 160
5.2.2.2 Advanced Method . . . 160
5.2.2.2.1 Flags . . . 161
5.2.2.2.2 Preparing LDAP Users . . . 161
5.2.2.2.3 Preparing SQreamDB Roles . . . 162
5.2.2.2.4 Advanced Method Configuration . . . 162
5.2.2.2.5 Example . . . 163
5.2.3 Disabling LDAP Authentication . . . 164
5.3 Single Sign-On . . . 164
5.3.1 Before You Begin . . . 164
5.3.2 Setting SQreamDB Acceleration Studio . . . 164

6 Architecture **167**
6.1 Internals and Architecture . . . 167
6.1.1 Concurrency and Admission Control . . . 168
6.1.2 Statement Compiler . . . 168
6.1.3 Building Blocks (GPU Workers) . . . 168
6.1.4 Storage Layer . . . 168
6.1.4.1 Metadata Layer . . . 168
6.1.4.2 Bulk Data Layer Optimization . . . 168
6.1.5 Transactions . . . 169
6.2 Filesystem and Usage . . . 169
6.2.1 Directory organization . . . 169
6.2.1.1 `databases` . . . 170
6.2.1.2 `metadata` or `rocksdb` . . . 172
6.2.1.3 `temp` . . . 172
6.2.1.4 `logs` . . . 172
6.3 Sizing . . . 172
6.3.1 Concurrency and Scaling in SQreamDB . . . 172
6.3.1.1 Scaling When Data Sizes Grow . . . 173
6.3.1.2 Scaling When Queries Are Queuing . . . 173
6.3.1.3 What To Do When Queries Are Slow . . . 173
6.3.2 Spooling Configuration . . . 173
6.3.2.1 Example . . . 173
6.3.2.1.1 Setting Spool Memory . . . 173

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
7.1.4 View Activity Report . . . 176
7.2 Executing Statements and Running Queries from the Editor . . . 176
7.2.1 Executing Statements from the Toolbar . . . 177

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 . . . 185
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 . . . 190
8.1.3.1.1 Before You Begin . . . 190
8.1.3.1.2 Setting Up a Connection to SQreamDB . . . 190
8.1.3.1.3 Limitation . . . 191
8.1.3.2 Informatica Cloud Services . . . 192
8.1.3.2.1 Overview . . . 192
8.1.3.2.1.1 Establishing a Connection between SQream and Informatica . . . 192
8.1.3.2.1.2 Establishing a Connection In Your Environment . . . 193
8.1.3.2.1.3 Establishing an ODBC DSN Connection In Your Environment . . . 193
8.1.3.2.1.4 Establishing a JDBC Connection In Your Environment . . . 193
8.1.3.2.1.5 Supported SQream Driver Versions . . . 194
8.1.3.3 MicroStrategy . . . 194
8.1.3.3.1 Overview . . . 194
8.1.3.3.1.1 What is MicroStrategy? . . . 194
8.1.3.3.1.2 Connecting a Data Source . . . 195
8.1.3.3.1.3 Supported SQream Drivers . . . 196
8.1.3.4 Pentaho Data Integration . . . 196
8.1.3.4.1 Overview . . . 196
8.1.3.4.1.1 Installing Pentaho . . . 197
8.1.3.4.1.2 Installing and Setting Up the JDBC Driver . . . 197
8.1.3.4.1.3 Creating a Transformation . . . 197

8.1.3.4.1.4 Defining Your Output . . . 198
8.1.3.4.1.5 Importing Data . . . 199
8.1.3.5 PHP . . . 199
8.1.3.5.1 Overview . . . 199
8.1.3.5.1.1 Installing PHP . . . 200
8.1.3.5.1.2 Configuring PHP . . . 200
8.1.3.5.1.3 Operating PHP . . . 200
8.1.3.6 BI Desktop . . . 201
8.1.3.6.1 Prerequisites . . . 201
8.1.3.6.2 Installing Power BI Desktop . . . 202
8.1.3.6.3 Best Practices for Power BI . . . 203
8.1.3.7 R . . . 203
8.1.3.7.1 JDBC . . . 203
8.1.3.7.1.1 A full example . . . 204
8.1.3.7.2 ODBC . . . 204
8.1.3.7.2.1 A full example . . . 205
8.1.3.8 SAP BusinessObjects . . . 205
8.1.3.8.1 Overview . . . 205
8.1.3.8.2 Establishing a New Connection Using a Generic JDCB Connector . . . 206
8.1.3.9 SAS Viya . . . 207
8.1.3.9.1 Installing SAS Viya . . . 207
8.1.3.9.1.1 Downloading SAS Viya . . . 207
8.1.3.9.1.2 Installing the JDBC Driver . . . 207
8.1.3.9.2 Configuring SAS Viya . . . 207
8.1.3.9.3 Operating SAS Viya . . . 208
8.1.3.9.3.1 Using SAS Viya Visual Analytics . . . 208
8.1.3.9.4 Troubleshooting SAS Viya . . . 209
8.1.3.9.4.1 Inserting Only Required Data . . . 209
8.1.3.9.4.2 Creating a Separate Service for SAS Viya . . . 209
8.1.3.9.4.3 Locating the SQreamDB JDBC Driver . . . 209
8.1.3.9.4.4 Supporting TEXT . . . 209
8.1.3.10 Semarchy . . . 210
8.1.3.10.1 Before You Begin . . . 210
8.1.3.10.2 Setting Up a Connection to SQreamDB . . . 210
8.1.3.10.3 JDBC Connection String . . . 210
8.1.3.10.3.1 Connection Parameters . . . 211
8.1.3.11 SQL Workbench . . . 211
8.1.3.11.1 Installing SQL Workbench with the SQream Installer . . . 212
8.1.3.11.2 Installing SQL Workbench Manually . . . 213
8.1.3.11.2.1 Install Java Runtime . . . 214
8.1.3.11.2.2 Get the SQream DB JDBC Driver . . . 214
8.1.3.11.2.3 Install SQL Workbench . . . 214
8.1.3.11.2.4 Setting up the SQream DB JDBC Driver Profile . . . 215
8.1.3.11.3 Create a New Connection Profile for Your Cluster . . . 217
8.1.3.11.4 Suggested Optional Configuration . . . 218
8.1.3.12 Tableau . . . 218
8.1.3.12.1 Prerequisites . . . 218
8.1.3.12.2 Setting Up JDBC . . . 218
8.1.3.12.3 Installing the Tableau Connector . . . 219
8.1.3.12.4 Connecting to SQream . . . 219
8.1.3.13 Talend . . . 220
8.1.3.13.1 Overview . . . 220
8.1.3.13.1.1 Creating a New Metadata JDBC DB Connection . . . 221
8.1.3.13.1.2 Supported SQream Drivers . . . 222

8.1.3.13.1.3 Supported Data Sources . . . 222
8.1.3.13.1.4 Known Issues . . . 222
8.1.3.14 TIBCO Spotfire . . . 222
8.1.3.14.1 Overview . . . 222
8.1.3.14.1.1 Establishing a Connection between TIBCO Spotfire and SQream . . . 222
8.1.3.14.1.2 Creating a JDBC Connection . . . 223
8.1.3.14.1.3 Creating an ODBC Connection . . . 223
8.1.3.14.1.4 Creating the SQream Data Source Template . . . 224
8.1.3.14.1.5 Creating a Data Source . . . 225
8.1.3.14.1.6 Creating an Information Link . . . 226
8.1.3.14.1.7 Troubleshooting . . . 228
8.1.3.14.1.8 The JDBC Driver does not Support Boolean, Decimal, or Numeric Types . . . 228
8.1.3.14.1.9 Information Services do not Support Live Queries . . . 228
8.2 Client Drivers . . . 228
8.2.1 Client Driver Downloads . . . 229
8.2.1.1 SQreamNET . . . 229
8.2.1.1.1 Before You Begin . . . 230
8.2.1.1.2 Integrating SQreamNET . . . 230
8.2.1.1.3 Connecting to SQream For the First Time . . . 230
8.2.1.1.3.1 Connection String Syntax . . . 230
8.2.1.1.3.2 Connection Parameters . . . 230
8.2.1.1.3.3 Connection String Examples . . . 231
8.2.1.1.3.4 Sample C# Program . . . 231
8.2.1.1.4 Limitations . . . 233
8.2.1.2 Dataiku . . . 233
8.2.1.2.1 Before You Begin . . . 233
8.2.1.2.2 Establishing a Dataiku Connection . . . 234
8.2.1.3 JDBC . . . 234
8.2.1.3.1 Installing the JDBC Driver . . . 234
8.2.1.3.1.1 Prerequisites . . . 235
8.2.1.3.1.2 Getting the JAR file . . . 235
8.2.1.3.1.3 Setting Up the Class Path . . . 235
8.2.1.3.2 Connecting to SQream Using a JDBC Application . . . 235
8.2.1.3.2.1 Driver Class . . . 235
8.2.1.3.2.2 Connection String . . . 236
8.2.1.3.2.3 Connection Parameters . . . 236
8.2.1.3.2.4 Connection String Examples . . . 237
8.2.1.3.2.5 Java Program Sample . . . 237
8.2.1.3.3 Prepared Statements . . . 238
8.2.1.3.3.1 Prepared Statement Sample . . . 239
8.2.1.3.3.2 Prepared Statement Limitations . . . 239
8.2.1.4 Node.JS . . . 239
8.2.1.4.1 Installing the Node.JS driver . . . 240
8.2.1.4.1.1 Prerequisites . . . 240
8.2.1.4.1.2 Install with NPM . . . 240
8.2.1.4.1.3 Install from an offline package . . . 240
8.2.1.4.2 Connect to SQream DB with a Node.JS application . . . 240
8.2.1.4.2.1 Create a simple test . . . 240
8.2.1.4.2.2 Run the test . . . 241
8.2.1.4.3 API reference . . . 241
8.2.1.4.3.1 Connection parameters . . . 241
8.2.1.4.3.2 Events . . . 241
8.2.1.4.3.3 Example . . . 241

8.2.1.4.3.4 Input placeholders . . . 242
8.2.1.4.4 Examples . . . 242
8.2.1.4.4.1 Setting configuration flags . . . 242
8.2.1.4.4.2 Lazyloading . . . 243
8.2.1.4.4.3 Reusing a connection . . . 243
8.2.1.4.4.4 Using placeholders in queries . . . 244
8.2.1.4.5 Troubleshooting and recommended configuration . . . 245
8.2.1.4.5.1 Preventing `heap out of memory` errors . . . 245
8.2.1.4.5.2 BIGINT support . . . 245
8.2.1.5 ODBC . . . 245
8.2.1.5.1 Install and Configure ODBC on Windows . . . 245
8.2.1.5.1.1 Installing the ODBC Driver . . . 246
8.2.1.5.1.2 Prerequisites . . . 246
8.2.1.5.1.3 Visual Studio 2015 Redistributables . . . 246
8.2.1.5.1.4 Administrator Privileges . . . 246
8.2.1.5.1.5 Running the Windows Installer . . . 246
8.2.1.5.1.6 Selecting Components . . . 247
8.2.1.5.1.7 Configuring the ODBC Driver DSN . . . 247
8.2.1.5.1.8 Connection Parameters . . . 250
8.2.1.5.1.9 Troubleshooting . . . 250
8.2.1.5.1.10 Solving “Code 126” ODBC errors . . . 250
8.2.1.5.1.11 Limitations . . . 250
8.2.1.5.2 Install and configure ODBC on Linux . . . 250
8.2.1.5.2.1 Prerequisites . . . 251
8.2.1.5.2.2 unixODBC . . . 251
8.2.1.5.2.3 Install unixODBC on RHEL . . . 251
8.2.1.5.2.4 Install the ODBC driver with a script . . . 251
8.2.1.5.2.5 Install the ODBC driver manually . . . 252
8.2.1.5.2.6 Install the driver dependencies . . . 253
8.2.1.5.2.7 Testing the connection . . . 253
8.2.1.5.2.8 ODBC DSN Parameters . . . 255
8.2.1.5.2.9 Limitations . . . 256
8.2.1.5.3 Getting the ODBC driver . . . 256
8.2.1.5.4 Install and configure the ODBC driver . . . 256
8.2.1.6 Python (pysqream) . . . 256
8.2.1.6.1 Installing the Python Connector . . . 257
8.2.1.6.1.1 Prerequisites . . . 257
8.2.1.6.1.2 Python . . . 257
8.2.1.6.1.3 PIP . . . 257
8.2.1.6.1.4 OpenSSL for Linux . . . 258
8.2.1.6.1.5 Installing via PIP with an internet connection . . . 258
8.2.1.6.1.6 Installing via PIP without an internet connection . . . 258
8.2.1.6.1.7 Upgrading an Existing Installation . . . 258
8.2.1.6.2 SQLAlchemy . . . 258
8.2.1.6.2.1 Before You Begin . . . 259
8.2.1.6.2.2 Limitation . . . 259
8.2.1.6.2.3 Creating a Standard Connection . . . 259
8.2.1.6.2.4 Pulling a Table into Pandas . . . 259
8.2.1.6.3 API . . . 260
8.2.1.6.3.1 Using the Cursor . . . 260
8.2.1.6.3.2 Reading Result Metadata . . . 262
8.2.1.6.3.3 Loading Data into a Table . . . 262
8.2.1.6.3.4 Using SQLAlchemy ORM to Create and Populate Tables . . . 264
8.2.1.6.4 Prepared Statements . . . 265

8.2.1.6.4.1 Prepared Statement Limitations . . . 265
8.2.1.6.4.2 Prepared Statements code example . . . 265
8.2.1.7 Spark . . . 266
8.2.1.7.1 Before You Begin . . . 267
8.2.1.7.2 Configuration . . . 267
8.2.1.7.2.1 Connecting Spark to SQreamDB . . . 269
8.2.1.7.3 Transferring Data . . . 269
8.2.1.7.3.1 Transferring Data From SQreamDB to Spark . . . 269
8.2.1.7.3.2 Transferring Data From Spark to SQreamDB . . . 269
8.2.1.7.4 Data Types and Mapping . . . 270
8.2.1.7.5 Example . . . 270
8.2.1.8 Trino . . . 271
8.2.1.8.1 Before You Begin . . . 272
8.2.1.8.2 Installation . . . 272
8.2.1.8.3 Connecting to SQreamDB . . . 272
8.2.1.8.4 Supported Data Types and Mapping . . . 272
8.2.1.8.5 Examples . . . 273
8.2.1.8.6 Limitations . . . 273

9 Data Ingestion Sources 275
9.1 Overview . . . 275
9.1.1 Getting Started . . . 275
9.1.2 Data Loading Considerations . . . 276
9.1.2.1 Verifying Data and Performance after Loading . . . 276
9.1.2.2 File Soure Location when Loading . . . 277
9.1.2.3 Supported Load Methods . . . 277
9.1.2.4 Unsupported Data Types . . . 277
9.1.2.5 Handing Extended Errors . . . 277
9.1.3 Foreign Data Wrapper Best Practice . . . 278
9.1.3.1 Best Practices for CSV . . . 278
9.1.3.2 Best Practices for Parquet . . . 278
9.1.3.2.1 Supported Types and Behavior Notes . . . 278
9.1.3.3 Best Practices for ORC . . . 279
9.1.3.3.1 Type Support and Behavior Notes . . . 279
9.1.4 Further Reading and Migration Guides . . . 280
9.2 Avro . . . 281
9.2.1 Foreign Data Wrapper Prerequisites . . . 281
9.2.2 Making Avro Files Accessible to Workers . . . 281
9.2.3 Preparing Your Table . . . 282
9.2.3.1 Creating a Table . . . 282
9.2.3.2 Creating a Foreign Table . . . 283
9.2.4 Mapping Between SQream and Avro Data Types . . . 284
9.2.4.1 Primitive Data Types . . . 284
9.2.4.2 Complex Data Types . . . 285
9.2.4.3 Logical Data Types . . . 285
9.2.5 Mapping Objects to Rows . . . 285
9.2.6 Ingesting Data into SQream . . . 286
9.2.6.1 Syntax . . . 286
9.2.6.2 Example . . . 286
9.2.7 Parameters . . . 286
9.2.8 Best Practices . . . 287
9.2.9 Additional Examples . . . 287
9.2.9.1 Omitting Unsupported Column Types . . . 288
9.2.9.2 Modifying Data Before Loading . . . 288

9.2.9.3 Loading a Table from a Directory of Avro Files on HDFS . . . 289
9.2.9.4 Loading a Table from a Directory of Avro Files on S3 . . . 289
9.3 CSV . . . 290
9.3.1 Foreign Data Wrapper Prerequisites . . . 290
9.3.2 Prepare CSVs . . . 290
9.3.3 Place CSVs where SQream DB workers can access . . . 291
9.3.4 Figure out the table structure . . . 291
9.3.5 Bulk load the data with COPY FROM . . . 292
9.3.5.1 Loading a standard CSV File From a Local Filesystem . . . 292
9.3.5.2 Loading a PSV (pipe separated value) file . . . 293
9.3.5.3 Loading a TSV (tab separated value) file . . . 293
9.3.5.4 Loading a text file with non-printable delimiter . . . 293
9.3.5.5 Loading a Text File With Multi-Character Delimiters . . . 293
9.3.5.6 Loading Files With a Header Row . . . 294
9.3.5.7 Loading Files Formatted for Windows (\r\n) . . . 294
9.3.5.8 Loading a File From a Public S3 Bucket . . . 294
9.3.5.9 Loading files from an authenticated S3 bucket . . . 295
9.3.5.10 Loading files from an HDFS storage . . . 295
9.3.5.11 Saving rejected rows to a file . . . 295
9.3.5.12 Stopping the load if a certain amount of rows were rejected . . . 296
9.3.5.13 Load CSV files from a set of directories . . . 296
9.3.5.14 Rearrange destination columns . . . 296
9.3.5.15 Loading non-standard dates . . . 297
9.4 Parquet . . . 297
9.4.1 Foreign Data Wrapper Prerequisites . . . 297
9.4.2 Preparing Your Parquet Files . . . 298
9.4.3 Making Parquet Files Accessible to Workers . . . 298
9.4.4 Creating a Table . . . 299
9.4.5 Ingesting Data into SQream . . . 300
9.4.5.1 Syntax . . . 300
9.4.5.2 Examples . . . 300
9.4.5.2.1 Omitting Unsupported Column Types . . . 300
9.4.5.2.2 Modifying Data Before Loading . . . 301
9.4.5.2.3 Loading a Table from a Directory of Parquet Files on HDFS . . . 301
9.4.5.2.4 Loading a Table from a Directory of Parquet Files on S3 . . . 302
9.4.6 Best Practices . . . 302
9.5 ORC . . . 303
9.5.1 Foreign Data Wrapper Prerequisites . . . 303
9.5.2 Prepare the files . . . 304
9.5.3 Place ORC files where SQream DB workers can access them . . . 305
9.5.4 Figure out the table structure . . . 305
9.5.5 Verify table contents . . . 306
9.5.6 Copying data into SQream DB . . . 307
9.5.6.1 Working Around Unsupported Column Types . . . 307
9.5.6.2 Modifying data during the copy process . . . 307
9.5.7 Further ORC loading examples . . . 308
9.5.7.1 Loading a table from a directory of ORC files on HDFS . . . 308
9.5.7.2 Loading a table from a bucket of files on S3 . . . 308
9.6 JSON . . . 309
9.6.1 Foreign Data Wrapper Prerequisites . . . 309
9.6.2 Making JSON Files Accessible to Workers . . . 309
9.6.3 Mapping between JSON and SQream . . . 310
9.6.3.1 Character Escaping . . . 310
9.6.4 Ingesting JSON Data into SQream . . . 310

9.6.4.1 Syntax . . . 310
9.6.4.2 Parameters . . . 311
9.6.4.3 Automatic Schema Inference . . . 313
9.6.4.4 Examples . . . 313
9.7 SQLoader As a Service . . . 314
9.7.1 Before You Begin . . . 314
9.7.1.1 Minimum Hardware Requirements . . . 315
9.7.1.2 Sizing Guidelines . . . 315
9.7.2 Installation and Connectivity . . . 315
9.7.2.1 Getting All Configuration and JAR Files . . . 315
9.7.2.2 Installation . . . 316
9.7.2.2.1 Deployment Parameters . . . 316
9.7.2.2.2 Installing the Admin Server and SQLoader Service . . . 318
9.7.2.3 Reconfiguration . . . 320
9.7.2.4 Connection String . . . 320
9.7.3 SQLoader Service Interface . . . 322
9.7.3.1 Supported HTTP Requests . . . 324
9.7.3.2 High Availability . . . 325
9.7.3.3 Log Rotation . . . 325
9.7.3.3.1 Log Automatic cleanup . . . 325
9.7.3.4 SQLoader Request Parameters . . . 325
9.7.3.4.1 Using the `loadTypeName` Parameter . . . 326
9.7.3.5 Using the SQLoader Service Web Interface . . . 327
9.7.3.5.1 SQLoader Service Web Interface Features . . . 327
9.7.4 Creating Summary and Catalog Tables . . . 327
9.7.4.1 Creating a Summary Table . . . 327
9.7.4.2 Creating Catalog Tables . . . 329
9.7.5 Data Type Mapping . . . 331
9.7.5.1 Automatic Mapping . . . 331
9.7.5.1.1 Greenplum . . . 331
9.7.5.1.2 Microsoft SQL Server . . . 331
9.7.5.1.3 Oracle . . . 332
9.7.5.1.4 Postgresql . . . 332
9.7.5.1.5 SAP HANA . . . 333
9.7.5.1.6 Sybase . . . 333
9.7.5.1.7 Teradata . . . 334
9.7.5.2 Manually Adjusting Mapping . . . 334
9.7.5.2.1 `names` Method . . . 334
9.7.5.2.1.1 Preparing Oracle for Data Migration . . . 335
9.7.5.2.1.2 Preparing CDC Tables . . . 335
9.7.5.2.1.3 Preparing Incremental Table . . . 337

10 External Storage Platforms **339**
10.1 Azure Blob Storage . . . 339
10.1.1 ABS Bucket File Location . . . 339
10.1.2 Connection String . . . 339
10.1.3 Examples . . . 340
10.2 Google Cloud Platform . . . 340
10.2.1 GCP Bucket File Location . . . 340
10.2.2 GCP Access . . . 340
10.2.2.1 Before You Begin . . . 340
10.2.2.2 Granting GCP Access . . . 340
10.2.3 Examples . . . 341
10.3 HDFS Environment . . . 341

10.3.1 Configuring an HDFS Environment for the User **sqream** . . . 341
10.3.2 Authenticating Hadoop Servers that Require Kerberos . . . 343
10.4 Amazon Web Services . . . 345
10.4.1 S3 Bucket File Location . . . 345
10.4.2 S3 Access . . . 345
10.4.3 Authentication . . . 345
10.4.4 Connecting to S3 Using SQreamDB Legacy Configuration File . . . 346
10.4.5 Examples . . . 346
10.4.5.1 Creating a Foreign Table . . . 346
10.4.5.2 Querying Foreign Tables . . . 347
10.4.5.3 Bulk Loading a File from a Public S3 Bucket . . . 347
10.4.5.4 Loading Files from an Authenticated S3 Bucket . . . 347

11 Feature Guides **349**
11.1 Query Healer . . . 349
11.1.1 Configuration . . . 349
11.1.2 Query Log . . . 349
11.1.3 Activating a Graceful Shutdown . . . 350
11.2 Compression . . . 350
11.2.1 Encoding . . . 350
11.2.2 Lossless Compression . . . 350
11.2.2.1 Automatic Compression . . . 351
11.2.2.2 Compression Methods . . . 351
11.2.2.3 Specifying Compression Strategies . . . 352
11.2.2.3.1 Explicitly Specifying Automatic Compression . . . 352
11.2.2.3.2 Forcing No Compression . . . 352
11.2.2.3.3 Forcing Compression . . . 352
11.2.2.4 Examining Compression Effectiveness . . . 353
11.2.2.4.1 Querying the Catalog . . . 353
11.2.2.4.2 Example Subset from “Ontime” Table . . . 354
11.2.2.4.3 Notes on Reading the “Ontime” Table . . . 356
11.2.3 Best Practices . . . 357
11.2.3.1 Letting SQream Determine the Best Compression Strategy . . . 357
11.2.3.2 Maximizing the Advantage of Each Compression Scheme . . . 357
11.2.3.3 Choosing Data Types that Fit Your Data . . . 357
11.3 Python User-Defined Functions . . . 357
11.3.1 Before You Begin . . . 358
11.3.2 SQreamDB’s UDF Support . . . 358
11.3.2.1 Scalar Functions . . . 358
11.3.2.2 Python . . . 358
11.3.2.3 Using Modules . . . 358
11.3.3 Working with Existing UDFs . . . 359
11.3.3.1 Finding Existing UDFs in the Catalog . . . 359
11.3.3.2 Getting Function DDL . . . 359
11.3.3.3 Handling Errors . . . 359
11.3.4 Permissions and Sharing . . . 359
11.3.5 Example . . . 360
11.3.6 Best Practices . . . 361
11.4 Workload Manager . . . 361
11.4.1 Setting Up Service Queues . . . 361
11.4.2 Example - Allocating ETL Resources . . . 361
11.4.2.1 Creating the Configuration . . . 362
11.4.2.2 Verifying the Configuration . . . 362
11.4.3 Configuring a Client Connection to a Specific Service . . . 363

11.4.3.1 Using SQream Studio . . . 363
11.4.3.2 Using the SQream SQL CLI Reference . . . 363
11.4.3.3 Using a JDBC Client Driver . . . 364
11.4.3.4 Using an ODBC Client Driver . . . 364
11.4.3.5 Using a Python Client Driver . . . 364
11.4.3.6 Using a Node.js Client Driver . . . 365
11.5 Concurrency and Locks . . . 365
11.5.1 Locking Modes . . . 365
11.5.2 When are Locks Obtained? . . . 366
11.5.3 Monitoring Locks . . . 366
11.6 Data Encryption . . . 367
11.6.1 Overview . . . 367
11.6.2 Encryption Methods . . . 367
11.6.2.1 Encrypting Data in Transit . . . 367
11.6.2.2 Encrypting Data at Rest . . . 367
11.6.3 Data Types . . . 368
11.6.4 Syntax . . . 368
11.6.5 Examples . . . 368
11.6.6 Limitations . . . 369
11.6.7 Permissions . . . 370

12 References **371**
12.1 SQL Statements and Syntax . . . 371
12.1.1 SQL Syntax Features . . . 371
12.1.2 SQL Statements . . . 372
12.1.2.1 Data Definition Commands (DDL) . . . 372
12.1.2.2 Data Manipulation Commands (DML) . . . 373
12.1.2.3 Utility Commands . . . 373
12.1.2.4 Workload Management . . . 374
12.1.2.5 Access Control Commands . . . 374
12.1.3 SQL Functions . . . 375
12.1.3.1 Summary of Functions . . . 375
12.1.3.1.1 Built-In Scalar Functions . . . 376
12.1.3.1.1.1 Bitwise Operations . . . 376
12.1.3.1.1.2 Conditionals . . . 376
12.1.3.1.1.3 Conversion . . . 376
12.1.3.1.1.4 Date and Time . . . 377
12.1.3.1.1.5 Numeric . . . 377
12.1.3.1.1.6 Strings . . . 378
12.1.3.1.2 User-Defined Scalar Functions . . . 379
12.1.3.1.3 Aggregate Functions . . . 379
12.1.3.1.4 Window Functions . . . 380
12.1.3.1.5 Workload Management Functions . . . 380
12.1.3.1.5.1 Built-In Scalar Functions . . . 381
12.1.3.1.5.2 User-Defined Functions . . . 381
12.1.3.1.5.3 Aggregate Functions . . . 381
12.1.3.1.5.4 Overview . . . 381
12.1.3.1.5.5 Available Aggregate Functions . . . 381
12.1.3.1.5.6 Window Functions . . . 382
12.2 Catalog Reference . . . 382
12.2.1 What Information Does the Schema Contain? . . . 382
12.2.2 How to Get Table Information? . . . 382
12.2.2.1 Database Management Tables . . . 383
12.2.2.2 Data Storage and Organization Tables . . . 383

12.2.2.2.1 Catalog Tables . . . 383
12.2.2.2.1.1 Clustering Keys . . . 384
12.2.2.2.1.2 Columns . . . 384
12.2.2.2.1.3 Columns . . . 384
12.2.2.2.1.4 External Table Columns . . . 385
12.2.2.2.1.5 Databases . . . 385
12.2.2.2.1.6 Parameters . . . 386
12.2.2.2.1.7 Permissions . . . 386
12.2.2.2.1.8 Permission Types . . . 386
12.2.2.2.1.9 Default Permissions . . . 387
12.2.2.2.1.10 Default Table Permissions . . . 387
12.2.2.2.1.11 Default Schema Permissions . . . 387
12.2.2.2.1.12 Table Permissions . . . 388
12.2.2.2.1.13 Database Permissions . . . 388
12.2.2.2.1.14 Schema Permissions . . . 388
12.2.2.2.1.15 Queries . . . 389
12.2.2.2.1.16 Roles . . . 389
12.2.2.2.1.17 Roles . . . 389
12.2.2.2.1.18 Role Memberships . . . 389
12.2.2.2.1.19 Schemas . . . 390
12.2.2.2.1.20 Tables . . . 390
12.2.2.2.1.21 Tables . . . 390
12.2.2.2.1.22 Foreign Tables . . . 390
12.2.2.2.1.23 Views . . . 391
12.2.2.2.1.24 User Defined Functions . . . 391
12.2.2.2.2 Additional Tables . . . 391
12.2.2.2.2.1 Extents . . . 392
12.2.2.2.2.2 Chunk Columns . . . 392
12.2.2.2.2.3 Chunks . . . 393
12.2.2.2.2.4 Delete Predicates . . . 393
12.2.2.2.3 Examples . . . 394
12.2.2.2.3.1 Listing All Tables in a Database . . . 394
12.2.2.2.3.2 Listing All Schemas in a Database . . . 394
12.2.2.2.3.3 Listing Columns and Their Types for a Specific Table . . . 395
12.2.2.2.3.4 Listing Delete Predicates . . . 395
12.2.2.2.3.5 Listing Saved Queries . . . 395
12.3 Command line programs . . . 395
12.3.1 metadata_server . . . 396
12.3.1.1 Command Line Arguments . . . 396
12.3.1.2 Starting metadata server . . . 396
12.3.1.2.1 Starting temporarily . . . 396
12.3.1.2.2 Starting temporarily with non-default port . . . 397
12.3.1.2.3 Stopping metadata server . . . 397
12.3.2 sqreamd . . . 397
12.3.2.1 Starting SQream DB . . . 397
12.3.2.1.1 Start SQream DB temporarily . . . 397
12.3.2.2 Command line arguments . . . 398
12.3.2.2.1 Positional command arguments . . . 398
12.3.3 Multi-Platform Sqream SQL . . . 398
12.3.3.1 Before You Begin . . . 399
12.3.3.2 Installing Sqream SQL . . . 399
12.3.3.3 Using Sqream SQL . . . 399
12.3.3.3.1 Running Commands Interactively (SQL shell) . . . 399
12.3.3.3.2 Executing Batch Scripts (`-f`) . . . 400

12.3.3.3.3 Executing Commands Immediately (-c) . . . 401
12.3.3.4 Examples . . . 401
12.3.3.4.1 Starting a Regular Interactive Shell . . . 401
12.3.3.4.2 Executing Statements in an Interactive Shell . . . 402
12.3.3.4.3 Executing SQL Statements from the Command Line . . . 402
12.3.3.4.4 Controlling the Client Output . . . 403
12.3.3.4.4.1 Exporting SQL Query Results to CSV . . . 403
12.3.3.4.4.2 Changing a CSV to a TSV . . . 403
12.3.3.4.5 Executing a Series of Statements From a File . . . 403
12.3.3.4.6 Connecting Using Environment Variables . . . 404
12.3.3.4.7 Connecting to a Specific Queue . . . 404
12.3.3.5 Operations and Flag References . . . 404
12.3.3.5.1 Command Line Arguments . . . 404
12.3.3.5.1.1 Supported Record Delimiters . . . 405
12.3.3.5.2 Meta-Commands . . . 405
12.3.3.5.3 Basic Commands . . . 405
12.3.3.5.4 Moving Around the Command Line . . . 406
12.3.3.5.5 Searching . . . 406
12.3.4 Server Picker . . . 406
12.3.4.1 Command Line Arguments . . . 407
12.3.4.1.1 Parameters . . . 407
12.3.4.1.2 Example . . . 407
12.3.4.2 Starting server picker . . . 407
12.3.4.2.1 Starting temporarily . . . 407
12.3.4.2.2 Starting temporarily with non-default port . . . 407
12.3.4.2.3 Stopping server picker . . . 408
12.3.5 SqreamStorage . . . 408
12.3.5.1 Running SqreamStorage . . . 408
12.3.5.2 Command Line Arguments . . . 408
12.3.5.3 Example . . . 408
12.3.6 upgrade_storage . . . 408
12.3.6.1 Running upgrade_storage . . . 409
12.3.6.1.1 Command line arguments and options . . . 409
12.3.6.1.2 Syntax . . . 409
12.3.6.2 Results and error codes . . . 409
12.3.6.3 Examples . . . 409
12.3.6.3.1 Upgrade SQream DB's storage cluster . . . 409
12.4 SQL Feature Checklist . . . 410
12.4.1 Data Types and Values . . . 411
12.4.2 Constraints . . . 411
12.4.3 Transactions . . . 411
12.4.4 Indexes . . . 411
12.4.5 Schema Changes . . . 412
12.4.6 Statements . . . 412
12.4.7 Clauses . . . 412
12.4.8 Table Expressions . . . 413
12.4.9 Scalar Expressions . . . 413
12.4.10 Permissions . . . 413
12.4.11 Extra Functionality . . . 414

13 Data Types **415**
13.1 Supported Data Types . . . 415
13.1.1 Primitive Data Types . . . 415
13.1.2 Array . . . 417

13.1.2.1 Syntax . . . 417
13.1.2.2 Supported Operators . . . 419
13.1.2.3 Examples . . . 420
13.1.2.3.1 `ARRAY` Statements . . . 420
13.1.2.3.2 Ingesting Arrayed Data from External Files . . . 421
13.1.2.4 Limitations . . . 422
13.1.2.4.1 Casting Limitations . . . 422
13.1.2.4.1.1 `NUMERIC` . . . 422
13.1.2.4.1.2 `TEXT` . . . 422
13.1.2.4.2 Connectors . . . 423
13.1.2.4.2.1 `.NET` and `ODBC` . . . 423
13.1.2.4.2.2 `Pysqream` . . . 423
13.1.2.4.3 Functions . . . 423
13.1.2.4.3.1 || (Concatenate) . . . 423
13.1.2.4.3.2 UNNEST . . . 423
13.1.2.4.3.3 Window . . . 423
13.2 Casts and Conversions . . . 423
13.2.1 Conversion Methods . . . 424
13.2.2 Supported Casts . . . 424
13.2.2.1 Value Dependent Conversions . . . 424
13.3 Supported Casts . . . 425
13.3.1 Numeric . . . 425
13.3.1.1 Numeric Examples . . . 425
13.3.2 Boolean . . . 426
13.3.2.1 Boolean Examples . . . 426
13.3.2.2 Boolean Casts and Conversions . . . 426
13.3.3 Integer . . . 426
13.3.3.1 Integer Types . . . 427
13.3.3.2 Integer Examples . . . 427
13.3.3.3 Integer Casts and Conversions . . . 427
13.3.4 Floating Point . . . 428
13.3.4.1 Floating Point Types . . . 428
13.3.4.2 Floating Point Examples . . . 428
13.3.4.3 Floating Point Casts and Conversions . . . 429
13.3.5 String . . . 429
13.3.5.1 Length . . . 429
13.3.5.2 Syntax . . . 429
13.3.5.3 Size . . . 429
13.3.5.4 String Examples . . . 429
13.3.5.5 String Casts and Conversions . . . 430
13.3.6 Date . . . 430
13.3.6.1 Date Types . . . 430
13.3.6.2 Aliases . . . 431
13.3.6.3 Syntax . . . 431
13.3.6.4 Size . . . 431
13.3.6.5 Date Examples . . . 431
13.3.6.6 Date Casts and Conversions . . . 432
13.3.7 DateTime2 . . . 432
13.3.7.1 Aliases . . . 432
13.3.7.2 Syntax . . . 432
13.3.7.3 Size . . . 432
13.3.7.4 Date Examples . . . 432
13.3.7.5 Date Casts and Conversions . . . 433

14 Release Notes **435**
14.1 SQDB release policy . . . 436
14.1.1 Release Cadence . . . 436
14.1.2 Transition to Maintenance Mode . . . 436
14.1.3 End of Support Timeline . . . 437
14.1.4 SQDB Releases Timeline . . . 437
14.2 4.x Release Notes . . . 437
14.2.1 Release Notes 4.3 . . . 437
14.2.1.1 Compatibility Matrix . . . 438
14.2.1.2 New Features and Enhancements . . . 438
14.2.1.3 SQreamDB Studio Updates and Improvements . . . 438
14.2.1.4 Known Issues . . . 438
14.2.1.5 Version 4.3 resolved Issues . . . 439
14.2.1.6 Configuration Adjustments . . . 439
14.2.1.7 Deprecations . . . 439
14.2.1.8 Upgrading to v4.3 . . . 440
14.2.2 Release Notes 4.5 . . . 441
14.2.2.1 Compatibility Matrix . . . 441
14.2.2.2 New Features and Enhancements . . . 442
14.2.2.3 Known Issues . . . 442
14.2.2.4 Version 4.5 resolved Issues . . . 442
14.2.2.5 Deprecations . . . 442
14.2.2.6 Upgrading to Version 4.5 . . . 442
14.2.3 Release Notes 4.8 . . . 443
14.2.3.1 Compatibility Matrix . . . 443
14.2.3.2 New Features and Enhancements . . . 444
14.2.3.3 Known Issues . . . 444
14.2.3.4 Version 4.8 resolved Issues . . . 444
14.2.3.5 Deprecations . . . 444
14.2.3.6 Upgrading to Version 4.8 . . . 444
14.2.4 Release Notes 4.9 . . . 445
14.2.4.1 Compatibility Matrix . . . 445
14.2.4.2 New Features and Enhancements . . . 446
14.2.4.3 Known Issues . . . 446
14.2.4.4 Version 4.9 resolved Issues . . . 446
14.2.4.5 Deprecations . . . 446
14.2.4.6 Upgrading to Version 4.9 . . . 446
14.2.5 Release Notes 4.10 . . . 447
14.2.5.1 Compatibility Matrix . . . 447
14.2.5.2 New Features and Enhancements . . . 448
14.2.5.3 Known Issues . . . 448
14.2.5.4 Version 4.10 resolved Issues . . . 448
14.2.5.5 Deprecations . . . 448
14.2.5.6 Upgrading to Version 4.10 . . . 449
14.2.6 Release Notes 4.11 . . . 449
14.2.6.1 Compatibility Matrix . . . 449
14.2.6.2 New Features and Enhancements . . . 450
14.2.6.3 Known Issues . . . 450
14.2.6.4 Version 4.11 resolved Issues . . . 450
14.2.6.5 Deprecation . . . 450
14.2.6.6 Upgrading to Version 4.11 . . . 450

15 Troubleshooting **453**
15.1 Remedying Slow Queries . . . 453

15.2 Resolving Common Issues . . . 454
15.2.1 Troubleshooting Cluster Setup and Configuration . . . 454
15.2.2 Troubleshooting Connectivity Issues . . . 455
15.2.3 Troubleshooting Query Performance . . . 455
15.2.4 Troubleshooting Query Behavior . . . 455
15.2.5 File an issue with SQream support . . . 455
15.3 Identifying Configuration Issues . . . 455
15.4 Lock Related Issues . . . 456
15.5 Log Related Issues . . . 456
15.5.1 Loading Logs with Foreign Tables . . . 456
15.5.2 Counting Message Types . . . 457
15.5.3 Finding Fatal Errors . . . 457
15.5.4 Countng Error Events Within a Certain Timeframe . . . 458
15.5.5 Tracing Errors to Find Offending Statements . . . 458
15.6 Core Dumping Related Issues . . . 458
15.7 Retrieving Execution Plan Output Using SQreamDB Studio . . . 459
15.7.1 Retrieving Execution Plan Output . . . 459
15.8 Gathering Information for SQream Support . . . 460
15.8.1 Getting Support and Reporting Bugs . . . 460
15.8.2 How SQream Debugs Issues . . . 460
15.8.2.1 Reproduce . . . 460
15.8.2.2 Logs . . . 461
15.8.2.3 Fix . . . 461
15.8.3 Collecting a Reproducible Example of a Problematic Statement . . . 461
15.8.3.1 SQL Syntax . . . 461
15.8.3.2 Parameters . . . 461
15.8.3.3 Example . . . 461
15.8.4 Collecting Logs and Metadata Database . . . 462
15.8.4.1 Examples . . . 462
15.8.5 Using the Command Line Utility: . . . 462

16 Glossary **463**

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	sqloader_as_a_service	
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.
sqream_sql_cli_reference		Describes how to use the SQreamDB client command line tool (CLI) with SQreamDB.
External Storage Platforms		
Amazon Web Services		Describes how to insert data over a native S3 connector.
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 v8.9 / 8.10
- 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),
   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 ingesting_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, 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 v8.9/8.10

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 v8.9 or Amazon Linux 2

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 v8.9 or Amazon Linux 2

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 v8.9

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 v8.9
- Amazon Linux 2

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 v8.9/8.10

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 A40 or A10
Operating System	Red Hat Enterprise Linux v8.9/8.10

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 essential that you tune your system for better performance and stability.

- *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 Basic Input/Output System Settings

The first step when setting your pre-installation configurations is to use the 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.
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 consumption.
HPe: **DIMM voltage**	**Optimized for Performance**	Setting a higher voltage for DIMMs may increase performance.

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 SQreamDB installation requires RHEL8.9/8.10
- Verify the exact 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

NOTE: the default shell of the sqream user must be bash

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.

Set the language of the locale:

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

2.1.1.3.3 Installing Required Software

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

2.1.1.3.3.1 Installing EPEL Repository

```
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.

```
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

```
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

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

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
```

2.1.1.3.3.5 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
cd /usr/bin
sudo ln -s ../local/node-v16.20.0-linux-x64/bin/pm2 pm2
```

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 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
```

2.1.1.3.5 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.

2.1.1.3.6 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
```

2.1.1.3.7 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
```

2.1.1.3.8 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
```

2.1.1.3.9 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
```

2.1.1.3.10 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.

2.1.1.4.1 Before You Begin

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:

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

2.1.1.4.2 Updating the Kernel Headers

1. Update the kernel headers on RHEL:

```
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)
```

2.1.1.4.3 Disabling Nouveau

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
```

2.1.1.4.4 Installing the CUDA Driver

The current recommendation is for CUDA 12.3.2 or CUDA 12.6.1

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

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

2.1.1.4.4.1 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:

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

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:

   - **RHEL8.8/8.9 CUDA 12.3.2 repository (INTEL) installation (Required for H/L Series GPU models):**

```
wget https://developer.download.nvidia.com/compute/cuda/12.3.2/local_
↪installers/cuda-repo-rhel8-12-3-local-12.3.2_545.23.08-1.x86_64.rpm
sudo dnf localinstall cuda-repo-rhel8-12-3-local-12.3.2_545.23.08-1.
↪x86_64.rpm
```

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

2.1.1.4.4.2 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*

2.1.1.4.4.3 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
```

2.1.1.4.4.4 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
```

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

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.

SQreamDB 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:

2.1.3.1 Getting Started

Before installing SQreamDB with Monit, verify that you have followed the required *Pre-Installation Configuration* section.

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

2.1.3.2 Installing Monit on RHEL:

1. Install Monit as a superuser:

```
$ sudo yum install monit
```

2.1.3.3 Building Monit

2.1.3.3.1 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 RHEL directly from the source code:

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

2.1.3.3.2 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.

2.1.3.4 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
   ```

2.1.3.5 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.
$  #                       }
$  #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.
```

(continues on next page)

(continued from previous page)

```
$  #
$  #
$  #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.
$  #                   }
$  #
$  #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.
$  #                   }
```

(continues on next page)

(continued from previous page)

```
$  #
$  #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)
$ 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 Studio on a Stand-Alone Server

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

- *Before You Begin*
- *Installing Studio*

2.2.1.1 Before You Begin

It is essential you have *NodeJS 16 installed*.

2.2.1.2 Installing Studio

1. Copy the SQreamDB Studio package from SQreamDB Artifactory into the target server.

   For access to the SQreamDB Studio package, contact SQreamDB 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 SQreamDB Studio. 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 and shows the following output:

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

5. To make the communication between Studio and SQreamDB secure, in your configuration file do the following:

   a. Change your `port` value to **3109**.

   b. 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
    }
  ]
}
```

   Note that for the `host` value, it is essential that you use the IP address of your SQreamDB machine.

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

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

2.2.1.2.1 Starting Studio

Start Studio by running 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
```

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  │␣
↪ online    │ 0%       │ 15.6mb   │ sqream   │ disabled │
└────┴──────────────────┴─────────────┴─────────┴─────────┴──────────┴────────┴────┴␣
↪──────────┴──────────┴──────────┴──────────┴──────────┘
```

1. 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
```

2. To verify the process is running, use the `pm2 list` command:

```
pm2 list
```

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

5. Verify that you can:

   a. Access Studio from your browser (`http://<IP_Address>:8080`)

b. Log in to SQreamDB

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
```

2.2.1.2.2 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
```

2.2.1.2.3 Maintaining Studio with the Process Manager (PM2)

SQreamDB 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`

2.2.1.2.4 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. Start PM2:

```
pm2 start all
```

7. To access Studio over a secure (HTTPS) connection, follow *NGINX instructions*.

2.2.2 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 Node.js platform that SQreamDB 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 RHEL 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 preferable.

- *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*

2.2.2.1 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

2.2.2.2 Installing NGINX and Adjusting the Firewall

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

Though NGINX is not available in the default RHEL 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
   ```

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.

2.2.2.3 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 SQreamDB 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.

2.2.2.4 Configuring NGINX to use SSL

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

The default RHEL 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.

SQreamDB 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 {
    }
}
```

2.2.2.5 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/;
```

2.2.2.6 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
   ```

2.2.2.7 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
   ```

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

2.3 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.

2.3.1 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.

2.3.2 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

THREE

SQREAMDB ON AWS

Private cloud deployment on AWS provides the AWS's scalable infrastructure, flexible resource management, and cost-efficient services.

The SQreamDB data processing and analytics acceleration platform on AWS marketplace is available here.

- *Before You Begin*
- *Usage Notes*
- *Configuration on AWS*
- *License*
- *Connecting to the Machine*
- *Connecting to SQreamDB*
- *Adding a Signed Certificate to the Cluster*

3.1 Before You Begin

It is essential that you have the following:

- An AWS account
- An existing EC2 key pair
- AWS administrator permissions

3.2 Usage Notes

If you need to access data from an external bucket (one that is not part of the SqreamDB installation or used for `tablespaceURL` or `tempPath`), you must manually grant access. Alternatively, you can copy data to and from the bucket using the AWS_ID and AWS_Secret parameters.

3.3 Configuration on AWS

Under the **CloudFormation > Stacks > Specify stack details** tab, configure the following parameters:

Parameter	Description
`environment`	The identifier used for naming all created resources
`region`	The AWS region where the machines will be deployed. For optimal performance and cost efficiency, the S3 bucket storing Sqream data should be in the same region
`availability_zon`	The availability zone within the specified region to place the machines. It should support GPU-enabled instances
`key_nam`	The name of an existing EC2 key pair in your AWS account, used to log into all created instances
`office_ci`	A list of IP ranges (CIDRs) that are allowed access to the product and SSH access to the machines for security purposes
`sqream_`	The Amazon Machine Image (AMI) pre-configured with Sqream. For Sqream 4.7, use `ami-07d82637b2dab962e`
`ui_inst`	The instance type for the UI server. A machine with 16GB of RAM and moderate CPU resources, such as a `r6i.2xlarge`, is recommended
`md_inst`	The instance type for the metadata and server picker machine. Recommended starting point is a `r6i.2xlarge`, but it may vary depending on your workload
`workers_ins`	The instance type for the worker machines, which must be GPU-enabled. Recommended options include `g6.8xlarge` or `g5.8xlarge`
`workers_cou`	The number of worker machines to be created
`tablespac`	The location where the database will be stored, ideally in the same region as the instances to minimize costs. Important: A `terraform_important` directory will also be created here and should not be deleted unless the installation is completely removed. Deleting this directory prematurely may cause issues during upgrades or changes, leading to a full reinstall of the environment
`tempPath`	The temporary storage path, usually set to `/mnt/ephemeral`, though it can also point to an S3 bucket. This storage is used for running queries and is automatically cleared once the queries are completed

3.4 License

1. Get a list of machines using your AWS console by filtering EC2 instances with:
   - The **worker** keyword
   - The environment name given
   - The AWS instance ID for each EC2
2. Send the machines to SqreamDB to generate license.
3. On each machine, install the license by:
   a. Connecting to the machine (check “connect to AWS machine” section above) - only available from IPs given access by parameter `office_cidrs`.
   b. Create a new file **in this path:**

```
sudo vi /etc/sqream/license.enc
```

4. Place the license given by SqreamDB in it.
5. Wait 1-2 minutes for the Worker to automatically start.

3.5 Connecting to the Machine

For security purposes, all machines are assigned private IP addresses. To enable connections, an EC2 endpoint is configured during installation. You can connect either via the AWS Console UI or through the CLI.

3.5.1 Connecting Using the CLI

You'll need your machine ID and region and the type of key file.

Run the following command:

```
ssh -i <key file> ec2-user@i-<ID> -o ProxyCommand="aws ec2-instance-connect␣
↪opentunnel --instance-id i-<ID> --region=<region>"
```

3.6 Connecting to SQreamDB

During installation, a Network Load Balancer (NLB) named `sqream-<environment>-nlb` is created to route traffic to various machines. After installation, SqreamDB is accessible via the NLB's DNS name. For the SqreamDB UI, use this URL in any browser, or connect to it from third-party software components.

1. To get the URL using AWS Console, copy the DNS of the Network Load Balancer.

3.6.1 Connection Troubleshooting

If you are unable to connect, please ensure the following:

- The license file has been generated and distributed to all Worker nodes.
- Your IP address is included in the `office_cidrs` parameter, as only the specified IPs are allowed access to the cluster.

3.7 Adding a Signed Certificate to the Cluster

To add your signed certificate to the Sqream cluster, follow these steps:

1. Create a new listener for the Network Load Balancer (sqream-<environment>-nlb) using the TLS protocol.
2. A TLS target group that points to the UI machine has already been created for your convenience. You can use it for the new listener. The group name is `sqream-<environment>-nlb-ui-443`.
3. If you require a new DNS, you can retrieve the public IP of the Network Load Balancer by either:
    - Running the host CLI command with the NLB's URL
    - Finding it in the AWS console

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 and drop schemas in the database (the schema must be empty for DRO
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
Column	
SELECT	Select from column
DDL	Column DDL operations
INSERT	insert into the column
UPDATE	update the value of certain columns in existing rows
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
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
ON { CATALOG <catalog_name> [, ...] }
TO <role> [, ...]

-- Grant permissions on the foreign table level:

GRANT {
{SELECT
| DDL } [, ...]
```

(continues on next page)

(continued from previous page)

```
| 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 at the column level:
GRANT
{
  { SELECT
  | DDL
  | INSERT
  | UPDATE } [, ...]
  | ALL [PERMISSIONS]
}
ON
{
  COLUMN <column_name> [,<column_name_2>] IN TABLE <table_name>
  | COLUMN <column_name> [,<column_name_2>] IN FOREIGN TABLE <table_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> [, ...]
```

(continues on next page)

(continued from previous page)

```
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
ON { CATALOG <catalog_name> [, ...] }
FROM <role> [, ...]

-- Revoke permissions on the foreign table level:
```

(continues on next page)

(continued from previous page)

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

-- Revoke function execution permission:
REVOKE {
ALL
| EXECUTE
| DDL }
ON FUNCTION <function_name>
FROM <role>

-- Revoke permissions at the column level:
REVOKE
{
  { SELECT
  | DDL } [, ...]
  | INSERT
  | UPDATE } [, ...]
  | ALL [PERMISSIONS]}
ON
{
  COLUMN <column_name> [,<column_name_2>] IN TABLE <table_name> | COLUMN <column_name>
↪ [,<column_name_2>] IN FOREIGN TABLE <table_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> [,...]
```

(continues on next page)

(continued from previous page)

```
-- 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_name
[IN schema_name, ...]
FOR {
     SCHEMAS
     | TABLES
     | FOREIGN TABLES
     | VIEWS
     | COLUMNS
     | SAVED_QUERIES
     | FUNCTIONS
    }
     { grant_clause
     | DROP grant_clause }
     TO { modified_role_name | public
            }

grant_clause ::=
GRANT
   { CREATE FUNCTION
    | SUPERUSER
    | CONNECT
    | 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 column-level permissions to a role:

```
GRANT SELECT, DDL ON COLUMN column_name IN TABLE schema_name.table_name TO role_name;
```

Grant view-level permissions to a role:

```
GRANT ALL PERMISSIONS ON VIEW "view_name" IN SCHEMA "schema_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 execute function permission from a role:

```
REVOKE EXECUTE ON FUNCTION function_name FROM role_name;
```

Revoke column-level permissions from a role:

```
REVOKE SELECT, DDL FROM COLUMN column_name IN TABLE schema_name.table_name FROM role_
↪name;
```

Revoke view-level permissions from a role:

```
REVOKE ALL PERMISSIONS ON VIEW "view_name" IN SCHEMA "schema_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

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
```

(continues on next page)

(continued from previous page)

```
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 USAGE ON SERVICE sqream 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 USAGE ON SERVICE sqream 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

CREATE  ROLE  jbarker;
GRANT  LOGIN  TO  jbarker;
GRANT  PASSWORD  'action_jacC%k'  TO jbarker;
GRANT  CONNECT  ON  DATABASE  my_database  TO  jbarker;
GRANT USAGE ON SERVICE sqream TO jbarker;
GRANT my_schema_readers TO jbarker;
```

(continues on next page)

(continued from previous page)

```
CREATE  ROLE  lbream;
GRANT  LOGIN  TO  lbream;
GRANT  PASSWORD  'artichoke123O$'  TO lbream;
GRANT  CONNECT  ON  DATABASE  my_database  TO  lbream;
GRANT USAGE ON SERVICE sqream 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 USAGE ON SERVICE sqream 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 USAGE ON SERVICE sqream 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

SQreamDB supports the following external data sources:

Amazon Web Services

HDFS Environment

Google Cloud Platform

Azure Blob Storage

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 SQreamDB first. SQreamDB 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 SQreamDB first.

Although foreign tables can be used without inserting data into SQreamDB, 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.

- *Supported Data Formats*
- *Supported Data Staging*
- *Using Foreign Tables*
- *Error Handling and Limitations*

4.4.1 Supported Data Formats

SQreamDB supports foreign tables using the following file formats:

- 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/....`)
- *Amazon Web Services* buckets
- *HDFS Environment*

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 *Amazon Web Services*, at `s3://sqream-demo-data/nba_players.csv`. We will make note of the file structure, to create a matching `CREATE FOREIGN 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',
       DELIMITER = '\r\n' -- DOS delimited file
     );
```

The file format in this case is CSV, and it is stored as an Amazon Web Services object (if the path is on *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

```
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
```

(continues on next page)

(continued from previous page)

```
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:

```
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  ␣
↪|  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.

```
CREATE TABLE real_nba AS
 SELECT name, team, number, position, age, height, (weight / 2.205) as weight,␣
↪college, salary
 FROM nba
ORDER BY weight;

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:

```
SELECT * FROM nba;

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 SQreamDB, 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 Health Monitoring

The Health Monitoring service enhances observability, enabling shorter investigation times and facilitating both high-level and detailed drill-downs.

- *Before You Begin*
- *Installation*
- *Deployment*
- *Using the Monitor Service*

4.5.1 Before You Begin

It is essential that you follow these prerequisites:

- *Log files* must be saved as `JSON` files
- Configure Grafana authentication, even if you're using LDAP for authentication management

4.5.2 Installation

All Health Monitoring-related installations are on a stand-alone installations.

4.5.2.1 Grafana

Grafana is an open-source analytics and monitoring platform designed for visualizing and analyzing real-time and historical data through customizable dashboards. It offers both an open-source version and an enterprise edition, catering to varying needs and scales of deployment.

For more details, refer to the Grafana specification.

Note: Log in as root user.

4.5.2.1.1 Disabling SELinux

1. Check for the current SELinux status:

```
getenforce
```

2. Open the SELinux configuration file:

```
vim /etc/sysconfig/selinux
```

3. Configure `SELINUX` to be `disabled`:

```
SELINUX=disabled
```

4. Reboot your system:

```
reboot
```

4.5.2.1.2 Installing Grafana via YUM Repository

1. Create a repo file:

```
vim /etc/yum.repos.d/grafana.repo
```

2. Add the following flags to the repo file:

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

3. Install Grafana

```
sudo yum install grafana
```

The installed package performs the following actions:

- Installs the Grafana server binary at `/usr/sbin/grafana-server`
- Copies the init.d script to `/etc/init.d/grafana-server`
- Places the default configuration file in `/etc/sysconfig/grafana-server`
- Copies the main configuration file to `/etc/grafana/grafana.ini`
- Installs the systemd service file (if systemd is supported) as `grafana-server.service`
- By default, logs are written to `/var/log/grafana/grafana.log`

4. Install free type and urw fonts:

```
yum install fontconfig
yum install freetype*
yum install urw-fonts
```

4.5.2.1.3 Enabling the Grafana Service

1. Check for the service status:

```
systemctl status grafana-server
```

2. If not active, start the service:

```
systemctl start grafana-server
```

3. Enable the Grafana service on system boot:

```
systemctl enable grafana-server.service
```

4.5.2.1.4 Modifying your Firewall

1. Enabling the Grafana port:

```
firewall-cmd --zone=public --add-port=3000/tcp --permanent
```

2. Reload Firewall service:

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

4.5.2.2 Prometheus

Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability. Prometheus can be used to scrape and store metrics, enabling real-time monitoring, alerting, and analysis of performance and health.

Your sqream installation includes a Prometheus `yml` file.

1. Download Prometheus.
2. Set the YML path:

```
PROMETHEUS_YML_PATH=<GRAFANA_PROJECT_PATH>/ymls/prometheus.yml
```

3. Run the following script:

```
Prometheus_Server_install () {
        echo "Prometheus_Server_install"
        sudo useradd --no-create-home --shell /bin/false prometheus
        sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
sudo touch /etc/prometheus/prometheus.yml
cat <<EOF | sudo tee /etc/prometheus/prometheus.yml

#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:
                - 0.0.0.0:9090
  - job_name: 'prosesses'
        scrape_interval: 5s
        static_configs:
          - targets:
                - <process exporters iP>:9256
                - <another process exporters iP>:9256
  - job_name: 'nvidia'
```

(continues on next page)

(continued from previous page)

```
        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
EOF
# Assign ownership of the files above to prometheus user
sudo chown -R prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /var/lib/prometheus

# Download prometheus and copy utilities to where they should be in the filesystem
#VERSION=2.2.1
#VERSION=$(curl https://raw.githubusercontent.com/prometheus/prometheus/master/
↪VERSION)
#wget https://github.com/prometheus/prometheus/releases/download/v2.31.1/
↪prometheus-2.31.1.linux-amd64.tar.gz
wget ftp://drivers:drivers11@ftp.sq.l/IT-Scripts+Packages/prometheus-2.31.1.linux-
↪amd64.tar.gz

tar xvzf prometheus-2.31.1.linux-amd64.tar.gz

sudo cp prometheus-2.31.1.linux-amd64/prometheus /usr/local/bin/
sudo cp prometheus-2.31.1.linux-amd64/promtool /usr/local/bin/
sudo cp -r prometheus-2.31.1.linux-amd64/consoles /etc/prometheus
sudo cp -r prometheus-2.31.1.linux-amd64/console_libraries /etc/prometheus

# Assign the ownership of the tools above to prometheus user
sudo chown -R prometheus:prometheus /etc/prometheus/consoles
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool

# Populate configuration files
#cat ./prometheus/prometheus.yml | sudo tee /etc/prometheus/prometheus.yml
#cat ./prometheus/prometheus.rules.yml | sudo tee /etc/prometheus/prometheus.
↪rules.yml
cat <<EOF | sudo tee /etc/systemd/system/prometheus.service
[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
```

(continues on next page)

(continued from previous page)

```
[Install]
WantedBy=multi-user.target
EOF
# systemd
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start prometheus

# Installation cleanup
#rm prometheus-${VERSION}.linux-amd64.tar.gz
#rm -rf prometheus-${VERSION}.linux-amd64
}

Prometheus_Server_install
```

This script generates a Prometheus service.

4. Ensure the user mentioned in the `/etc/systemd/system/prometheus.service` Prometheus service has permissions to trigger Prometheus.

4.5.2.3 Loki and Promtail

Loki is a log aggregation system designed to store and query logs, while Promtail is an agent that collects logs and forwards them to Loki.

1. Download Loki:

```
wget https://github.com/grafana/loki/releases/download/v3.0.0/loki-3.0.0.x86_64.
↪rpm
```

2. Download Promtail:

```
wget https://github.com/grafana/loki/releases/download/v3.0.0/promtail-3.0.0.x86_
↪64.rpm
```

3. Extract the RPM files onto the appropriate machines:

```
sudo rpm -i ~/loki-3.0.0.x86_64.rpm
rpm -i promtail-3.0.0.x86_64.rpm
```

4. Open the `loki.service` file:

```
sudo vim /etc/systemd/system/loki.service
```

5. Configure the service file:

```
[Unit]
Description=Loki

[Service]
ExecStart=/usr/bin/loki -config.file=<LOKI_YML>
User=root
Group=<GROUP>
```

(continues on next page)

(continued from previous page)

```
[Install]
WantedBy=multi-user.target
```

6. Reload systemd to recognize the new service:

```
systemctl daemon-reload
```

7. Restart the Promtail service:

```
sudo systemctl restart promtail
```

4.5.2.4 Exporters

An Exporter is a software component that gathers metrics from various sources (such as hardware, software, or services) and exposes them in a format that Prometheus can scrape and store.

1. Download **`Exporters<https://github.com/utkuozdemir/nvidia_gpu_exporter/releases>`_**.
2. Install Exporters:

```
rpm -i <rpm_file>
```

3. Reload your system:

```
sudo systemctl daemon-reload
```

4. Restart Exporters service:

```
sudo systemctl restart nvidia_gpu_exporter
```

4.5.2.4.1 CPU Exporter

1. Download the CPU Exporter.
2. Extract package content:

```
tar -xvf <package>
```

3. Move the `node_exporter` binary to the `/usr/bin directory`:

```
sudo mv <node_exporter_folder>/node_exporter /usr/bin
```

4. Open the `/etc/systemd/system/node_exporter.service` file:

```
sudo vim /etc/systemd/system/node_exporter.service
```

Add the following to the service file:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
```

(continues on next page)

(continued from previous page)

```
User=prometheus
Group=prometheus
Restart=always
SyslogIdentifier=prometheus
ExecStart=/usr/bin/node_exporter

[Install]
WantedBy=default.target
```

1. Reload the **systemd** manager configuration:

```
sudo systemctl daemon-reload
```

2. Restart the **Node Exporter** service managed by **systemd**

```
sudo systemctl restart node_exporter
```

4.5.2.4.2 Process Exporter

1. (Prometheus Exporter installation)-Slavi
2. Start the Exporter:

```
/usr/bin/process-exporter --config.path /etc/process-exporter/all.yaml --web.
↪listen-address=:9256 &> process_exporter.out &
```

4.5.3 Deployment

4.5.3.1 Grafana

1. Access the Grafana web interface by entering your server IP or host name to the following URL:

```
http://<server ip or host name>:3000/
```

2. Type in `admin` for both user name and password.
3. Change your password.
4. Go to **Data Sources** and choose **prometheus**.
5. Go to **Data Sources** and choose **loki**.
6. Set **URL** as your Prometheus server ip.
7. Go to **Dashboards** and choose **Import**.
8. Import dashboards one by one.

4.5.4 Using the Monitor Service

The Monitor service package includes two files (which must be placed in the same folder):

- `monitor_service` (an executable)
- `monitor_input.json`

4.5.4.1 Configuring the Monitor Service Worker

Before running the monitor service worker, ensure the following Sqream configuration flags are properly set:

Flag	Configuration File	Description
`"cudaMemQuota": 0`	*Worker configuration file*	This setting disables GPU memory usage for the monitor service. Consequently, the Worker must be a non-GPU Worker to avoid exceptions from the monitor service.
`"initialSubscribedServices": "monitor"`	*Worker configuration file*	This configuration specifies that the monitor service should run on a non-GPU Worker. To avoid mixing with GPU Worker processes, the monitor service is set to operate on a designated non-GPU Worker. By default, it runs under the service name `monitor`, but this can be adjusted if needed.
`"enableNvprofMarkers" : false`	*Cluster and session configuration file*	Enabling this flag while using a non-GPU Worker results in exceptions. Ensure this flag is turned off to avoid issues since there are no GPU instances involved.

4.5.4.2 Execution Arguments

When executing the Monitor service, you can configure the following flags:

Flag	Type	Description	State	Default
`-h, --help`	option		Shows help message	
`--host`	string	The SQreamDB host address	Optional	`local-host`
`--port`	integer	The SQreamDB port number	Optional	`5000`
`--database`	string	The SQreamDB database name	Optional	`master`
`--username`	string	The SQreamDB username	Mandatory	`sqream`
`--password`	string	The SQreamDB password	Mandatory	`sqream`
`--clustered`	option	An option if the `server_picker` is running	Optional	`False`
`--service`	string	The SQreamDB service name	Optional	`monitor`
`--loki_host`	string	The Loki instance host address	Optional	`local-host`
`--loki_port`	integer	The Loki port number	Optional	`3100`
`--log_file_path`	string	The path to where log files are saved	Optional	NA
`--metrics_json_p`	string	The path to where the `monitor_input.json` file is stored	Optional	

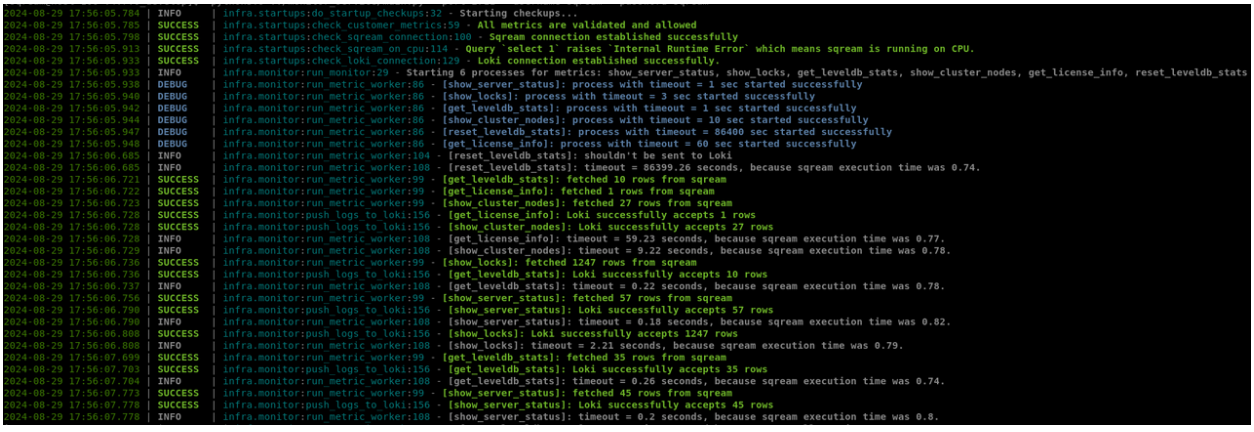
4.5.4.2.1 Example

Execution example:

```
./monitor_service --username=sqream --password=sqream --host=1.2.3.4 --port=2711 --
↪service=monitor --loki_host=1.2.3.5 --loki_port=3100 --metrics_json_path='/home/
↪arielw/monitor_service/monitor_input.json'
```

4.5.4.3 Monitor Service Output Example

Type	Color
Information about monitor service triggering	Blue
Successful insertion	Green
Error	Red



4.6 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*

4.6.1 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.

4.6.2 Usage Notes

4.6.2.1 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;
```

4.6.2.2 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.

4.6.3 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),
(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*

4.6.3.1 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
```

4.6.3.2 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
```

4.6.3.3 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
```

4.6.3.3.1 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>';
```

4.6.4 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.

4.7 Logging

4.7.1 Locating the Log Files

The `logs` directory path is controlled by a `DefaultPathToLogs` cluster flag (legacy config). By default it is set to `~/tmp_log`, the best practice is to set it to `<cluster home>/logs`.

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

4.7.1.1 Log Structure and Contents

By default, logs are saved as `CSV` files. To configure your log files to be saved as `JSON` instead, use the `logFormat` flag in your *legacy config file*.

For effective *Health Monitoring*, it's essential that logs are saved in `JSON` format, as Health Monitoring does not support `CSV` files. If your current logs are in `CSV` format and you require RCA, it's advisable to configure your logs to be saved in both `CSV` and `JSON` formats as outlined above.

Table 2: 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 3: 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
`DEBUG`	Information helpful for debugging
`TRACE`	In-depth information helpful for debugging, such as tracing system function executions and identifying specific error conditions or performance issues.

Table 4: 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 version{v2020.2}"
1002	SYSTEM	Metadata	Metadata server location
1003	SYSTEM	Show all configuration val-	

4.7.1.2 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.

4.7.2 Log Control and Maintenance

4.7.2.1 Changing Log Verbosity

A few configuration settings alter the verbosity of the logs:

Table 5: Log verbosity configuration

Flag	Description	Default	Values
`logClientLev`	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.

4.7.2.2 Changing Log Rotation

A few configuration settings alter the log rotation policy:

Table 6: Log rotation configuration

Flag	Description	Default	Values
`logMaxFileSizeMB`	Sets the size threshold in megabytes after which a new log file will be opened.	`100`	`1` to `1024` (1MB to 1GB)
`logFileRotateTimeFrequency`	Frequency of log rotation	`daily`	`daily`, `weekly`, or `monthly`

4.7.3 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.

4.7.3.1 SQL Syntax

```
SELECT REPORT_COLLECTION(output_path, mode)
;

output_path ::=
   filepath

mode ::=
   log | db | db_and_log
```

4.7.3.2 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>
```

4.7.3.3 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

4.7.3.4 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
```

4.7.4 Troubleshooting with Logs

4.7.4.1 Loading Logs with Foreign Tables

Assuming logs are stored at `/home/rhendricks/sqream_storage/logs/`, a database administrator can access the logs using the *Foreign Tables* concept through SQreamDB.

```
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.

4.7.4.2 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
```

4.7.4.3 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
```

4.7.4.4 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
```

4.7.4.5 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
```

4.8 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*

4.8.1 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
```

4.8.2 Example

- *Creating a Sample Table and Query*
- *Splitting the Query*

4.8.2.1 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;
```

4.8.2.2 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.

4.8.3 Best Practices

4.8.3.1 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.

4.8.3.2 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.

4.8.3.3 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

4.8.3.4 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.

4.9 Monitoring Query Performance

The initial step in query tuning involves a thorough analysis of the query plan and its execution. The query plan and execution details illuminate how SQreamDB handles a query and pinpoint where time resources are consumed. This document offers a comprehensive guide on analyzing query performance through execution plans, with a specific emphasis on recognizing bottlenecks and exploring potential optimization strategies to enhance query efficiency.

It's important to note that performance tuning approaches can vary for each query, necessitating adaptation of recommendations and tips to suit specific workloads. Additionally, for further insights into data loading considerations and other best practices, refer to our *Optimization and Best Practices* guide.

- *Setting Up System Monitoring Preferences*
- *Using the `SHOW_NODE_INFO` Command*
- *Understanding the Query Execution Plan Output*
- *Examples*
- *Further Reading*

4.9.1 Setting Up System Monitoring Preferences

By default, SQreamDB automatically logs execution details for any query that runs longer than 60 seconds. This means that by default, queries shorter than 60 seconds are not logged. You can adjust this parameter to your own preference.

4.9.1.1 Adjusting the Logging Frequency

To customize statement logging frequency to be more frequent, consider reducing the interval from the default 60 seconds to a shorter duration like 5 or 10 seconds. This adjustment can be made by modifying the `nodeInfoLoggingSec` in your SQreamDB *configuration files* and setting the parameter to your preferred value.

```
{
   "compileFlags":{
   },
   "runtimeFlags":{
   },
   "runtimeGlobalFlags":{
      "nodeInfoLoggingSec" : 5,
   },
   "server":{
   }
}
```

After customizing the frequency, please restart your SQreamDB cluster. Execution plan details are logged to the default SQreamDB *log directory* as *message type* `200`.

You can access these log details by using a text viewer or by creating a dedicated *foreign table* to store the logs in a SQreamDB table.

4.9.1.2 Creating a Dedicated Foreign Table to Store Log Details

Utilizing a SQreamDB table for storing and accessing log details helps simplify log management by avoiding direct handling of raw logs.

To create a foreign table for storing your log details, use the following table DDL:

```
CREATE FOREIGN TABLE logs (
  start_marker TEXT,
  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
)
WRAPPER
  csv_fdw
```

(continues on next page)

(continued from previous page)

```
OPTIONS
  (
    LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
    DELIMITER = '|'
  );
```

Use the following query structure as an example to view previously logged execution plans:

```
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
```

4.9.2 Using the SHOW_NODE_INFO Command

The show_node_info command provides a snapshot of the current query plan. Similar to periodically-logged execution plans, SHOW_NODE_INFO displays the compiler's execution plan and runtime statistics for a specified statement at the moment of execution.

You can execute the SHOW_NODE_INFO utility function using sqream sql, *SQream Studio Editor*, or other *third party tool*.

In this example, we inspect a statement with statement ID of 176:

```
SELECT
  SHOW_NODE_INFO(176);
```

(continues on next page)

(continued from previous page)

```
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-
↪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
```

You may also *download the query execution plan* to a CSV file using the **Execution Details View** feature.

4.9.3 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.

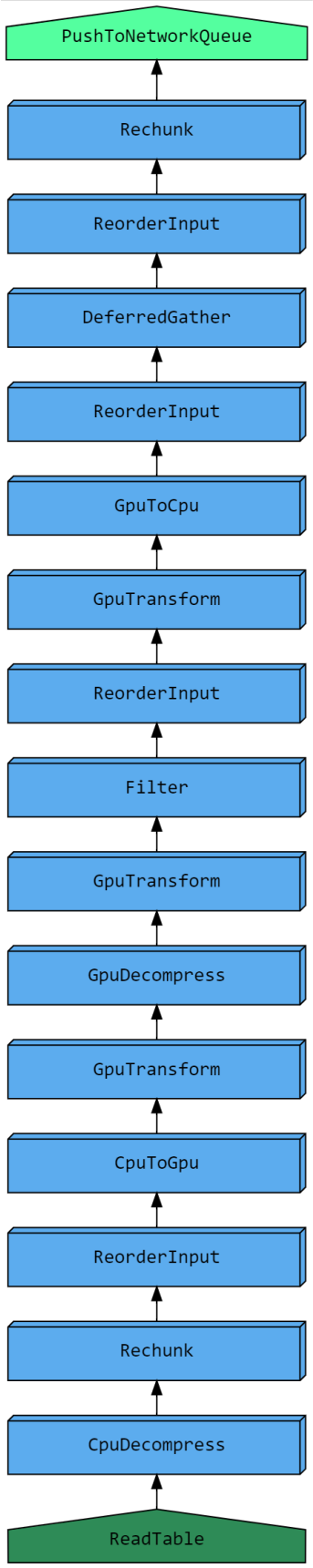


Fig. 1: This graph explains how the query execution details are arranged in a logical order, from the bottom up.

4.9.3.1 Information Presented in the Execution Plan

If the statement has finished, or the statment ID does not exist, the utility returns an empty result set.

4.9.3.2 Commonly Seen Nodes

Table 7: 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*
`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.

4.9.4 Examples

Typically, examining the top three longest running nodes (detailed in the `timeSum` column) can highlight major bottlenecks. The following examples will demonstrate how to identify and address common issues.

- *Spooling to Disk*
- *Queries with Large Result Sets*
- *Inefficient Filtering*
- *Joins with `TEXT` Keys*
- *Sorting on Big `TEXT` Fields*
- *High Selectivity Data*
- *Performance of Unsorted Data in Joins*
- *Manual Join Reordering*

4.9.4.1 Spooling to Disk

When SQreamDB doesn't have enough RAM to process a statement, it will temporarily store overflow data in the `temp` folder on the storage disk. While this ensures that statements complete processing, it can significantly slow down performance. It's important to identify these statements to assess cluster configuration and potentially optimize statement size.

To identify statements that spill data to disk, check the `write` column in the execution details. Nodes that write to disk will display a value (in megabytes) in this column. Common nodes that may write spillover data include `Join` and `LoopJoin`.

4.9.4.1.1 Identifying the Offending Nodes

1. Run a query.

   This example is 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,
      n2.n_name AS nation
    FROM
      lineitem
      JOIN part ON p_partkey = CAST (l_partkey AS INT)
      JOIN orders ON l_orderkey = o_orderkey
```

(continues on next page)

(continued from previous page)

```
      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. Use a foreign table or `SHOW_NODE_INFO` to view the execution information.

   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`:

```
     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,
↪1915MB,inner,4.94
  [...]
```

(continues on next page)

(continued from previous page)

```
  : 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
```

Due to the machine's limited RAM and the large dataset of approximately 10TB, SQreamDB requires spooling.

The total spool used by this query amounts to approximately 20GB (1915MB + 2191MB + 3064MB + 12860MB).

4.9.4.1.2 Common Solutions for Reducing Spool

Solution	Description
Increasing Spool Memory Amount	Increase the amount of spool memory available for the Workers relative to the maximum statement memory. By increasing spool memory, SQreamDB may avoid the need to write to disk. This setting is known as `spoolMemoryGB`. Refer to the *Sizing* guide for details.
Reducing Workers Per Host	Reduce the number of Workers per host and allocate more spool memory to the reduced number of active Workers. This approach may decrease concurrent statements but can enhance performance for resource-intensive queries.

4.9.4.2 Queries with Large Result Sets

When queries produce large result sets, you may encounter a node called `DeferredGather`. This node is responsible for assembling the result set in preparation for sending it to the client.

4.9.4.2.1 Identifying the Offending Nodes

1. Run a query.

   This example is 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
```

(continues on next page)

(continued from previous page)

```
  JOIN customer c ON o_custkey = c_custkey
  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. Use a foreign table or `SHOW_NODE_INFO` to view the execution information.

   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`:

```
  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
```

   If you notice that `DeferredGather` operations are taking more than a few seconds, it could indicate that you're selecting a large amount of data. For example, in this case, the `DeferredGather` with node ID 166 took over 21 seconds.

3. Modify the statement by making the `SELECT` clause more restrictive.

   This adjustment will reduce the `DeferredGather` time from several seconds to just a few milliseconds.

```
SELECT
  DATEPART(year, o_orderdate) AS o_year,
  l_extendedprice * (1 - l_discount / 100.0) as volume,
  n2.n_name as nation
FROM ...
```

4.9.4.2.2 Common Solutions for Reducing Gather Time

Solution	Description
minimizing preparation time	To minimize preparation time, avoid selecting unnecessary columns (e.g., `SELECT * FROM …`) or reduce the result set size by applying more filters.

4.9.4.3 Inefficient Filtering

When executing statements, SQreamDB optimizes data retrieval by *skipping unnecessary chunks*. However, if statements lack efficient filtering, SQreamDB may end up reading excessive data from disk.

4.9.4.3.1 Identifying the Situation

Filtering is considered inefficient when the `Filter` node processes less than one-third of the rows passed into it by the `ReadTable` node.

1. 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
```

(continues on next page)

(continued from previous page)

```
    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;
```

2. Use a foreign table or `SHOW_NODE_INFO` to view the execution information.

   The execution below has been shortened, but note the highlighted rows for `ReadTable` and `Filter`:

```
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
```

(continues on next page)

(continued from previous page)

```
↪| 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
    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
```

Note the following:

- 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.

3. modify the statement by adding a `WHERE` condition on the clustered `l_orderkey` column of the `lineitem` table.

   This adjustment will enable SQreamDB to skip reading unnecessary data.

```
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)

(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 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;
```

```
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
[...]
```

Note the following:

- 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.

4.9.4.3.2 Common Solutions for Improving Filtering

Solution	Description
Clustering keys and ordering data	Utilize clustering keys and naturally ordered data to enhance filtering efficiency.
Avoiding full table scans	Minimize full table scans by applying targeted filtering conditions.

4.9.4.4 Joins with `TEXT` Keys

Joins on long `TEXT` keys may result in reduced performance compared to joins on `NUMERIC` data types or very short `TEXT` keys.

4.9.4.4.1 Identifying the Situation

When a join is inefficient, you may observe that a query spends a significant amount of time on the `Join` node.

Consider these two table structures:

```
CREATE TABLE
  t_a (
    amt FLOAT NOT NULL,
    i INT NOT NULL,
    ts DATETIME NOT NULL,
    country_code TEXT NOT NULL,
    flag TEXT NOT NULL,
    fk TEXT NOT NULL
  );

CREATE TABLE
 t_b (
   id TEXT NOT NULL,
   prob FLOAT NOT NULL,
   j INT NOT NULL
  );
```

1. Run a query.

   In this example, we join `t_a.fk` with `t_b.id`, both of which are `TEXT`.

   ```
   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. Use a foreign table or `SHOW_NODE_INFO` to view the execution information.

   The execution below has been shortened, but note the highlighted rows for `Join`.

```
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
--------+---------+----------------------+------------+--------+-------------------+---------------------+----------------+-------+-------+-------------+--------
[...]
      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
```

   Note the following:

   - The `Join` node is the most time-consuming part of this statement, taking 69.7 seconds to join 1.5 billion records.

4.9.4.4.2 Common Solutions for 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.

Solution	Description
Mapping	Use a dimension table to map `TEXT` values to `NUMERIC` types, and then reconcile these values as needed by joining the dimension table.
Conversion	Use functions like crc64 to convert `TEXT` values into BIGINT hashes directly before running the query. 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.

```
1  SELECT SHOW_NODE_INFO(6);
2  stmt_id | node_id | node_type          | rows       | chunks | avg_rows_in_chunk | time                | parent_node_id | read  | write | comment    | timeSum
3  --------+---------+--------------------+------------+--------+-------------------+---------------------+----------------+-------+-------+------------+--------
4  [...]
5        6 |      19 | GpuTransform       | 1497366528 |     85 |          17825792 | 2020-09-08 18:57:04 |             18 |       |       |            |    1.48
6        6 |      20 | ReorderInput       | 1497366528 |     85 |          17825792 | 2020-09-08 18:57:04 |             19 |       |       |            |       0
7        6 |      21 | ReorderInput       | 1497366528 |     85 |          17825792 | 2020-09-08 18:57:04 |             20 |       |       |            |       0
8        6 |      22 | Join               | 1497366528 |     85 |          17825792 | 2020-09-08 18:57:04 |             21 |       |       | inner      |    6.67
9        6 |      24 | AddSortedMinMaxMet.. |    6291456 |      1 |           6291456 | 2020-09-08 18:55:12 |             22 |       |       |            |       0
10 [...]
11       6 |      32 | ReadTable          |    6291456 |      1 |           6291456 | 2020-09-08 18:55:12 |             31 | 235MB |       | public.t_b |    0.02
12 [...]
13       6 |      43 | CpuDecompress      |
```

4.9.4.5 Sorting on Big `TEXT` Fields

In SQreamDB, a `Sort` node is automatically added to organize data prior to reductions and aggregations. When executing a `GROUP BY` operation on extensive `TEXT` fields, you might observe that the `Sort` and subsequent `Reduce` nodes require a considerable amount of time to finish.

4.9.4.5.1 Identifying the Situation

If you see `Sort` and `Reduce` among your top five longest running nodes, there is a potential issue.

Consider this `t_inefficient` table which contains 60,000,000 rows, and the structure is simple, but with an oversized `country_code` column:

```
CREATE TABLE t_inefficient (
   i INT NOT NULL,
   amt DOUBLE NOT NULL,
   ts DATETIME NOT NULL,
   country_code TEXT NOT NULL,
   flag TEXT NOT NULL,
   string_fk TEXTNOT NULL
);
```

1. Run a query.

```
SELECT
      country_code,
  SUM(amt)
FROM t_inefficient
GROUP BY country_code;


country_code | sum
-------------+-----------
VUT          | 1195416012
GIB          | 1195710372
TUR          | 1195946178
[...]
```

2. Use a foreign table or `SHOW_NODE_INFO` to view the execution information.

```
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
```

(continues on next page)

(continued from previous page)

```
    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 |        |        |                    |  ␣
↪ 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
```

3. Look to see if there's any shrinking that can be done on the `GROUP BY` key:

```
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.

4. Recreate the table with a more restrictive `TEXT(3)`, and examine the difference in performance:

```
CREATE TABLE
  t_efficient AS
SELECT
  i,
  amt,
  ts,
  country_code :: TEXT(3) AS country_code,
  flag
FROM
  t_inefficient;

SELECT
  country_code,
  SUM(amt :: bigint)
FROM
  t_efficient
GROUP BY
  country_code;
```

(continues on next page)

(continued from previous page)

```
country_code | sum
-------------+-----------
VUT          | 1195416012
GIB          | 1195710372
TUR          | 1195946178
[...]
```

This time, the query should be about 91% faster.

4.9.4.5.2 Common Solutions for Improving Sort Performance on `TEXT` Keys

Solution	Description
Using Appropriate Text Length	Define the maximum length of `TEXT` fields in your table structure as small as necessary. For example, if you're storing phone numbers, avoid defining the field as `TEXT(255)` to optimize sort performance.
Optimize Column Length	Execute a query to determine the maximum length of data in the column (e.g., `MAX(LEN(a_column))`) and consider modifying the table structure based on this analysis.

4.9.4.6 High Selectivity Data

In SQreamDB, selectivity refers to the ratio of distinct values to the total number of records within a chunk. It is defined by the formula: $\frac{\text{Distinct values}}{\text{Total number of records in a chunk}}$

SQreamDB provides a hint called `HIGH_SELECTIVITY` that can be used to optimize queries. When you wrap a condition with this hint, it signals to SQreamDB that the result of the condition will yield a sparse output. As a result, SQreamDB attempts to rechunk the results into fewer, fuller chunks for improved performance.

Note: SQreamDB does not apply this optimization automatically because it introduces significant overhead for naturally ordered and well-clustered data, which is the more common scenario.

4.9.4.6.1 Identifying the Situation

This condition is easily identifiable when the average number of rows in a chunk is small, particularly after a Filter operation.

Consider the following execution plan:

```
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
[...]
```

(continues on next page)

(continued from previous page)

```
    30 |      44 | ReadTable          |  77000000 |     74 |           1040540 | 2020-
↪09-10 12:17:09 |             43 | 277MB |       | public.dim |    0.058
```

The table was initially read entirely, containing 77 million rows divided into 74 chunks. After applying a filter node, the output was reduced to just 18,160 relevant rows, which are still distributed across the original 74 chunks. However, all these rows could fit into a single chunk instead of spanning across 74 sparsely populated chunks.

4.9.4.6.2 Common Solutions for Improving Performance with High Selectivity Hints

Solution	Description
Using `HIGH_SELECTIVITY` hint	• When a `WHERE` condition is used on an unclustered column, especially if you anticipate the filter to reduce more than 60% of the result set • When the data is uniformly distributed or random

4.9.4.7 Performance of Unsorted Data in Joins

When data is not well-clustered or naturally ordered, a join operation can take a long time.

4.9.4.7.1 Identifying the Situation

If you identify `Join` and `DeferredGather` as two of the top five longest running nodes, this could indicate a potential issue. Additionally, it's important to consider the number of chunks generated by these nodes in such cases.

Consider this execution plan:

```
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-
```

(continues on next page)

(continued from previous page)

```
↪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 `Join` node performs row matching between table relations, while `DeferredGather` is responsible for gathering necessary column chunks for decompression. Notably, closely monitor the data volume filtered out by the `Filter` node.

The table of 77 million rows was read into 74 chunks. After applying a filter, only 18,160 relevant rows remained, dispersed across these 74 chunks. Ideally, these rows could be consolidated into a single chunk rather than spanning multiple sparse chunks.

4.9.4.7.2 Improving Join Performance when Data is Sparse

To optimize performance in SQreamDB, especially when dealing with aggressive filtering, you can use the *HIGH_SELECTIVITY* hint as described *above*. This hint instructs the compiler to rechunk the data into fewer chunks.

To apply this optimization, wrap your filtering condition (or conditions) with the `HIGH_SELECTIVITY` hint like this:

```
-- 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';
```

4.9.4.8 Manual Join Reordering

When performing joins involving multiple tables, consider changing the join order to start with the smallest tables first.

4.9.4.8.1 Identifying the situation

When joining more than two tables, the `Join` nodes typically represent the most time-consuming operations.

4.9.4.8.2 Changing the Join Order

It's advisable to prioritize joining the smallest tables first. By small tables, we mean tables that retain a relatively low number of rows after applying filtering conditions, regardless of the total row count in the table. Changing the join order in this way can lead to a significant reduction in query runtime. For instance, in specific examples, this approach has resulted in a remarkable 76.64% reduction in query time.

Listing 1: 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
  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
```

Listing 2: 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
```

4.9.5 Further Reading

See our *Optimization and Best Practices* guide for more information about query optimization and data loading considerations.

4.10 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*

4.10.1 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.

4.10.2 Security best practices for SQream DB

4.10.2.1 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.

4.10.2.2 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.

4.10.2.3 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.

4.10.2.4 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.

4.10.2.5 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 `'`)

4.10.2.6 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.

4.11 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.

4.11.1 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
```

4.11.1.1 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.

4.11.2 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_save	Showing a saved query requires `SELECT` permissions on the saved query.
list_saved_	Listing saved queries requires no special permissions.
execute_saved	Executing a saved query requires `USAGE` permissions on the saved query and `SELECT` permissions to access the tables referenced in the query.
drop_save	Dropping a saved query requires `DDL` permissions on the saved query and `SELECT` permissions to access the tables referenced in the query.

4.11.3 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 = ?');
```

4.12 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*

4.12.1 Table design

4.12.1.1 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.

4.12.1.2 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.

4.12.1.3 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.

4.12.1.4 Leveraging Column Data Information

Knowing the data types and their ranges can help design a better table.

4.12.1.4.1 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.

4.12.2 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.

4.12.3 Query Best Practices

This section describes best practices for writing SQL queries.

4.12.3.1 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
   WHERE p_date BETWEEN '2018-07-01' AND '2018-07-31'
   group by 2) AS fact
ON dim.store_id=fact.store_id;
```

4.12.3.2 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 3: 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 4: 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;
```

4.12.3.3 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*.

4.12.3.4 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;
```

4.12.3.5 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.

4.12.3.6 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.

4.12.3.7 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.

4.12.3.8 Reducing JOIN Complexity

Filter and reduce table sizes prior to joining on them

```
SELECT store_name,
       SUM(amount)
FROM dimension 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 dimension 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;
```

4.12.4 Data Loading Considerations

4.12.4.1 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.

4.13 Oracle Migration Guide

This guide is designed to assist those who wish to migrate their database systems from Oracle to SQreamDB. Use this guide to learn how to use the most commonly used Oracle functions with their equivalents in SQreamDB. For functions that do not have direct equivalents in SQreamDB, we provide *User-Defined Functions (UDFs)*. If you need further assistance, our SQream support team is available to help with any custom UDFs or additional migration questions.

- *Using SQream Commands, Statements, and UDFs*
  - *Operation Functions*
  - *Conditional Functions*
  - *Conversion Functions*
  - *Numeric Functions*
  - *Character Functions Returning Character Values*
  - *Character Functions Returning Number Values*
  - *Datetime Functions*
  - *General Comparison Functions*
  - *NULL-Related Functions*
  - *Aggregate Functions*
  - *Analytic Functions*

4.13.1 Using SQream Commands, Statements, and UDFs

4.13.1.1 Operation Functions

Oracle	SQream	Description
+ (unary)	+ (unary)	+a
+	+	a+ b
− (unary)	− (unary)	-a
−	−	a - b
*	*	a * b
/	/	a / b
%	%	a % b
&	&	AND
~	~	NOT
|	|	OR
<<	<<	Shift left
>>	>>	Shift right
XOR	XOR	XOR

4.13.1.2 Conditional Functions

Oracle	SQream	Description
BETWEEN	BETWEEN	Value is in [or not within] the range
CASE	CASE	Tests a conditional expression, depending on the result
COALESCE	COALESCE	Evaluate first non-NULL expression
IN	IN	Value is in [or not within] a set of values
ISNULL	ISNULL	Alias for COALESCE with two expressions
IS_ASCII	IS_ASCII	Test a TEXT for ASCII-only characters
IS_NULL	IS NULL	Check for NULL [or non-NULL] values
DECODE	DECODE	Decodes or extracts binary data from a textual input string

4.13.1.3 Conversion Functions

Oracle	SQream	Description
TO_DATE	CAST	Converts a string to a date

TO_NUMBER	<pre>CREATE OR REPLACE␣
	↪FUNCTION SIGN(n,numeric)
	RETURNS numeric
	AS $$
	CAST(TEXT AS NUMERIC)
	$$ LANGUAGE SQL
	;</pre>
		Converts a string to a number

4.13.1.4 Numeric Functions

Oracle	SQream	Description
ABS	ABS	Calculates the absolute value of an argument
ACOS	ACOS	Calculates the inverse cosine of an argument
ASIN	ASIN	Calculates the inverse sine of an argument
ATAN	ATAN	Calculates the inverse tangent of an argument
ATN2	ATN2	Calculates the inverse tangent for a point (y, x)
BITAND	&	Computes an AND operation on the bits of expr1 and expr2
CEIL	CEILING, CEIL	Calculates the next integer for an argument
COS	COS	Calculates the cosine of an argument
COSH	`CREATE or replace␣ ↪FUNCTION COSH(x double) RETURNS double AS $$ SELECT (exp(x) + exp(-1*x))/2 $$ LANGUAGE SQL ;`	Returns the hyperbolic cosine of n
NA	COT	Calculates the cotangent of an argument
NA	CRC64	Calculates a CRC-64 hash of an argument
NA	DEGREES	Converts a value from radian values to degrees
EXP	EXP	Calculates the natural exponent for an argument
FLOOR	FLOOR	Calculates the largest integer smaller than the argument
LN	LOG	Returns the natural logarithm of n
LOG(b,n)	`CREATE or replace␣ ↪FUNCTION log(b double,␣ ↪n double) RETURNS double AS $$ SELECT (log(n)/log(b)) $$ LANGUAGE SQL ;`	Calculates the natural log for an argument
LOG(10,x)	LOG10	Calculates the 10-based log for an argument
MOD	MOD, %	Calculates the modulus (remainder) of two arguments

continues on next page

Table 8 – continued from previous page

Oracle	SQream	Description
NA	PI	Returns the constant value for π
NANVL	NA	Useful only for floating-point numbers of type
POWER	POWER	Calculates x to the power of y (xy)
NA	SQUARE	Returns the square value of a numeric expression (x2)
NA	RADIANS	Converts a value from degree values to radians
REMAINDER	`CREATE or replace␣ ↪FUNCTION remainder(n1␣ ↪bigint, n2 bigint) RETURNS bigint AS $$ SELECT (n1 - floor(n1/ ↪n2)*n2) $$ LANGUAGE SQL ;`	Returns the arguments any numeric datatype
ROUND (number)	ROUND	Rounds an argument down to the nearest integer
SIGN	`CREATE or replace␣ ↪FUNCTION my_sign(n␣ ↪bigint) RETURNS int AS $$ SELECT case when n < 0␣ ↪then -1 when n = 0␣ ↪then 0 when n > 0 then␣ ↪1 end $$ LANGUAGE SQL ;`	Returns the sign of the input value
SIN	SIN	Calculates the sine
SINH	`CREATE or replace␣ ↪FUNCTION SINH(x double) RETURNS double AS $$ SELECT (exp(x) - exp(- ↪1*x))/2 $$ LANGUAGE SQL ;`	Calculates the hyperbolic sine
SQRT	SQRT	Calculates the square root
TAN	TAN	Calculates the tangent

continues on next page

Table 8 – continued from previous page

Oracle	SQream	Description
TANH	`CREATE or replace␣ ↪FUNCTION TANH(x double) RETURNS double AS $$ SELECT (exp(x) - exp(- ↪1*x))/(exp(x) + exp(- ↪1*x)) $$ LANGUAGE SQL ;`	Calculates the hyperbolic tangent
TRUNC (number)	TRUNC	Rounds a number to its integer representation towards 0
WIDTH_BUCKET(value, low, high, num_buckets)	`CREATE or replace␣ ↪FUNCTION myWIDTH_ ↪BUCKET(value float, low␣ ↪float, high float, num_ ↪buckets int ) RETURNS INT AS $$ select CASE WHEN value < low THEN 0 WHEN value >= high THEN␣ ↪num_buckets + 1 ELSE CEIL(((value - low) / ↪ ((high - low) / num_ ↪buckets))+1)::INT END $$ LANGUAGE SQL ;`	Returns the ID of the bucket into which the value of a specific expression falls
NA	TO_HEX	Converts an integer to a hexadecimal representation

4.13.1.5 Character Functions Returning Character Values

Oracle	SQream	Description
CHR	CHR	Returns the character having the binary equivalent
CONCAT	|| (Concatenate)	Concatenates all the specified strings and returns the final string
INITCAP	NA	Returns char, with the first letter of each word in uppercase
LOWER	LOWER	Returns char, with all letters lowercase
LPAD	NA	Returns expr1, left-padded to length n characters
LTRIM	LTRIM	Removes from the left end of char
NLS_INITCAP	NA	Returns char, with the first letter of each word in uppercase
NLS_LOWER	NA	Returns char, with all letters lowercase
NLSSORT	NA	Returns the string of bytes used to sort char
NLS_UPPER	NA	Returns char, with all letters uppercase
REGEXP_REPLACE	REGEXP_REPLACE	Replaces a substring in a string that matches a specified pattern
REGEXP_SUBSTR	REGEXP_SUBSTR	Returns a substring of an argument that matches a regular expression
REPLACE	REPLACE	Replaces characters in a string
RPAD	NA	Right pads a string to a specified length
RTRIM	RTRIM	Removes the space from the right side of a string
SOUNDEX	NA	Converts a normal string into a string of the SOUNDEX type
SUBSTR	SUBSTRING, SUBSTR	Returns a substring of an argument
TRANSLATE	NA	Returns `expr` with all occurrences of each character in `from_string`, replaced by its corresponding character
TRIM	TRIM	Trims whitespaces from an argument
UPPER	UPPER	Converts an argument to an upper-case equivalent
NA	REPEAT	Repeats a string as many times as specified
NA	REVERSE	Returns a reversed order of a character string
NA	LEFT	Returns the left part of a character string with the specified number of characters
NA	RIGHT	Returns the right part of a character string with the specified number of characters
NA	LIKE	Tests if a string matches a given pattern. SQL patterns
NA	RLIKE	Tests if a string matches a given regular expression pattern. POSIX regular expressions
NA	ISPREFIXOF	Checks if one string is a prefix of the other

4.13.1.6 Character Functions Returning Number Values

Oracle	SQream	Description
ASCII	NA	Returns the decimal representation in the database character set
INSTR	CHARINDEX	Search string for substring
LENGTH	CHAR_LENGTH	Calculates the length of a string in characters
NA	LEN	Calculates the number of characters in a string. (This function is provided for SQL Server compatibility)
NA	OCTET_LENGT	Calculates the number of bytes in a string
NA	CHARINDEX	Returns the starting position of a string inside another string
NA	PATINDEX	Returns the starting position of a string inside another string
REGEXP_COUNT	REGEXP_COUNT	Calculates the number of matches of a regular expression
REGEXP_INSTR	REGEXP_INSTR	Returns the start position of a regular expression match in an argument

4.13.1.7 Datetime Functions

Oracle	SQream	Description
ADD_MONTHS	NA	Returns a number of months are added to a specified date
NA	CURDATE	This function is equivalent to CURRENT_DATE
CURRENT_DATE	CURRENT_DATE	Returns the current date as DATE
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	Equivalent to GETDATE
DBTIMEZONE	NA	Returns the value of the database time zone
EXTRACT (datetime)	EXTRACT	ANSI syntax for extracting date or time element from a date expre
FROM_TZ	NA	Converts a timestamp value and a time zone
LAST_DAY	EOMONTH	Returns the last day of the month in which the specified date value
NA	CURRENT_TIMESTAMP	Returns the current date and time in the session time zone
MONTHS_BETWEEN	DATEDIFF	Returns the number of months between specified date values
NEW_TIME	NA	returns the date and time in time zone
NEXT_DAY	NA	Returns the date of the first weekday that is later than a specified d
NUMTODSINTERVAL	NA	Converts n to an INTERVAL DAY TO SECOND literal
NUMTOYMINTERVAL	NA	Converts number n to an INTERVAL YEAR TO MONTH literal
ORA_DST_AFFECTED	NA	Changing the time zone data file
ORA_DST_CONVERT	NA	Changing the time zone data file for specify error handling
ORA_DST_ERROR	NA	Changing the time zone data file for takes as an argument a datetim
ROUND (date)	ROUND	Rounds an argument down to the nearest integer, or an arbitrary pr
SESSIONTIMEZONE	NA	Returns the time zone of the current session
SYS_EXTRACT_UTC	NA	extracts the UTC from a datetime value with time zone offset
SYSDATE	SYSDATE	Equivalent to GETDATE
SYSTIMESTAMP	CURRENT_TIMESTAMP	Returns the current timestamp
TO_CHAR (datetime)	NA	Converts a date value to a string in a specified format
TO_TIMESTAMP	NA	Converts datatype to a value of TIMESTAMP datatype
TO_TIMESTAMP_TZ	NA	Converts datatype to a value of TIMESTAMP WITH TIME ZON
TO_DSINTERVAL	NA	Converts a character string of CHAR datatype
TO_YMINTERVAL	NA	Converts a character string of CHAR datatype
TRUNC (date)	TRUNC	Truncates a date element down to a specified date or time element
TZ_OFFSET	NA	Returns the time zone offset
NA	DATEADD	Adds or subtracts an interval to DATE or DATETIME value.

Table 9 – continued from previous page

Oracle	SQream	Description
NA	DATEDIFF	Calculates the difference between two DATE or DATETIME expr
NA	DATEPART	Extracts a date or time part from a DATE or DATETIME value
NA	GETDATE	Returns the current date and time of the system
NA	TO_UNIXTS, TO_UNIXTSMS	Converts a DATETIME value to a BIGINT representing a UNIX t
NA	FROM_UNIXTS, FROM_UNIXTSMS	Converts a BIGINT representing a UNIX timestamp to a DATETI

4.13.1.8 General Comparison Functions

Oracle	SQream	Description
GREATEST	NA	Returns the greatest of a list of one or more expressions
LEAST	NA	Returns the least of a list of one or more expressions

4.13.1.9 NULL-Related Functions

Oracle	SQream	Description
COALESCE	COALESCE	Returns the first non-null
LNNVL	NA	Provides a concise way to evaluate a condition when one or both operands of the condition may be null
NANVL	NA	Takes as arguments any numeric data type or any nonnumeric data type
NULLIF	IS NULL	If they are equal, then the function returns null
NVL	ISNULL	Replace null (returned as a blank) with a string in the results of a query
NVL2	NA	Determine the value returned by a specified expression is null or not null

4.13.1.10 Aggregate Functions

Oracle	SQream	Description
AVG	AVG	Calculates the average of all of the values
CHECKSUM	NA	Detect changes in a table
COLLECT	NA	Takes as its argument a column of any type and creates a nested table
CORR	CORR	Calculates the Pearson correlation coefficient
COUNT	COUNT	Calculates the count of all of the values or only distinct values
COVAR_POP	COVAR_POP	Calculates population covariance of values
COVAR_SAMP	COVAR_SAMP	Calculates sample covariance of values
CUME_DIST	CUME_DIST	Calculates the cumulative distribution of a value in a group of values
FIRST	FIRST_VALUE	The FIRST_VALUE function returns the value located in the selected column
GROUP_ID	NA	Distinguishes duplicate groups resulting from a GROUP BY specification
GROUPING	NA	Distinguishes superaggregate rows from regular grouped rows
GROUPING_ID	NA	Returns a number corresponding to the GROUPING bit vector associated with
LAST	LAST_VALUE	The LAST_VALUE function returns the value located in the selected column
NA	NTH_VALUE	The NTH_VALUE function returns the value located in the selected column of
MAX	MAX	Returns maximum value of all values

Table 10 – continued from previous page

Oracle	SQream	Description
MEDIAN	NA	Calculates the median value of a column
MIN	MIN	Returns minimum value of all values
NA	NTILE	Divides an ordered data set into a number of buckets
PERCENTILE_CONT	PERCENTILE_CONT	Inverse distribution function that assumes a continuous distribution model
PERCENTILE_DISC	PERCENTILE_DISC	Inverse distribution function that assumes a discrete distribution model
PERCENT_RANK	PERCENT_RANK	Range of values returned by PERCENT_RANK is 0 to 1, inclusive
RANK	RANK	Calculates the rank of a value in a group of values
DENSE_RANK	DENSE_RANK	Computes the rank of a row in an ordered group of rows
STATS_BINOMIAL_TEST	NA	Exact probability test used for dichotomous variables
STATS_CROSSTAB	NA	Method used to analyze two nominal variables
STATS_F_TEST	NA	Tests whether two variances are significantly different
STATS_KS_TEST	NA	Kolmogorov-Smirnov function that compares two samples to test
STATS_MODE	NA	Takes as its argument a set of values and returns the value
STDDEV	STDDEV	Returns the population standard deviation of all input values
STDDEV_POP	STDDEV_POP	Calculates population standard deviation of values
STDDEV_SAMP	STDDEV_SAMP	Calculates sample standard deviation of values
SUM	SUM	Calculates the sum of all of the values or only distinct values
VAR_POP	VAR_POP	Calculates population variance of values
VAR_SAMP	VAR_SAMP	Calculates sample variance of values
VARIANCE	VAR, VARIANCE	Returns the variance of expr

4.13.1.11 Analytic Functions

Oracle	SQrea	Description
NA	MODE	The MODE function returns the most common value in the selected column. If there are no repeating values, or if there is the same frequency of multiple values, this function returns the top value based on the ORDER BY clause
FEATURE_DETAI	NA	Returns feature details for each row in the selection
FEATURE_ID	NA	Returns the identifier of the highest value feature for each row
FEATURE_SET	NA	Returns a set of feature ID and feature value pairs for each row
FEATURE_VALUE	NA	Returns a feature value for each row in the selection
LEAD	LEAD	Returns a value from a subsequent row within the partition of a result set
LAG	LAG	Returns a value from a previous row within the partition of a result set
PREDICTION	NA	Returns a prediction for each row in the selection
PREDICTION_COST	NA	Returns prediction details for each row in the selection
PREDICTION_DETAI	NA	Returns prediction details for each row in the selection
PREDICTION_PROBA	NA	Returns a probability for each row in the selection
PREDICTION_SET	NA	Returns a set of predictions with either probabilities or costs for each row
ROW_NUMBER	ROW_N	Assigns a unique number to each row to which it is applied

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
blockNewVarcharObjects	SUPERUSER	Session	Disables the creation of new ta
cacheDiskDir	Anyone	Session	Sets the ondisk directory locat
cacheDiskGB	Anyone	Session	Sets the amount of memory (G
cacheEvictionMilliseconds	Anyone	Session	Sets how long the cache stores
cachePartitions	Anyone	Session	Sets the number of partitions t
cachePersistentDir	Anyone	Session	Sets the persistent directory lo
cachePersistentGB	Anyone	Session	Sets the amount of data (GB) i
cacheRamGB	Anyone	Session	Sets the amount of memory (G
checkCudaMemory	SUPERUSER	Session	Sets the pad device memory al
compilerGetsOnlyUFs	SUPERUSER	Session	Sets the runtime to pass only u
clientReconnectionTimeout	Anyone	Cluster	Reconnection time out for the
copyToRestrictUtf8	SUPERUSER	Session	Sets the custom bin size in the
csvLimitRowLength	SUPERUSER	Cluster	Sets the maximum supported C
cudaMemcpyMaxSizeBytes	SUPERUSER	Session	Sets the chunk size for copying
CudaMemcpySynchronous	SUPERUSER	Session	Indicates if copying from/to G
defaultGracefulShutdownTimeoutMinutes	SUPERUSER	Cluster	Used for setting the amount of
developerMode	SUPERUSER	Session	Enables modifying R&D flags.
enableDeviceDebugMessages	SUPERUSER	Session	Checks for CUDA errors after
enableNvprofMarkers	SUPERUSER	Session	Activates the Nvidia profiler (n
extentStorageFileSizeMB	SUPERUSER	Cluster	Sets the minimum size in meb
flipJoinOrder	Anyone	Session	Reorders join to force equijoin
gatherMemStat	SUPERUSER	Session	Monitors all pinned allocations
increaseChunkSizeBeforeReduce	SUPERUSER	Session	Increases the chunk size to red
increaseMemFactors	SUPERUSER	Session	Adds rechunker before expens
leveldbWriteBufferSize	SUPERUSER	Session	Sets the buffer size.
maxAvgBlobSizeToCompressOnGpu	Anyone	Session	Sets the CPU to compress colu
memoryResetTriggerMB	SUPERUSER	Session	Sets the size of memory used c
mtRead	SUPERUSER	Session	Splits large reads to multiple sm
mtReadWorkers	SUPERUSER	Session	Sets the number of workers to
orcImplicitCasts	SUPERUSER	Session	Sets the implicit cast in orc file
QueryTimeoutMinutes	Anyone	Session	Terminates queries that have e
sessionTag	Anyone	Session	Sets the name of the session ta
spoolMemoryGB	Anyone	Session	Sets the amount of memory (G
statementLockTimeout	SUPERUSER	Session	Sets the timeout (seconds) for
useLegacyDecimalLiterals	SUPERUSER	Session	Interprets decimal literals as D
cpuReduceHashtableSize	SUPERUSER	Session	Sets the hash table size of the C
maxPinnedPercentageOfTotalRAM	SUPERUSER	Session	Sets the maximum percentage
memMergeBlobOffsetsCount	SUPERUSER	Session	Sets the size of memory used c
queueTimeoutMinutes	Anyone	Session	Terminates queries that have e
timezone	Anyone	Session	The timezone flag dictates the

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
healerDetectionFrequencySeconds	SUPERUSER	Determines the default frequency for the healer to check that its conditions are met.		86,400 (seconds)
isHealerOn	SUPERUSER	Enables the Query Healer, which periodically examines the progress of running statements and logs statements exceeding the healerMaxInactivityHours flag setting.	boole	TRUE
logFormat	SUPERUSER	Determines the file format of the log files. Format may by csv, json, or both (all logs will be written saved both as csv and json files)	string	csv
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
maxConnectionInactivitySeconds	SUPERUSER	Determines the maximum period of session idleness, after which the connection is terminated.	bigint	86,400 (seconds)
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).	boole	FALSE
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
limitQueryMemoryGB	Anyone	Prevents a query from processing more memory than the defined value.	uint	100000

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
 "logFormat": "csv","json"
}
```

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,
 "logMaxFileSizeMB": 20,
```

(continues on next page)

(continued from previous page)

```
 "logFileRotateTimeFrequency": "daily",
 "waitForClientSeconds": 18000
}
```

5.1.3.3 Metadata Configuration File

When attempting to free up disk space in Oracle Object Store by executing `DELETE`, `cleanup`, `TRUNCATE`, or `DROP`, ensure that the following four flags are consistently configured in both the *Worker* and Metadata configuration files.

Flag Name	Who May Configure	Description	Data Type	Default Value
`OciBaseRegion`	SUPERUSER	Sets your Oracle Cloud Infrastructure (OCI) region	String	NA
`OciVerifySsl`	SUPERUSER	Controls whether SSL certificates are verified. By default, verification is enabled. To disable it, set the variable to `FALSE`	boolean	`TRUE`
`OciAccessKey`	SUPERUSER	Sets your Oracle Cloud Infrastructure (OCI) access key	String	NA
`OciAccessSecret`	SUPERUSER	Sets your Oracle Cloud Infrastructure (OCI) access secret	String	NA

The following is an example of the metadata configuration file:

```
{
 "OciBaseRegion": "us-ashburn-1",
 "OciVerifySsl": false,
 "OciAccessKey": "587f59dxxxxxxxxxxxxxxxxxxxxxxxx",
 "OciAccessSecret": "LrSEb+RZgxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
}
```

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 has been configured as an authentication service for SQreamDB, authentication for all existing and newly added roles is handled by an LDAP server. The exception for this rule is the out-of-the-box administrative `sqream` role, which will always use the conventional SQreamDB authentication instead LDAP authentication.

- *Before You Begin*
- *Setting LDAP Authentication Management*
- *Disabling LDAP Authentication*

5.2.1 Before You Begin

- If SQreamDB is being installed within an environment where LDAP is already configured, it is best practice to ensure that the newly created SQreamDB role names are consistent with existing LDAP user names.
- When setting up LDAP for an existing SQreamDB installation, it's recommended to ensure that newly created LDAP usernames match existing SQreamDB role names. If SQreamDB roles were not configured in LDAP or have different names, they'll be recreated in SQreamDB as roles without login capabilities, permissions, or default schemas.

5.2.2 Setting LDAP Authentication Management

To set LDAP authentication for SQreamDB, choose one of the following configuration methods:

- *Basic Method*
- *Advanced Method*

5.2.2.1 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.

5.2.2.1.1 Flags

Flag	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

5.2.2.1.2 Basic Method Configuration

Only roles with admin privileges or higher may enable LDAP Authentication.

1. Set the `authenticationMethod` flag:

```
ALTER SYSTEM SET authenticationMethod = 'ldap';
```

2. Set the `ldapIpAddress` flag:

```
ALTER SYSTEM SET ldapIpAddress = '<ldaps://...>';
```

3. Set the `ldapPrefix` flag:

```
ALTER SYSTEM SET ldapPrefix = '<DN_binding_string_prefix>=';
```

4. Set the `ldapSuffix` flag:

```
ALTER SYSTEM SET ldapSuffix  = '<DN_binding_string_suffix>';
```

5. To set the `ldapPort` flag (optional), run:

```
ALTER SYSTEM SET ldapPort = <port_number>
```

6. To set the `ldapConnTimeoutSec` flag (optional), run:

```
ALTER SYSTEM SET ldapConnTimeoutSec = <15>;
```

7. Restart all sqreamd servers.

5.2.2.1.3 Example

After completing the setup above, we can 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
```

5.2.2.2 Advanced Method

This method lets users be grouped into categories. Each category can then be given or denied access to SQreamDB, giving administrators control over access.

5.2.2.2.1 Flags

Flag	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`	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
`ldapSearcAttribute`	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
`ldapSearcFilter`	Filters `ldapAdvancedMode` authentication. `ALTER SYSTEM SET ldapSearchFilter = '(<attribute>=<value>)(<attribute2>=<value2>)(...)';`
`ldapGetAttributeLi`	Enables you to include LDAP user attributes, as they appear in LDAP, in your SQreamDB metadata. After having set this flag, you may execute the ldap_get_attr utility function which will show you the attribute values associated with each SQreamDB role.

5.2.2.2.2 Preparing LDAP Users

If installing SQreamDB in an environment with LDAP already set up, it's best to ensure the new SQreamDB role names match the existing LDAP user names.

It is also recommended to:

- Group Active Directory users so that they may be filtered during setup, using the `ldapSearchFilter` flag.
- Provide a unique attribute to each user name, such as an employee ID, to be easily searched for when using the `ldapSearchAttribute` flag.

5.2.2.2.3 Preparing SQreamDB Roles

For a SQreamDB admin to be able to manage role permissions, for every Active Directory user connecting to SQreamDB, there must be an existing SQreamDb role name that is consistent with existing LDAP user names.

You may either rename SQream roles or create new roles, such as in the following example:

1. Create a new role:

```
CREATE ROLE role12345;
```

2. Grant the new role login permission:

```
GRANT LOGIN TO role12345;
```

3. Grant the new role `CONNECT` permission:

```
GRANT CONNECT ON DATABASE master TO role12345;
```

5.2.2.2.4 Advanced Method Configuration

Only roles with admin privileges and higher may enable LDAP Authentication.

1. Configure your LDAP server bind password to be stored in SQreamDB metadata:

```
GRANT PASSWORD <'binding_user_password'> TO ldap_bind_dn_admin_password;
```

   This action emulates the execution of a `GRANT` command, but it's solely necessary for configuring the password. Note that `ldap_bind_dn_admin_password` is not an actual SQreamDB role. This password is encrypted within your SQreamDB metadata.

2. Set the `authenticationMethod` flag:

```
ALTER SYSTEM SET authenticationMethod = 'ldap';
```

3. Set the `ldapAdvancedMode` flag:

```
ALTER SYSTEM SET ldapAdvancedMode = true;
```

4. Set the `ldapIpAddress` flag:

```
ALTER SYSTEM SET ldapIpAddress = '<ldaps://<IpAddress>';
```

5. Set the `ldapBindDn` flag:

```
ALTER SYSTEM SET ldapBindDn = <binding_user_DN>;
```

6. Set the `ldapBaseDn` flag:

```
ALTER SYSTEM SET ldapBaseDn = '<search_root_DN>';
```

7. Set the `ldapSearchAttribute` flag:

```
ALTER SYSTEM SET ldapSearchAttribute = '<search_attribute>';
```

8. To set the `ldapSearchFilter` flag (optional), run:

```
ALTER SYSTEM SET ldapSearchFilter = '(<attribute>=<value>)(<attribute2>=<value2>
↪)[...]';
```

9. To set the `ldapPort` flag (optional), run:

```
ALTER SYSTEM SET ldapPort = <port_number>
```

10. To set the `ldapConnTimeoutSec` flag (optional), run:

```
ALTER SYSTEM SET ldapConnTimeoutSec = <15>;
```

11. To set the `ldapGetAttributeList` flag (optional), run:

```
ALTER SYSTEM SET ldapGetAttributeList = <'ldap_attribute1'>,<'ldap_attribute2
↪'>,<'ldap_attribute3'>,[,...];
```

    a. To see the LDAP user attributes associated with SQreamDB roles in your metadata, execute the ldap_get_attr utility function.

12. Restart all sqreamd servers.

5.2.2.2.5 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`.

```
GRANT PASSWORD 'LdapPassword12#4%' TO ldap_bind_dn_admin_password;

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 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
```

5.2.3 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.

5.3 Single Sign-On

Here you can learn how to configure a SSO login for SQreamDB Acceleration Studio by integrating with an identity provider (IdP). A SSO authentication allows users to authenticate once and then seamlessly access SQreamDB as one of multiple services.

- *Before You Begin*
- *Setting SQreamDB Acceleration Studio*

5.3.1 Before You Begin

It is essential you have the following installed:

- SQreamDB Acceleration Studio v5.9.0
- There should be an NGINX (or similar) service installed on your Acceleration Studio machine, which will serve as a reverse proxy. This service will accept HTTPS traffic from external sources and communicate with Studio via HTTP internally
- You have *LDAP* set as your authentication management service.

5.3.2 Setting SQreamDB Acceleration Studio

1. In your `sqream_legacy.json` file, add the `ssoValidateUrl` flag with your IdP URL.

   Example:

```
"ssoValidateUrl": "https://auth.pingone.eu/9db5d1c6-6dd6-4e40-b939-e0e4209e0ac5/
↪as/userinfo"
```

2. Set Acceleration Studio to use SSO by adding the following flag to your `sqream_admin_config.json` file:
   - `mfaRedirectUrl` flag with your redirect URL

   Example:

```
"mfaRedirectUrl": "https://auth.pingone.eu/9db5d1c6-6dd6-4e40-b939-e0e4209e0ac5/
↪as/authorize?client_id=e5636823-fb99-4d38-bbd1-6a46175eddab&redirect_uri=https:/
↪/ivans.sq.l/login&response_type=token&scope=openid profile p1:read:user",
```

   If Acceleration Studio is not yet installed, you can set both URLs during its installation process.

1. Restart SQreamDB.

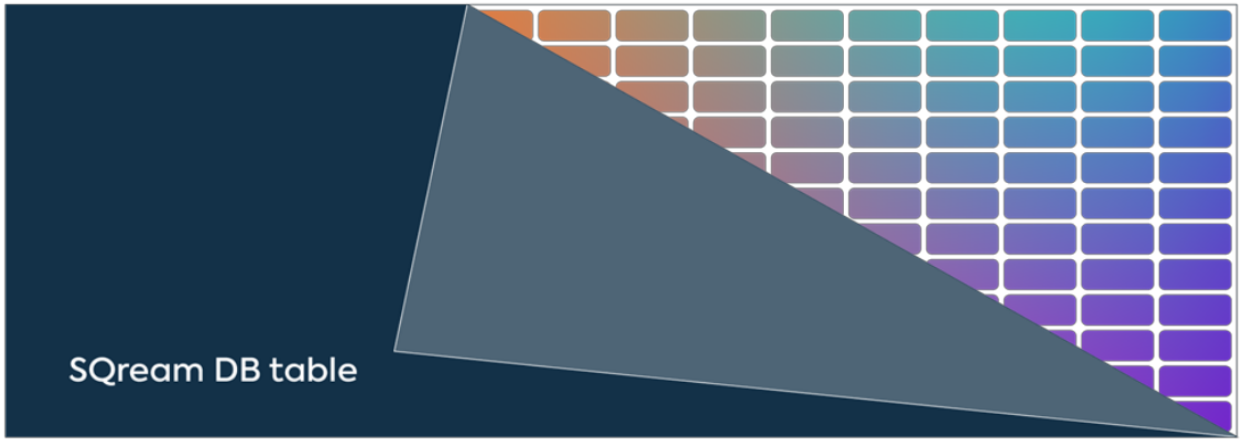
2. Restart SQreamDB Acceleration Studio.

CHAPTER

SIX

ARCHITECTURE

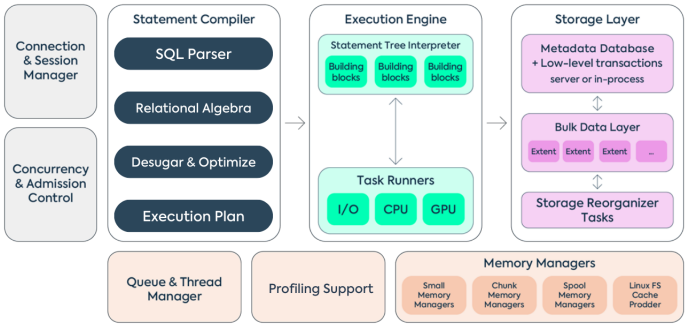
The *Internals and Architecture*, *Sizing*, and *Filesystem and Usage* guides are walk-throughs for end-users, database administrators, and system architects who wish to get familiarized with the SQreamDB system and its unique capabilities.



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*.

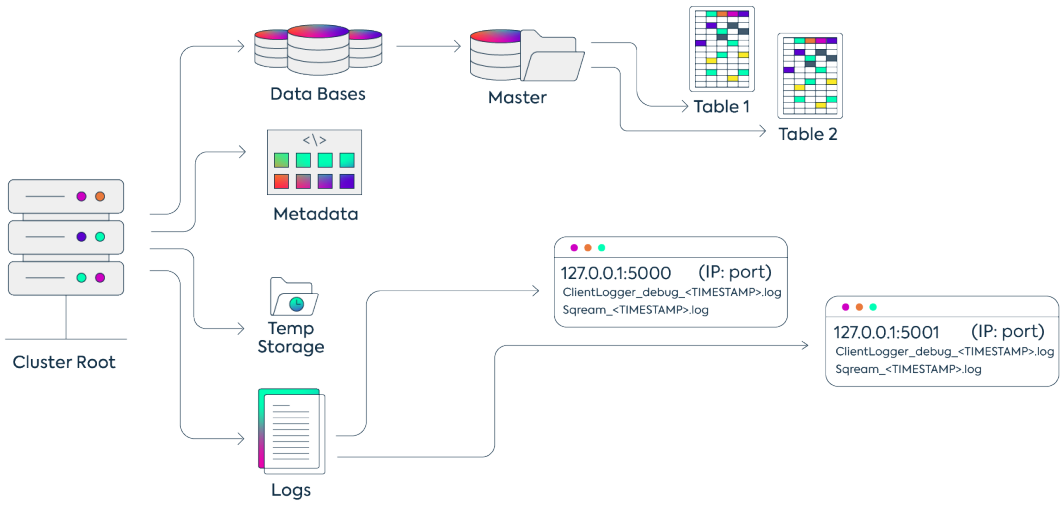
6.2 Filesystem and Usage

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.

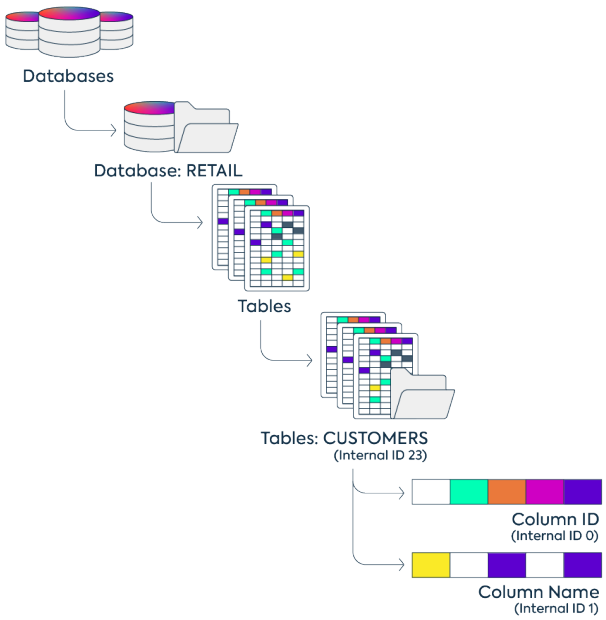
SQreamDB storage cluster directories

- *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:

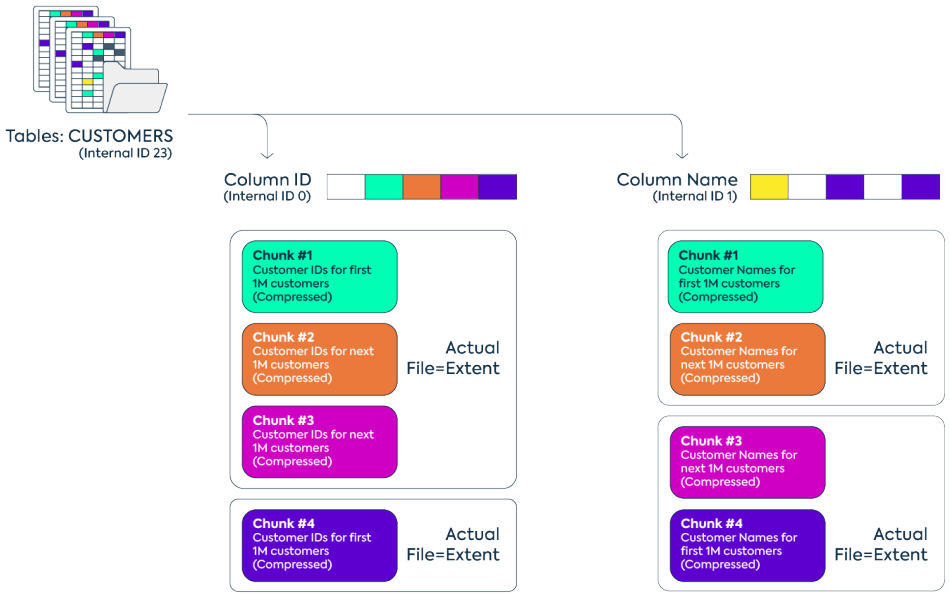
```
master=> SELECT table_name, table_id FROM sqream_catalog.tables WHERE table_name =
↪'customers';
table_name | table_id
-----------+---------
 customers |       23
```

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 *legacy 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 A10 (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

$$limitQueryMemoryGB = \frac{\text{Total RAM - Internal Operation - metadata Server - Server picker}}{\text{Number of Workers}}$$

$$spoolMemoryGB = limitQueryMemoryGB - 50GB$$

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": "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,
        "insertCompressors": 8,
        "insertParsers": 8,
        "isUnavailableNode": false,
        "logBlackList": "webui",
        "logDebugLevel": 6,
        "nodeInfoLoggingSec": 60,
        "useClientLog": true,
        "useMetadataServer": true,
        "spoolMemoryGB": 175,
        "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

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 view the following:

- User information
- Connection type
- SQream version
- SQream Studio version
- License expiration date
- License storage capacity
- *Activity report*
- Log out

7.1.4 View Activity Report

The **View activity report** menu item enables you to monitor storage and resource usage, including GPUs, workers, and machines. You can select different time frames to view cluster activity and export the data as a PDF for use in financial records, briefings, or quarterly and yearly reports.

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
$$
SELECT 1;
SELECT 2;
SELECT 3;
$$
```

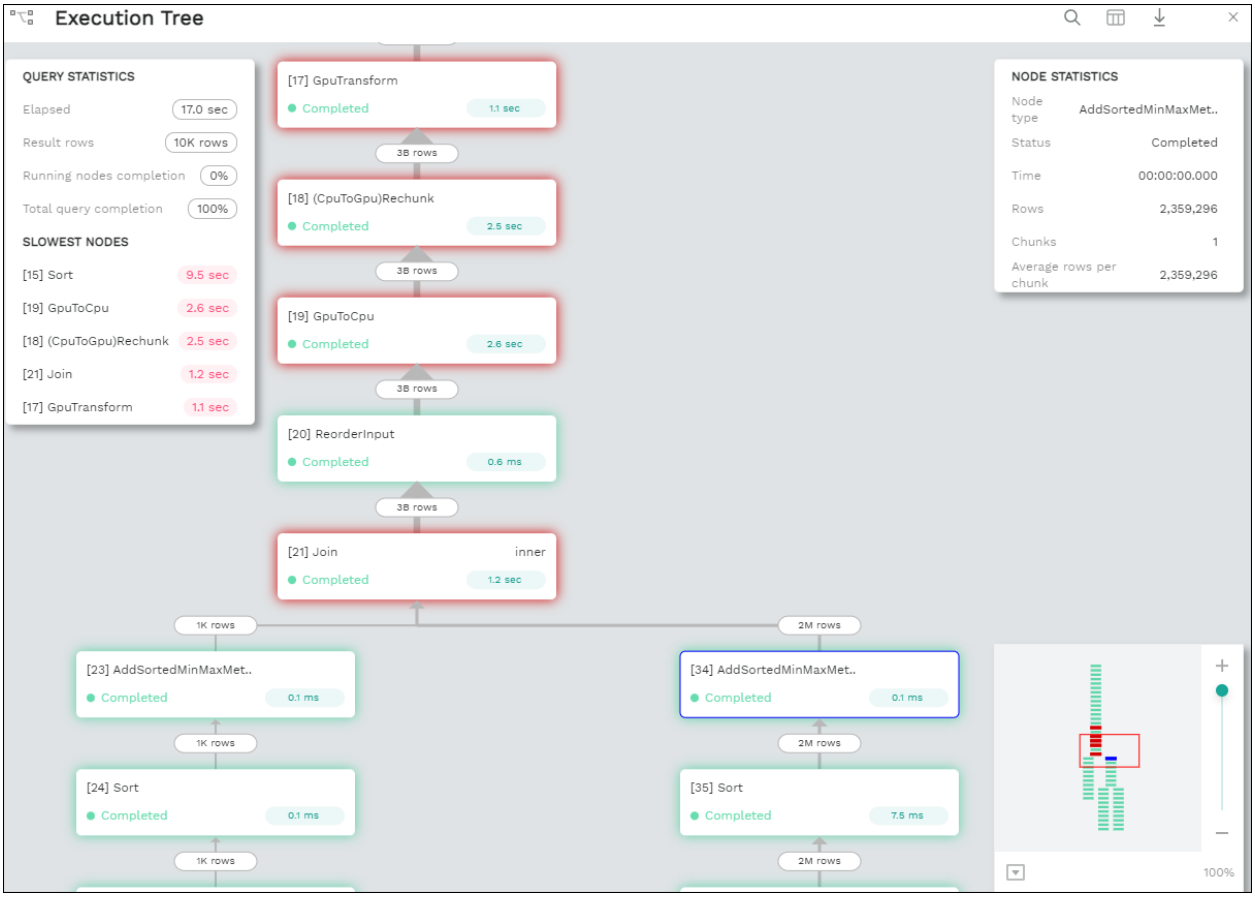
7.2.4.2 Execution Details View

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.

CHAPTER

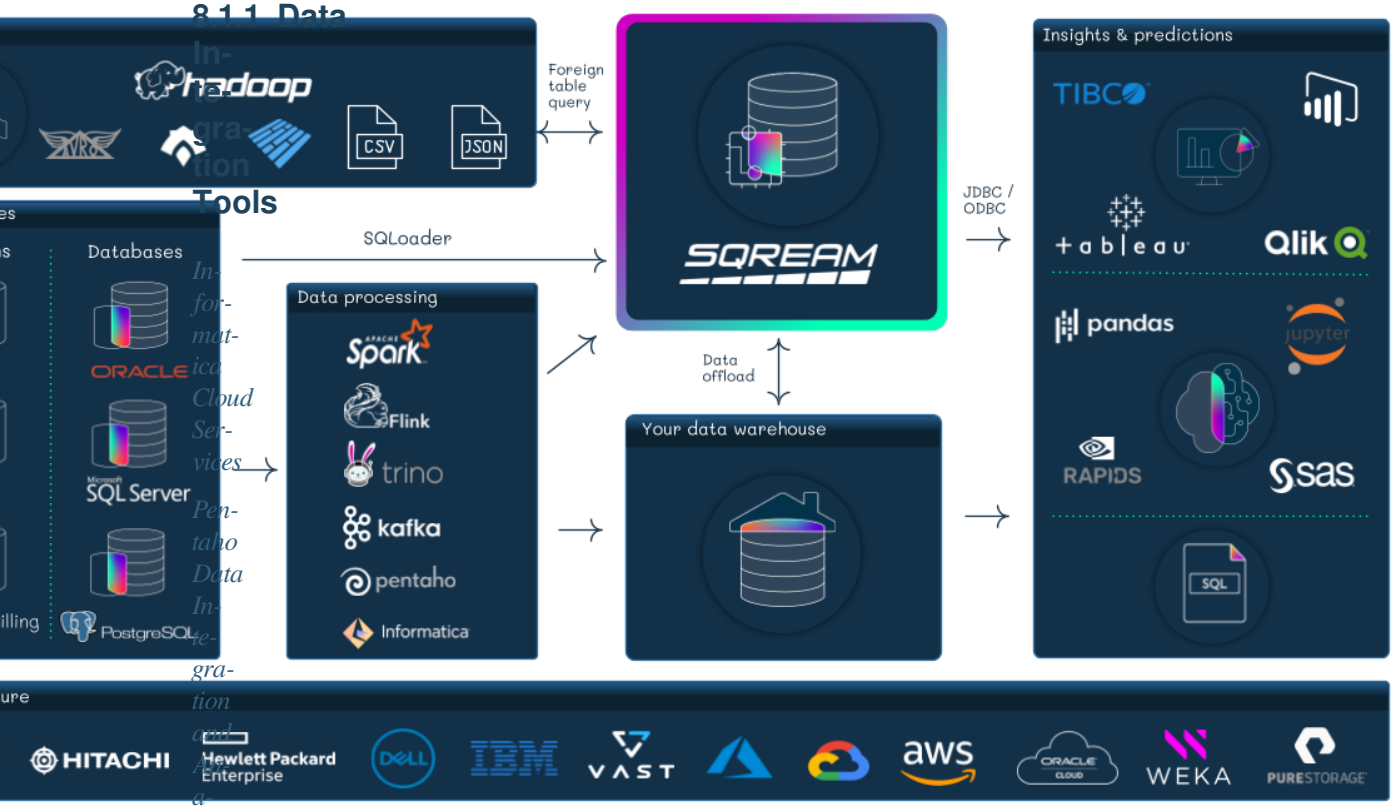
EIGHT

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

Semarchy

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 under 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 **Connection** tab and fill in the data source information:

Field name	Description	Value	Example
Name	The name of the data source	`sqream`	
Databas 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`	
Databas 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 verify your newly created connection, 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 Informatica*
- *Establishing a Connection In Your Environment*
  - *Establishing an ODBC DSN Connection In Your Environment*
  - *Establishing a JDBC Connection In Your Environment*
- *Supported SQream Driver Versions*

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 Identity 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 begins.
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 environment:

1. Click **Add**.
2. Click **Configure**.

   Note: Verify that **Use Server Picker** is selected.

3. Click **Test**.
4. Verify that the connection has tested successfully.
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 establish 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 successfully.

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 variety 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 information 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** dropdown menu, select **Generic**. The **Host Name**, **Port Number**, and **Database Name** fields are removed from the panel.
   2. In the **Version** dropdown menu, verify that **Generic DBMS** is selected.
   3. Click **Show Connection String**.
   4. Select the **Edit connection string** checkbox.
   5. From the **Driver** dropdown menu, select a driver for one of the following connectors:
      - **JDBC** - The SQream driver is not integrated with MicroStrategy and does not appear in the dropdown 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 **Connection String** text box, type the relevant connection string and path to the JDBC jar file using 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\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>;
      ↪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 displayed.
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 analytics.

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 Community 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 installing 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:
    a. On the CSV file input step item, click the **new hop connection** icon.
    b. Drag an arrow from the **CSV file input** step item to the **Table output** step item.
    c. Release the mouse button. The following options are displayed.
    d. Select **Main output of step**.
5. Double-click **Table output** to open the **Table output** dialog box.
6. In the **Target table** 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 PHP

8.1.3.5.1 Overview

PHP is an open source scripting language that executes scripts on servers. The **Connect to PHP** page explains how to connect to a SQream cluster, and describes the following:

- *Installing PHP*
- *Configuring PHP*
- *Operating PHP*

8.1.3.5.1.1 Installing PHP

To install PHP:

1. Download the JDBC driver installer from the SQream Drivers page.
2. Create a DSN.
3. Install the **uODBC** extension for your PHP installation.

   For more information, navigate to PHP Documentation and see the topic menu on the right side of the page.

8.1.3.5.1.2 Configuring PHP

You can configure PHP in one of the following ways:

- When compiling, configure PHP to enable uODBC using `./configure --with-pdo-odbc=unixODBC,/usr/local`.
- Install `php-odbc` and `php-pdo` along with PHP using your distribution package manager. SQream recommends a minimum of version 7.1 for the best results.

Note: PHP's string size limitations truncates fetched text, which you can override by doing one of the following:

- Increasing the **php.ini** default setting, such as the *odbc.defaultlrl* to **10000**.
- Setting the size limitation in your code before making your connection using **ini_set("odbc.defaultlrl", "10000");**.
- Setting the size limitation in your code before fetchng your result using **odbc_longreadlen($result, "10000");**.

8.1.3.5.1.3 Operating PHP

After configuring PHP, you can test your connection.

To test your connection:

1. Create a test connection file using the correct parameters for your SQream installation, as shown below:

```
<?php // Construct a DSN connection string
$dsn  = "SqreamODBC"; // Create a connection
$conn = odbc_connect($dsn, '', '');
if (!($conn)) {
    echo "Connection to SQream DB via ODBC failed: " . odbc_errormsg(
↪$conn);
}
$sql = "SELECT show_version()"; // Execute the query
$rs  = odbc_exec($conn, $sql);
while (odbc_fetch_row($rs)) {
    for ($i = 1; $i <= odbc_num_fields($rs); $i++) {
        echo "Result is " . odbc_result($rs, $i);
    }
}
echo "\n";
odbc_close($conn); // Finally, close the connection
?>
```

For more information, download the sample `PHP example connection file` shown above.

The following is an example of a valid DSN line:

```
$dsn = "odbc:Driver={SqreamODBCDriver};Server=192.168.0.5;Port=5000;
↪Database=master;User=rhendricks;Password=super_secret;Service=sqream";
```

2. Run the PHP file either directly with PHP (`php test.php`) or through a browser.

   For more information about supported DSN parameters, see *ODBC DSN Parameters*.

8.1.3.6 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*

8.1.3.6.1 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.

8.1.3.6.2 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. Navigate to **File > Options and Settings > Option > Security > Data Extensions**, and select **(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 **DirectQuery mode**.

12. Click **Connect**.

13. Provide your user name and password and click **Connect**.

8.1.3.6.3 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.

8.1.3.7 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*

8.1.3.7.1 JDBC

1. Get the *SQream DB JDBC driver*.
2. In R, install RJDBC

```
> install.packages("RJDBC")
Installing package into 'C:/Users/r/...'
(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 Windows:

```
> 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\\2020.1-3.2.0\\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")
   xint  xtinyint xsmallint xbigint
1    1       82      5067       1
2    2       14      1756       2
3    3       91     22356       3
4    4       84     17232       4
5    5       13     14315       5
```

6. Close the connection

```
> close(con)
```

8.1.3.7.1.1 A full example

```
> library(RJDBC)
> 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\\2020.1-3.2.0\\sqream-jdbc-3.2.jar")
> 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")
   xint  xtinyint xsmallint xbigint
1    1       82      5067       1
2    2       14      1756       2
3    3       91     22356       3
4    4       84     17232       4
5    5       13     14315       5
> close(con)
```

8.1.3.7.2 ODBC

1. Install the *SQream DB 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

```
> 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")
   xint  xtinyint xsmallint xbigint
1    1        82      5067       1
2    2        14      1756       2
3    3        91     22356       3
4    4        84     17232       4
5    5        13     14315       5
```

6. Close the connection

```
> close(ch)
```

8.1.3.7.2.1 A full example

```
> library(RODBC)
> ch <- odbcConnect("my_cool_dsn",believeNRows=F)
> sqlQuery(ch,"select top 5 * from t")
   xint  xtinyint xsmallint xbigint
1    1        82      5067       1
2    2        14      1756       2
3    3        91     22356       3
4    4        84     17232       4
5    5        13     14315       5
> close(ch)
```

8.1.3.8 SAP BusinessObjects

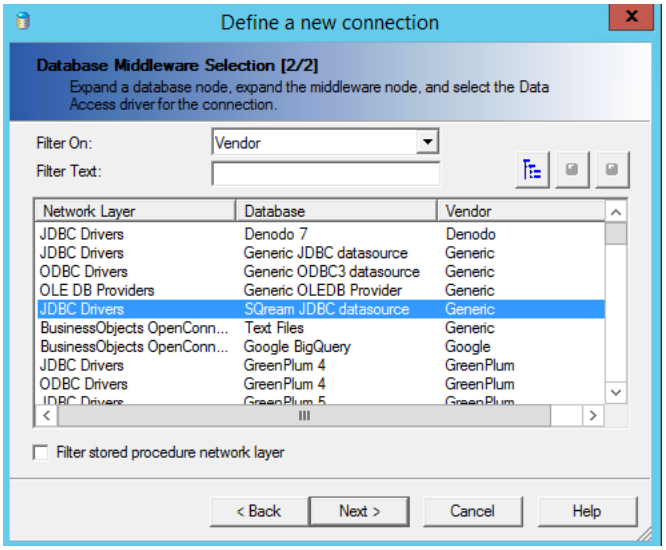
The **Connecting to SQream Using SAP BusinessObjects** guide includes the following sections:

- *Overview*
- *Establishing a New Connection Using a Generic JDCB Connector*

8.1.3.8.1 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.

8.1.3.8.2 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 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.jdbc.SQDriver

   </JDBCDriver>
   </DataBase>
   ```

3. Restart the BusinessObjects server.

   When the connection is established, **SQream** is listed as a driver selection.

8.1.3.9 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*

8.1.3.9.1 Installing SAS Viya

The **Installing SAS Viya** section describes the following:

8.1.3.9.1.1 Downloading SAS Viya

Integrating with SQreamDB has been tested with SAS Viya v.03.05 and newer.

To download SAS Viya, see SAS Viya.

8.1.3.9.1.2 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.

8.1.3.9.2 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_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"
   schema="public"
   PRESERVE_TAB_NAMES=YES
   PRESERVE_COL_NAMES=YES;
```

8.1.3.9.3 Operating SAS Viya

The **Operating SAS Viya** section describes the following:

8.1.3.9.3.1 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 credentials:
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 information 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**.

8.1.3.9.4 Troubleshooting SAS Viya

The **Best Practices and Troubleshooting** section describes the following best practices and troubleshooting procedures when connecting to SQreamDB using SAS Viya:

8.1.3.9.4.1 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.

8.1.3.9.4.2 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.

8.1.3.9.4.3 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 access.
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.

8.1.3.9.4.4 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.

8.1.3.10 Semarchy

Semarchy's Intelligent Data eXchange (IDX) facilitates seamless data integration and interoperability across systems. IDX ensures reliable data exchange between different applications, enhancing overall data quality, governance, and adaptability for critical business operations.

8.1.3.10.1 Before You Begin

It is essential that you use Semarchy version 2023.01 or later.

8.1.3.10.2 Setting Up a Connection to SQreamDB

1. Install the Semarchy SQreamDB component as described in Semarchy documentation.
2. Install SQreamDB *JDBC*.

8.1.3.10.3 JDBC Connection String

The following is a SQreamDB JDBC connection string template:

```
jdbc:Sqream://<host and port>/<database name>;user=<username>;password=<password>;[
↪<optional parameters>; ...]
```

8.1.3.10.3.1 Connection Parameters

Item	State	De-fault	Description
`<host and port>`	Manc tory	None	Hostname and port of the SQream DB worker. For example, `127.0.0.1:5000`, `sqream.mynetwork.co:3108`
`<databas name>`	Manc tory	None	Database name to connect to. For example, `master`
`user-name=<us`	Op-tional	None	Username of a role to use for connection. For example, `username=SqreamRole`
`pass-word=<pa`	Op-tional	None	Specifies the password of the selected role. For example, `pass-word=SqreamRolePassword2023`
`ser-vice=<se`	Op-tional	`sqre`	Specifices service queue to use. For example, `service=etl`
`<ssl>`	Op-tional	`fals`	Specifies SSL for this connection. For example, `ssl=true`
`<cluster`	Op-tional	`true`	Connect via load balancer (use only if exists, and check port).
`<fetchSi`	Op-tional	`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.
`<insertB`	Op-tional	`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.
`<loggerL`	Op-tional	`true`	Defines the logger level as either `debug` or `trace`.
`<logFile`	Op-tional	`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.
`<idlecor`	Op-tional	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.

8.1.3.11 SQL Workbench

You can use SQL Workbench to interact with a SQream DB cluster. SQL Workbench/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 Manually*
  - *Install Java Runtime*
  - *Get the SQream DB JDBC Driver*
  - *Install SQL Workbench*

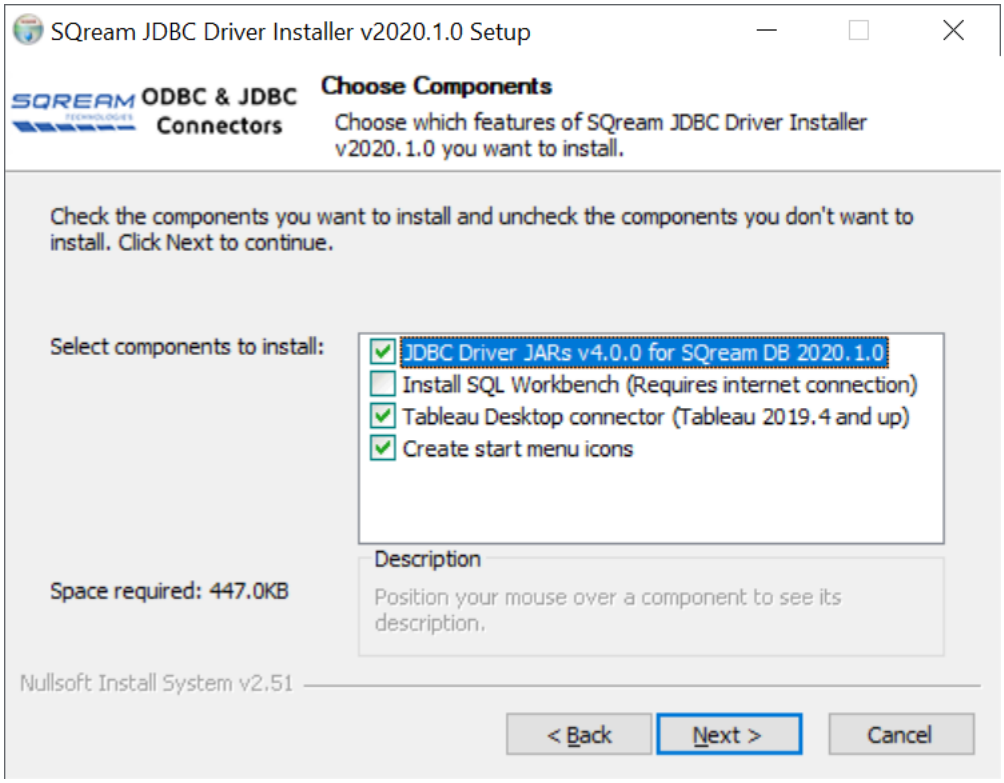
  - *Setting up the SQream DB JDBC Driver Profile*
- *Create a New Connection Profile for Your Cluster*
- *Suggested Optional Configuration*

8.1.3.11.1 Installing SQL Workbench with the SQream Installer

This section applies to Windows only.

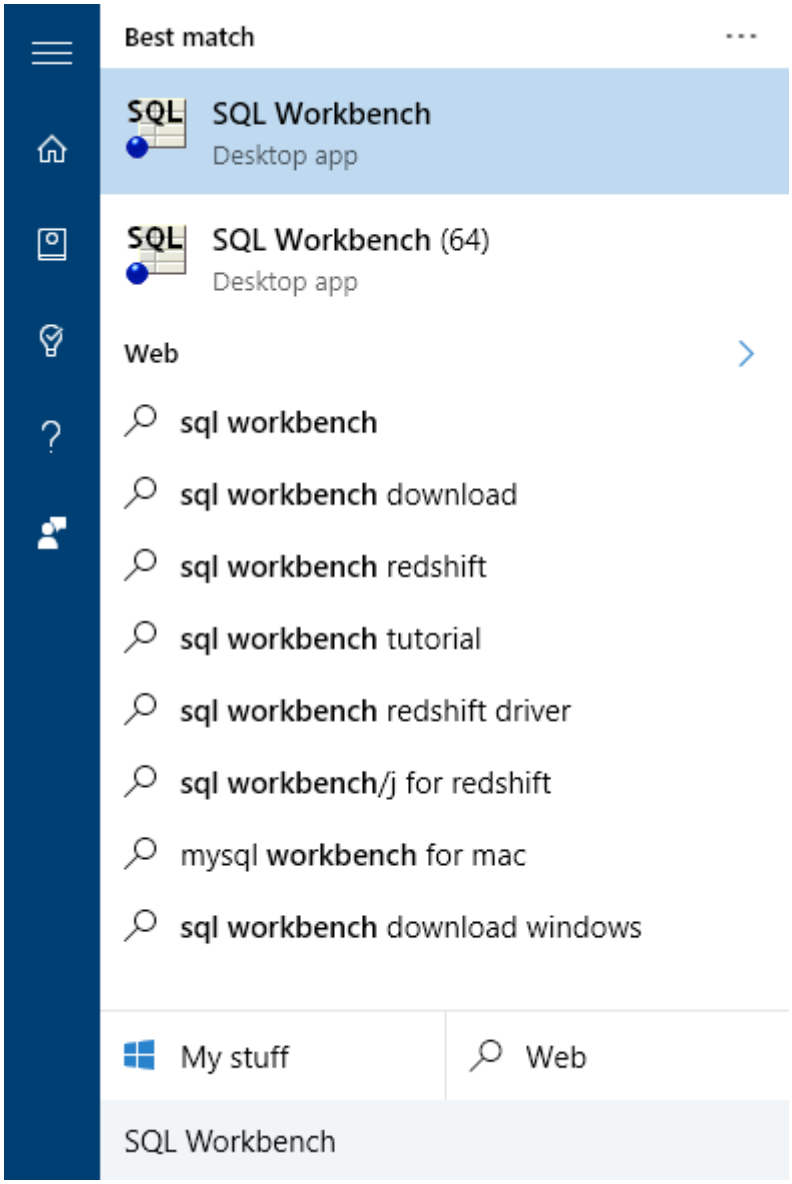
SQream DB's driver installer for Windows can install the Java prerequisites and SQL Workbench for you.

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 connecting to SQream DB clusters.
2. Start SQL Workbench from the Windows start menu. Be sure to select **SQL Workbench (64)** if you're on 64-bit Windows.



You are now ready to create a profile for your cluster. Continue to *Creating a new connection profile*.

8.1.3.11.2 Installing SQL Workbench Manually

This section applies to Linux and MacOS only.

8.1.3.11.2.1 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

8.1.3.11.2.2 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.

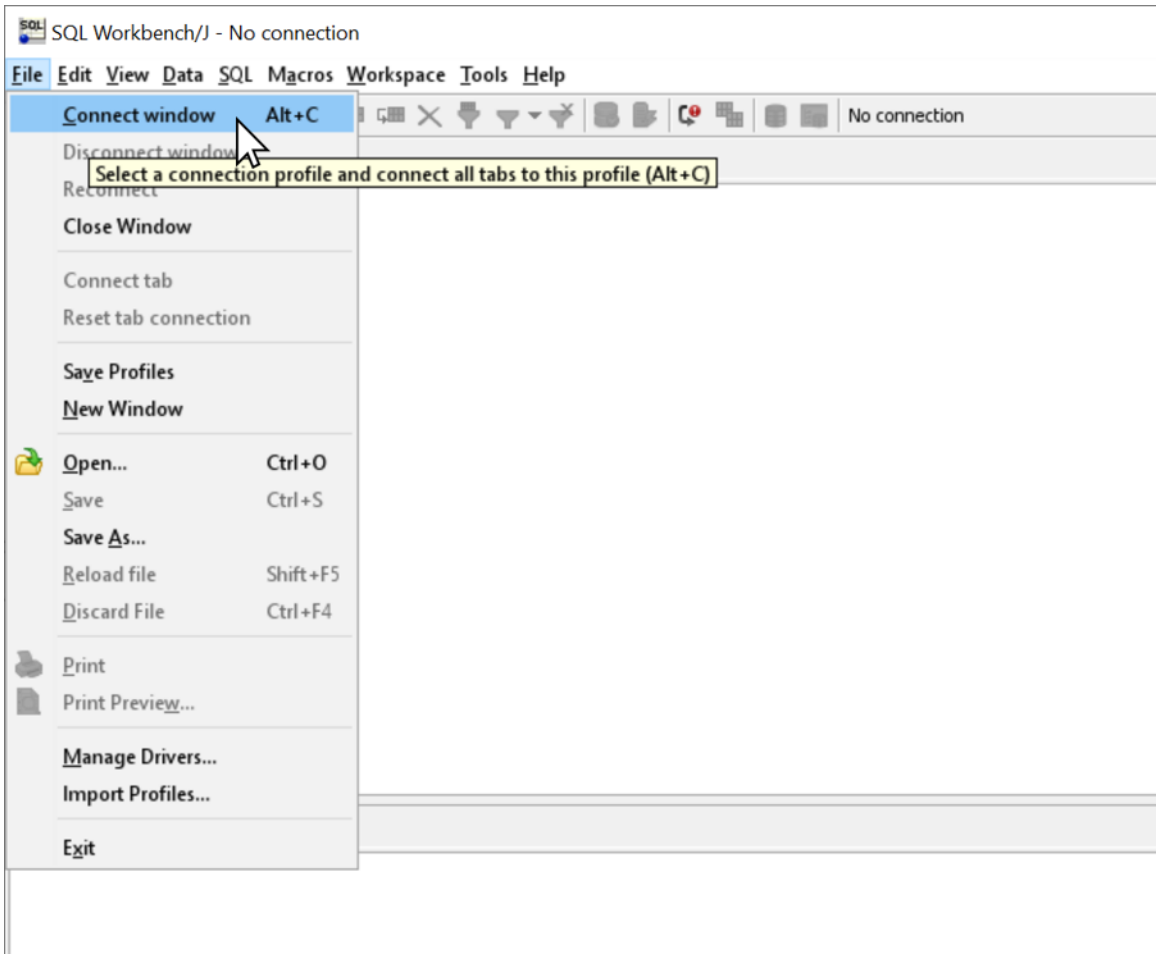
Download and extract the JAR file from the zip archive.

8.1.3.11.2.3 Install SQL Workbench

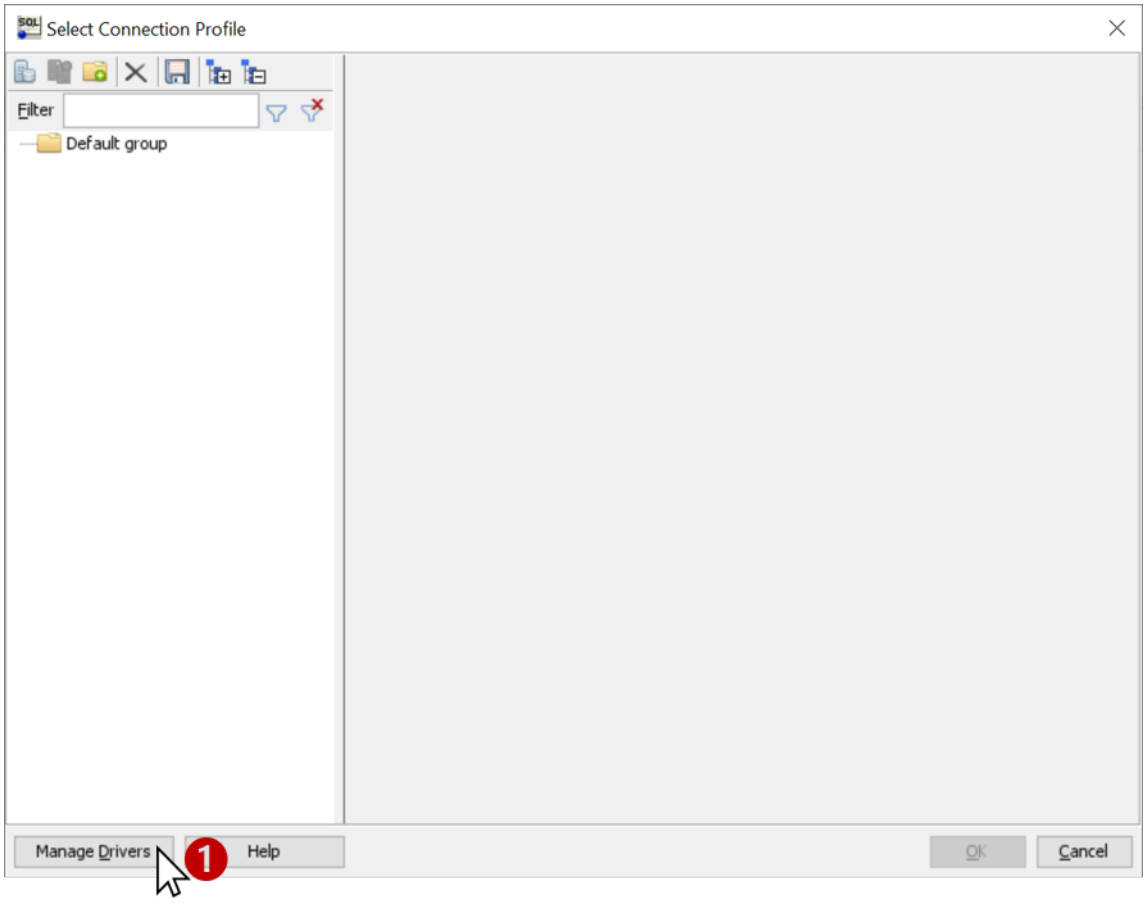
1. Download the latest stable release from https://www.sql-workbench.eu/downloads.html . The **Generic package for all systems** is recommended.
2. Extract the downloaded ZIP archive into a directory of your choice.
3. Start SQL workbench. If you are using 64 bit windows, run `SQLWorkbench64.exe` instead of `SQLWOrkbench.exe`.

8.1.3.11.2.4 Setting up the SQream DB JDBC Driver Profile

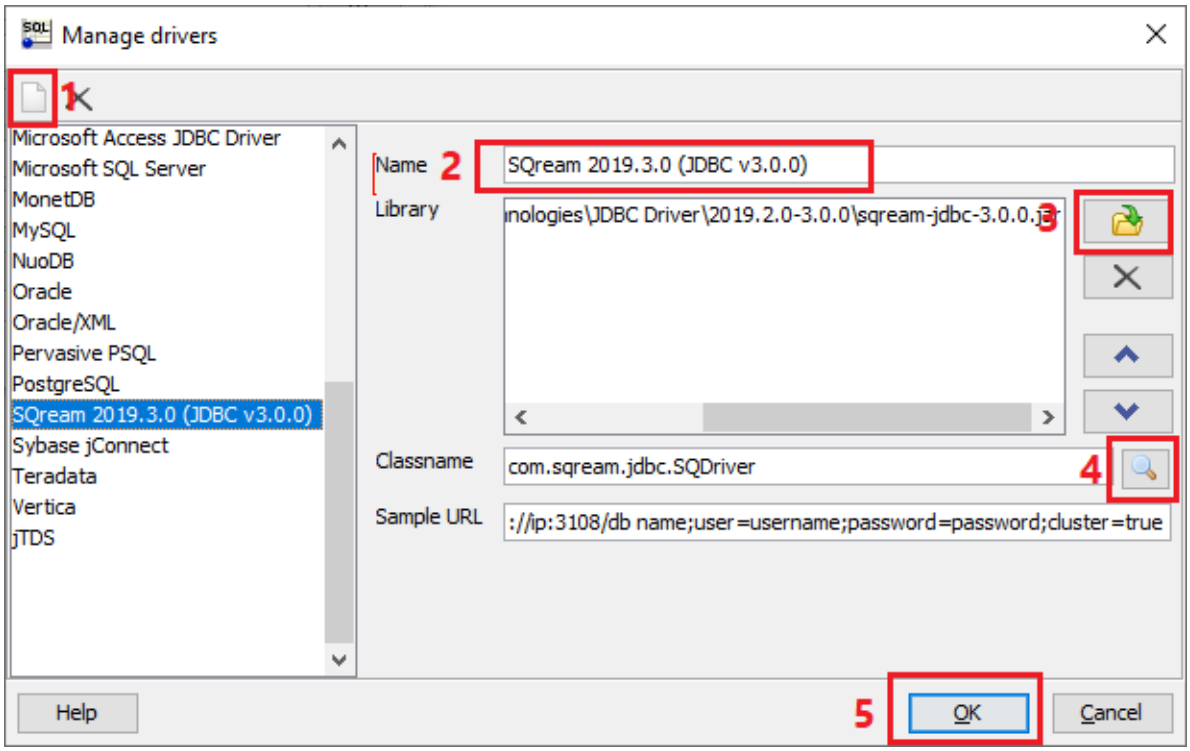
1. Define a connection profile - *File ▸ Connect window (Alt+C)*



2. Open the drivers management window - *Manage Drivers*



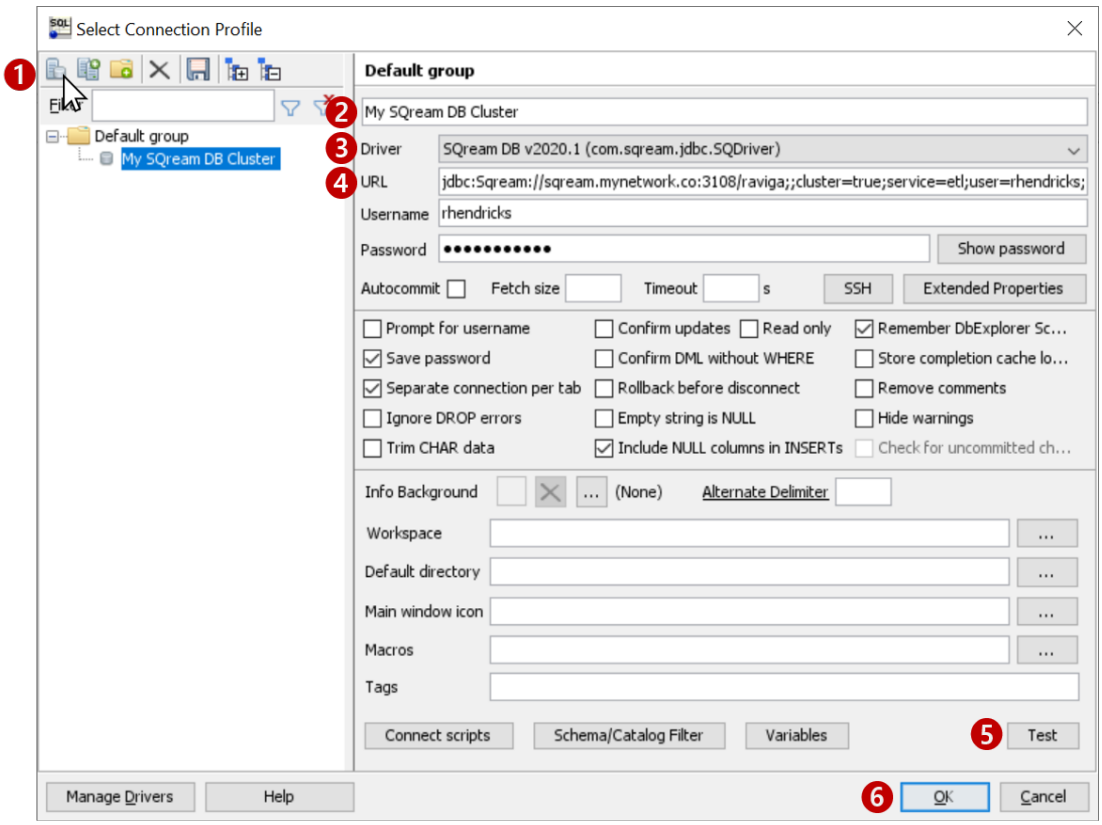
3. Create the SQream DB driver profile



1. Click on the Add new driver button (“New” icon)
2. Name the driver as you see fit. We recommend calling it SQream DB <version>, 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 magnifying glass button to detect the classname automatically. Other details are purely optional
5. Click OK to save and return to “new connection screen”

8.1.3.11.3 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.
5. Text the connection details
6. Click OK to save the connection profile and connect to SQream DB

8.1.3.11.4 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

8.1.3.12 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*

8.1.3.12.1 Prerequisites

It is essential that you have the following installed:

- Tableau version 9.2 or newer

8.1.3.12.2 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 Windows: `C:\Program Files\Tableau\Drivers`
   - Tableau on Linux: `/opt/tableau/tableau_driver/jdbc`

8.1.3.12.3 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.

8.1.3.12.4 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 connection 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.

8.1.3.13 Talend

8.1.3.13.1 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*

8.1.3.13.1.1 Creating a New Metadata JDBC DB Connection

To create a new metadata JDBC DB connection:

1. In the **Repository** panel, nagivate to **Metadata** and right-click **Db connections**.
2. Select **Create connection**.
3. In the **Name** field, type a name.

   Note that the name cannot 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 **"newLine"** entry is added.
8. One the **"newLine'** entry, click the ellipsis.

   The **Module** window is displayed.
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.

8.1.3.13.1.2 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.

8.1.3.13.1.3 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 information on the complete list of connectors and datasets that Talend supports, see Introducing Talend Connectors.

8.1.3.13.1.4 Known Issues

As of 6/1/2021 schemas were not displayed for tables with identical names.

8.1.3.14 TIBCO Spotfire

8.1.3.14.1 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*

8.1.3.14.1.1 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*

8.1.3.14.1.2 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.

8.1.3.14.1.3 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 Connection** 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.

8.1.3.14.1.4 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>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>
      <from>Bool</from>
      <to>Integer</to>
    </type-mapping>
    <type-mapping>
      <from>TEXT(2048)</from>
      <to>String</to>
    </type-mapping>
    <type-mapping>
      <from>INT</from>
      <to>Integer</to>
    </type-mapping>
    <type-mapping>
      <from>BIGINT</from>
      <to>LongInteger</to>
    </type-mapping>
    <type-mapping>
      <from>Real</from>
      <to>Real</to>
```

(continues on next page)

(continued from previous page)

```
    </type-mapping>
         <type-mapping>
      <from>Decimal</from>
      <to>Float</to>
    </type-mapping>
     <type-mapping>
      <from>Numeric</from>
      <to>Float</to>
    </type-mapping>
    <type-mapping>
      <from>Date</from>
      <to>DATE</to>
    </type-mapping>
    <type-mapping>
      <from>DateTime</from>
      <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.

8.1.3.14.1.5 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.

8.1.3.14.1.6 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, select **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 filter 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 filtering 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.

8.1.3.14.1.7 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*

8.1.3.14.1.8 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 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*.

8.1.3.14.1.9 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
SQream DB Java CLI	SQream DB Java Command Line Interface	sqream_sql_cl	Replaces the Deprecated Haskell Command Line Tool	All
Apache Spark	Apache Spark Connector	*Spark*		All
Dataiku	Plugin Git repository: `git@github.com:SQream/dataiku_plugin.git`	*Dataiku*		All
JDBC	sqream-jdbc	*JDBC*	Recommended installation via `mvn`	All
Node.JS	sqream-v4.2.4	*Node.JS*	Recommended installation via `npm`	All
ODBC	Windows ODBC Installer , Linux ODBC	*Windows*, *Linux*		Windows, Linux
Power BI	Power BI Power Query Connector	*BI Desktop*		All
Python	pysqream	*Python (pysqream)*	Recommended installation via `pip`	All
Python-SQLAlchem	pysqream-sqlalchemy	*pysqream*	Recommended installation via `pip`	All
SQream-Net	.NET .dll file	*SQreamNET*		All
Tableau	Tableau Connector	*Tableau*		All
Trino	Trino Connector	*Trino*		All

8.2.1.1 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*

8.2.1.1.1 Before You Begin

- The SQreamNET provider requires a .NET version 6 or newer
- Download the SQreamNET driver from the *client drivers page*

8.2.1.1.2 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 upgrade SQreamNET within an existing project, replace your existing .dll file with an updated one or change the project's reference location to a new one.

8.2.1.1.3 Connecting to SQream For the First Time

An initial connection to SQream must be established by creating a **SqreamConnection** object using a connection string.

8.2.1.1.3.1 Connection String Syntax

```
Data Source=<hostname or ip>,<port>;User=<username>;Password=<password>;Initial␣
↪Catalog=<database name>;Integrated Security=true;
```

8.2.1.1.3.2 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	`sqrea`	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).

8.2.1.1.3.3 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 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;
```

8.2.1.1.3.4 Sample C# Program

You can download the `.NET Application Sample File` below by right-clicking and saving it to your computer.

Listing 1: .NET Application Sample

```
1        public void Test()
2        {
3            var connection = OpenConnection("192.168.4.62", 5000, "sqream", "sqream",
↪"master");
4
5            ExecuteSQLCommand(connection, "create or replace table tbl_example as␣
↪select 1 as x , 'a' as y;");
6
7            var tableData = ReadExampleData(connection, "select * from tbl_example;");
8        }
9
10       /// <summary>
11       /// Builds a connection string to sqream server and opens a connection
12       /// </summary>
13       /// <param name="ipAddress">host to connect</param>
14       /// <param name="port">port sqreamd is running on</param>
15       /// <param name="username">role username</param>
16       /// <param name="password">role password</param>
17       /// <param name="databaseName">database name</param>
18       /// <param name="isCluster">optional - set to true when the ip,port endpoint␣
↪is a server picker process</param>
19       /// <returns>
20       /// SQream connection object
21       /// Throws SqreamException if fails to open a connction
22       /// </returns>
23       public SqreamConnection OpenConnection(string ipAddress, int port, string␣
↪username, string password, string databaseName, bool isCluster = false)
24       {
25           // create the connection string according to the format
26           var connectionString = string.Format(
27               "Data Source={0},{1};User={2};Password={3};Initial Catalog={4};
```

(continues on next page)

(continued from previous page)

```
↪Cluster={5}",
                ipAddress,
                port,
                username,
                password,
                databaseName,
                isCluster
                );

            // create a sqeram connection object
            var connection = new SqreamConnection(connectionString);

            // open a connection
            connection.Open();

            // returns the connection object
            return connection;
        }

        /// <summary>
        /// Executes a SQL command to sqream server
        /// </summary>
        /// <param name="connection">connection to sqream server</param>
        /// <param name="sql">sql command</param>
        /// <exception cref="InvalidOperationException"> thrown when the connection␣
↪is not open</exception>
        public void ExecuteSQLCommand(SqreamConnection connection, string sql)
        {
            // validates the connection 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));

            // creates a new command object utilizing the sql and the connection
            var command = new SqreamCommand(sql, connection);

            // executes the command
            command.ExecuteNonQuery();
        }

        /// <summary>
        /// Executes a SQL command to sqream server, and reads the result set usiing␣
↪DataReader
        /// </summary>
        /// <param name="connection">connection to sqream server</param>
        /// <param name="sql">sql command</param>
        /// <exception cref="InvalidOperationException"> thrown when the connection␣
↪is not open</exception>
        public List<Tuple<int, string>> ReadExampleData(SqreamConnection connection,␣
↪string sql)
        {
            // validates the connection 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));

            // creates a new command object utilizing the sql and the connection
```

(continues on next page)

(continued from previous page)

```
            var command = new SqreamCommand(sql, connection);

            // creates a reader object to iterate over the result set
            var reader = (SqreamDataReader)command.ExecuteReader();

            // list of results
            var result = new List<Tuple<int, string>>();

            //iterate the reader and read the table int,string values into a result tuple object
            while (reader.Read())
                result.Add(new Tuple<int, string>(reader.GetInt32(0), reader.GetString(1)));

            // return the result set
            return result;
        }
```

8.2.1.1.4 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.

8.2.1.2 Dataiku

This Plugin accelerates data transfer from Amazon S3 to SqreamDB within Dataiku DSS. It enables direct loading of data from S3 to SqreamDB, ensuring rapid transfers without external steps.

The Plugin includes a code environment that automatically installs the SqreamDB Python Connector (pysqream) alongside the Plugin.

The following file formats are supported:

- Avro
- JSON
- CSV (requires manual data type mapping as the default for all columns is `TEXT`)

8.2.1.2.1 Before You Begin

It is essential you have the follwoing:

- Sqreamdb *JDBC* connection set up in DSS
- Amazon S3 connection set up in DSS
- Python 3.9

8.2.1.2.2 Establishing a Dataiku Connection

In your Dataiku web interface:

1. Upload the plugin from the following SQreamDB Git repository:

```
-- Repository URL:
git@github.com:SQream/dataiku_plugin.git

-- Path in repository:
s3_bulk_load
```

2. Define a DSS S3 dataset.
3. Add the plugin to your flow.
4. Set the S3 Dataset as Input of the Plugin (mandatory).
5. Assign a name for the output dataset stored in your SQreamDB connection.
6. Provide AWS Access Key and Secret Key by either:

a. Filling in the values in the Plugin form

b. Set the Project Variables or set the Global Variables when DSS Variables are used

8.2.1.3 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*

8.2.1.3.1 Installing the JDBC Driver

The **Installing the JDBC Driver** section describes the following:

- *Prerequisites*
- *Getting the JAR file*
- *Setting Up the Class Path*

8.2.1.3.1.1 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

8.2.1.3.1.2 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.

8.2.1.3.1.3 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
```

8.2.1.3.2 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*

8.2.1.3.2.1 Driver Class

Use `com.sqream.jdbc.SQDriver` as the driver class in the JDBC application.

8.2.1.3.2.2 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>; ...]
```

8.2.1.3.2.3 Connection Parameters

The following table shows the connection string parameters:

Item	State	De-fault	Description
`<host and port>`	Manc tory	None	Hostname and port of the SQream DB worker. For example, `127.0.0.1:5000`, `sqream.mynetwork.co:3108`
`<databas name>`	Manc tory	None	Database name to connect to. For example, `master`
`user-name=<us`	Op-tional	None	Username of a role to use for connection. For example, `username=SqreamRole`
`pass-word=<pa`	Op-tional	None	Specifies the password of the selected role. For example, `pass-word=SqreamRolePassword2023`
`ser-vice=<se`	Op-tional	`sqre`	Specifices service queue to use. For example, `service=etl`
`<ssl>`	Op-tional	`fals`	Specifies SSL for this connection. For example, `ssl=true`
`<cluster`	Op-tional	`true`	Connect via load balancer (use only if exists, and check port).
`<fetchSi`	Op-tional	`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.
`<insertB`	Op-tional	`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.
`<loggerL`	Op-tional	`true`	Defines the logger level as either `debug` or `trace`.
`<logFile`	Op-tional	`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.
`<idlecor`	Op-tional	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.

8.2.1.3.2.4 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;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
```

8.2.1.3.2.5 Java Program Sample

You can download the `JDBC Application Sample File` below by right-clicking and saving it to your computer.

Listing 2: JDBC Application Sample

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.Statement;
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
    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;
    Statement stmt = null;
    ResultSet rs = null;
    DatabaseMetaData dbmeta = null;

    int res = 0;

    public void testJDBC() throws SQLException, IOException {

        // Create a connection
        conn = DriverManager.getConnection(url,"rhendricks","Tr0ub4dor&3");

```

(continues on next page)

(continued from previous page)

```
        // Create a table with a single integer column
        String sql  =  "CREATE TABLE test (x INT)";
        stmt = conn.createStatement();  // Prepare the statement
        stmt.execute(sql); // Execute the statement
        stmt.close(); // Close the statement handle

        // Insert some values into the newly created table
        sql = "INSERT INTO test VALUES (5),(6)";
        stmt = conn.createStatement();
        stmt.execute(sql);
        stmt.close();

        // Get values from the table
        sql = "SELECT * FROM test";
        stmt = conn.createStatement();
        rs  = stmt.executeQuery(sql);
        // Fetch all results one-by-one
        while(rs.next()) {
            res = rs.getInt(1);
            System.out.println(res); // Print results to screen
        }
        rs.close(); // Close the result set
        stmt.close(); // Close the statement handle
        conn.close();
    }


    public  static  void  main(String[]  args)  throws  SQLException, ␣
↪KeyManagementException,  NoSuchAlgorithmException,  IOException, ␣
↪ClassNotFoundException{

        // Load SQream DB JDBC driver
        Class.forName("com.sqream.jdbc.SQDriver");

        // Create test object and run
        SampleTest  test  =  new  SampleTest();
        test.testJDBC();
    }
}
```

8.2.1.3.3 Prepared Statements

Prepared statements, also known as parameterized queries, are a safer and more efficient way to execute SQL statements. They prevent SQL injection attacks by separating SQL code from data, and they can improve performance by reusing prepared statements. In SQream, we use `?` as a placeholder for the relevant value in parameterized queries. Prepared statements `INSERT`, `SELECT`, `UPDATE` and `DELETE`

8.2.1.3.3.1 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.

8.2.1.3.3.2 Prepared Statement Limitations

- Prepared Statement do not support the use of keywords_and_identifiers as input parameters.
- `SELECT`, `UPDATE` and `DELETE` statements require the use of `add_batch` prior to each execution.

8.2.1.4 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 configuration*
  - *Preventing `heap out of memory` errors*
  - *BIGINT support*

8.2.1.4.1 Installing the Node.JS driver

8.2.1.4.1.1 Prerequisites

- Node.JS 10 or newer. Follow instructions at nodejs.org .

8.2.1.4.1.2 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
```

8.2.1.4.1.3 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
```

8.2.1.4.2 Connect to SQream DB with a Node.JS application

8.2.1.4.2.1 Create a simple test

Replace the connection parameters with real parameters for a SQream DB installation.

Listing 3: sqreamdb-test.js

```
const Connection = require('@sqream/sqreamdb');

const config  =  {
  host: 'localhost',
  port: 3109,
  username: 'rhendricks',
  password: 'super_secret_password',
  connectDatabase: 'raviga',
  cluster: true,
  is_ssl: true,
  service: 'sqream'
  };

const query1  =  'SELECT 1 AS test, 2*6 AS "dozen"';

const sqream = new Connection(config);
sqream.execute(query1).then((data) => {
   console.log(data);
}, (err) => {
   console.error(err);
});
```

8.2.1.4.2.2 Run the test

A successful run should look like this:

```
$ node sqreamdb-test.js
[  { test: 1, dozen: 12  }  ]
```

8.2.1.4.3 API reference

8.2.1.4.3.1 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`

8.2.1.4.3.2 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.

8.2.1.4.3.3 Example

```
const myConnection  = new Connection(config);

myConnection.runQuery(query1, function  (err, data){
  myConnection.events.on('getConnectionId', function(data){
      console.log('getConnectionId', data);
  });

  myConnection.events.on('getStatementId', function(data){
```

(continues on next page)

(continued from previous page)

```
      console.log('getStatementId', data);
  });

  myConnection.events.on('getTypes', function(data){
      console.log('getTypes', data);
  });
});
```

8.2.1.4.3.4 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.

8.2.1.4.4 Examples

8.2.1.4.4.1 Setting configuration flags

SQream DB 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 Connection(config);
myConnection.runQuery(query_string, function  (err, data){
  console.log(err, data);
}, setFlag);
```

8.2.1.4.4.2 Lazyloading

To process rows without keeping them in memory, you can lazyload the rows with an async:

```
const Connection = require('@sqream/sqreamdb');

const config  =  {
  host: 'localhost',
  port: 3109,
  username: 'rhendricks',
  password: 'super_secret_password',
  connectDatabase: 'raviga',
  cluster: true,
  is_ssl: true,
  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;
  for await (let rows of cursor.fetchIterator(100)) {
    // fetch rows in chunks of 100
    count += rows.length;
  }
  await cursor.close();
  return count;
})().then((total) => {
  console.log('Total rows', total);
}, (err) => {
  console.error(err);
});
```

8.2.1.4.4.3 Reusing a connection

It is possible to execeute 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',
  cluster: true,
  is_ssl: true,
  service: 'sqream'
  };

const sqream = new Connection(config);

(async () => {
```

(continues on next page)

(continued from previous page)

```
  const conn = await sqream.connect();
  try {
    const res1 = 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};
  } catch (err) {
    conn.disconnect();
    throw err;
  }

})().then((res) => {
  console.log('Results', res)
}, (err) => {
  console.error(err);
});
```

8.2.1.4.4.4 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: 'localhost',
  port: 3109,
  username: 'rhendricks',
  password: 'super_secret_password',
  connectDatabase: 'raviga',
  cluster: true,
  is_ssl: true,
  service: 'sqream'
  };

const sqream = 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`

8.2.1.4.5 Troubleshooting and recommended configuration

8.2.1.4.5.1 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:

```
$ node --max-old-space-size=2048 my-application.js
```

8.2.1.4.5.2 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'
      ? value.toString()
      : value // return everything else unchanged
));
console.log(json); // [{"test": "1"}]
```

8.2.1.5 ODBC

8.2.1.5.1 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*

8.2.1.5.1.1 Installing the ODBC Driver

8.2.1.5.1.2 Prerequisites

8.2.1.5.1.3 Visual Studio 2015 Redistributables

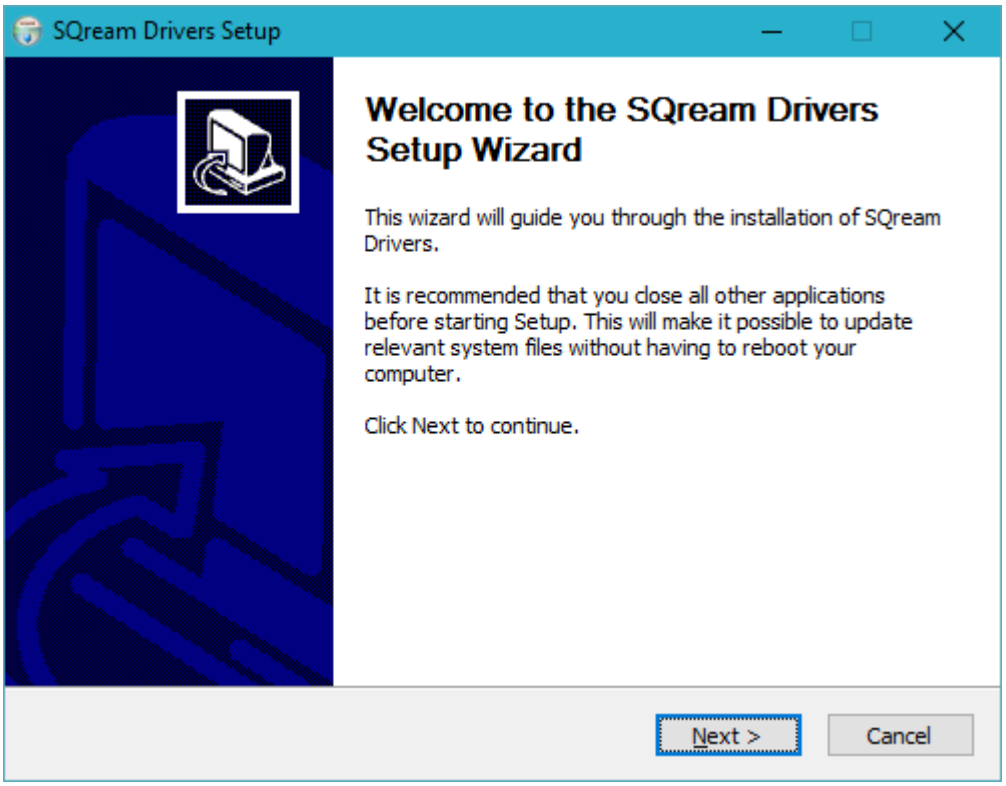
To install the ODBC driver you must first install Microsoft's **Visual C++ Redistributable for Visual Studio 2015**. To install Visual C++ Redistributable for Visual Studio 2015, see the Install Instructions.

8.2.1.5.1.4 Administrator Privileges

The SQream DB ODBC driver requires administrator privileges on your computer to add the DSNs (data source names).

8.2.1.5.1.5 Running the Windows Installer

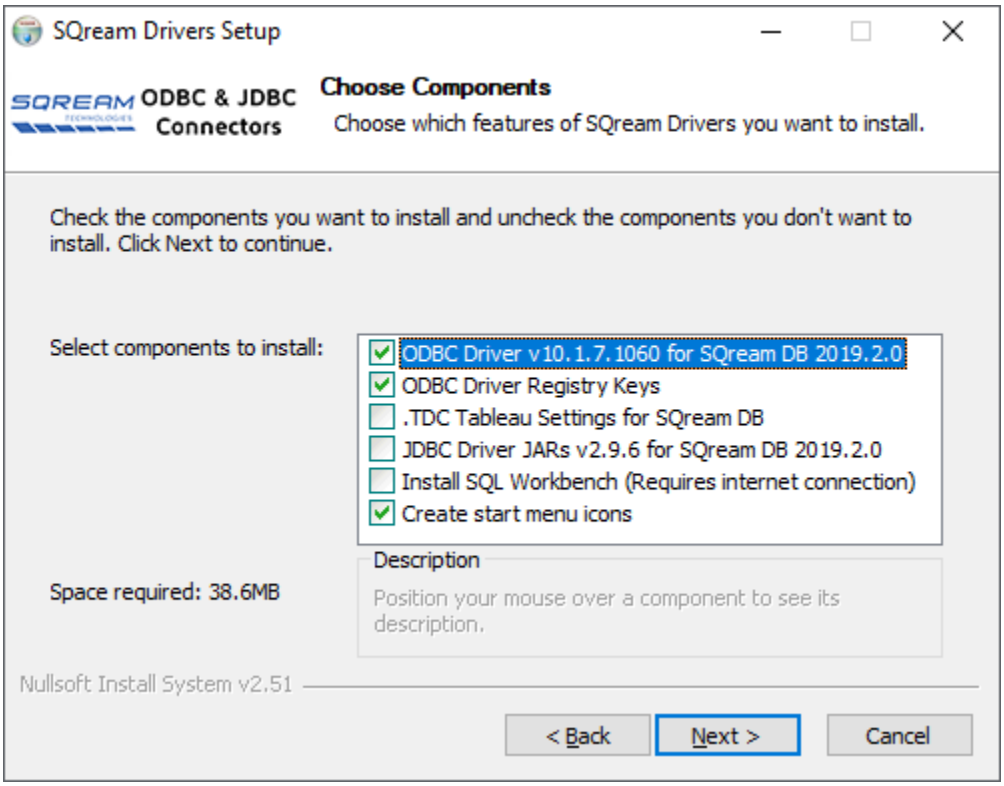
Install the driver by following the on-screen instructions in the easy-to-follow installer.



Note: The installer will install the driver in `C:\Program Files\SQream Technologies\ODBC Driver` by default. This path is changable during the installation.

8.2.1.5.1.6 Selecting Components

The installer includes additional components, like JDBC and Tableau customizations.



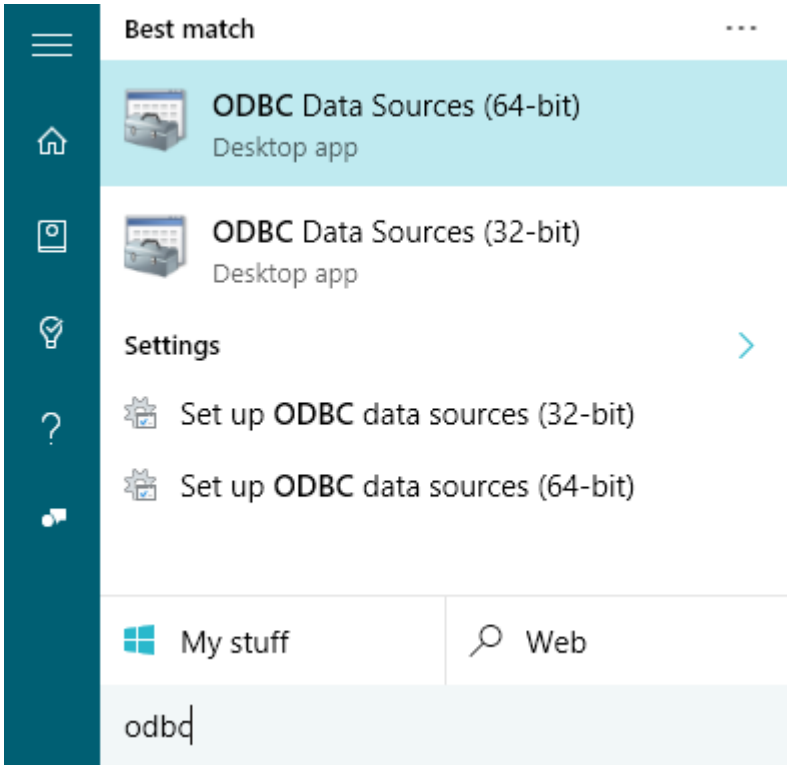
You can deselect items you don't want to install, but the items named **ODBC Driver DLL** and **ODBC Driver Registry Keys** must remain selected for a complete installation of the ODBC driver.

Once the installer finishes, you will be ready to configure the DSN for connection.

8.2.1.5.1.7 Configuring the ODBC Driver DSN

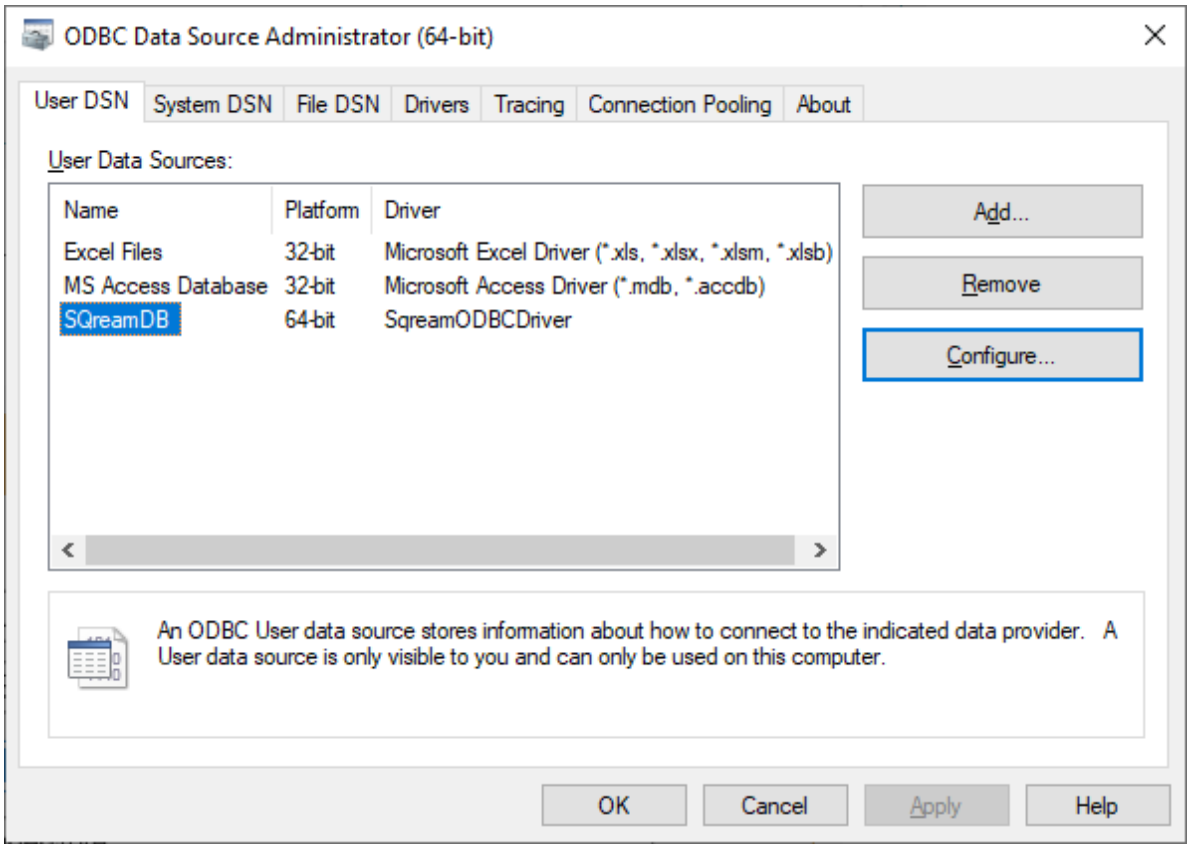
ODBC driver configurations are done via DSNs. Each DSN represents one SQream DB database.

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.

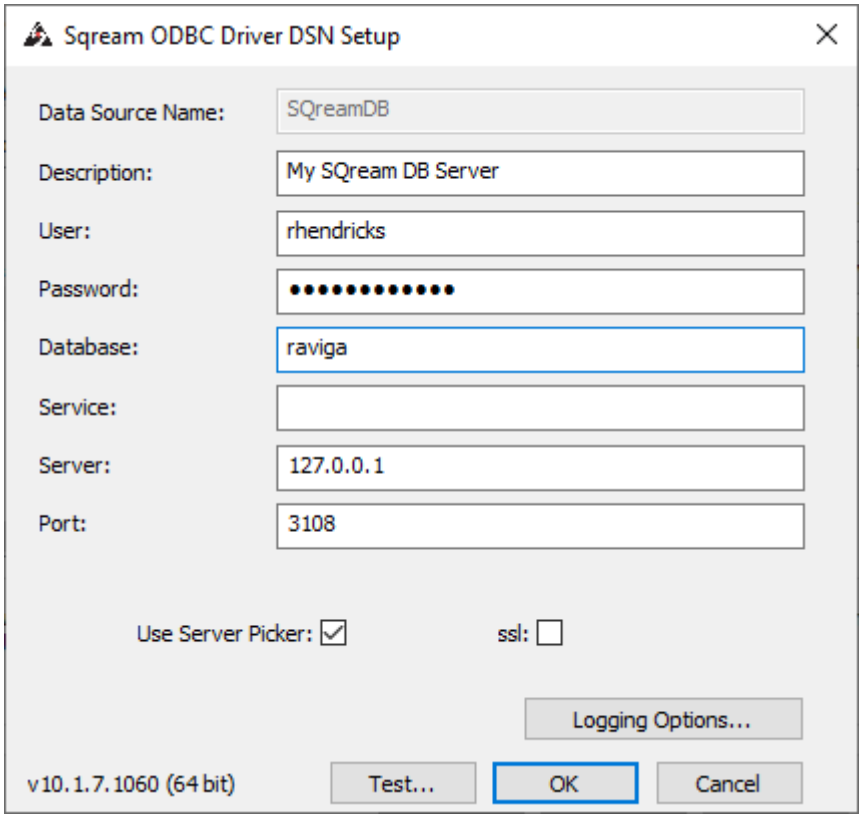


3. The installer has created a sample User DSN named **SQreamDB**

   You can modify this DSN, or create a new one (*Add ▸ SQream ODBC Driver ▸ Next*)

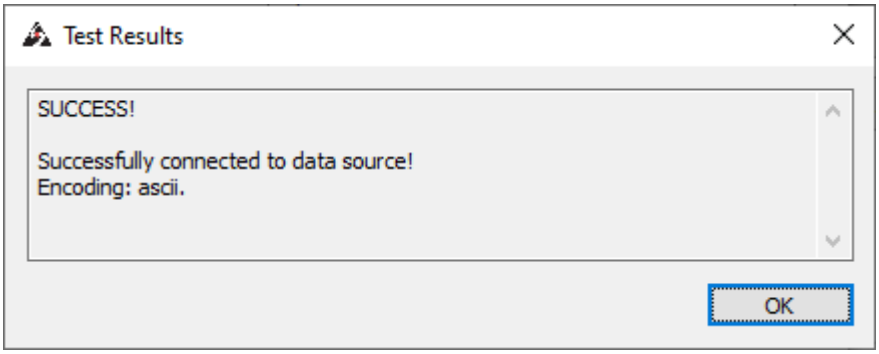


4. Enter your connection parameters. See the reference below for a description of the parameters.



5. When completed, save the DSN by selecting *OK*

Tip: Test the connection by clicking *Test* before saving. A successful test looks like this:



8.2.1.5.1.8 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.

8.2.1.5.1.9 Troubleshooting

8.2.1.5.1.10 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.
(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.

8.2.1.5.1.11 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.

8.2.1.5.2 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*

8.2.1.5.2.1 Prerequisites

8.2.1.5.2.2 unixODBC

The ODBC driver requires a driver manager to manage the DSNs. SQreamDB's driver is built for unixODBC.

Verify unixODBC is installed by running:

```
$ odbcinst -j
unixODBC 2.3.4
DRIVERS............: /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.: 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*

8.2.1.5.2.3 Install unixODBC on RHEL

```
$ yum install -y unixODBC unixODBC-devel
```

8.2.1.5.2.4 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
   $ 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 --install
```

3. Edit the DSN created by editing `/etc/.odbc.ini`. See the parameter explanation in the section *ODBC DSN Parameters*.

8.2.1.5.2.5 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 highlighted paths to match your specific 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
      ```

      Parameters are in the form of `parameter = value`. For details about the parameters that can be set for each DSN, see the section *ODBC DSN Parameters*.

3. Create a file called `.sqream_odbc.ini` for managing the driver settings and logging. This file should be created 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-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
```

8.2.1.5.2.6 Install the driver dependencies

Add the ODBC driver path to `LD_LIBRARY_PATH`:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/rhendricks/sqream_odbc64/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

8.2.1.5.2.7 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
```

8.2.1.5.2.8 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 SQreamDB worker. For example, `Server="127.0.0.1"` or `Server="sqream.mynetwork.co"`
Port	None	TCP port of the SQreamDB 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 SQreamDB Support. For example, `LogLevel=1` • 0 = Disable tracing • 1 = Error tracing • 2 = Error tracing • 3 = Warning tracing • 4 = Info tracing • 5 = Debug tracing

8.2.1.5.2.9 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) 8.x

Other distributions may also work, but are not officially supported by SQream.

8.2.1.5.3 Getting the ODBC driver

Download the relevant driver (Windows or Linux) from the *client drivers download page*.

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:

8.2.1.5.4 Install and configure the ODBC driver

Continue based on your platform:

- *Install and Configure ODBC on Windows*
- *Install and configure ODBC on Linux*

8.2.1.6 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 SQreamDB. 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*

- *Prepared Statements*

8.2.1.6.1 Installing the Python Connector

8.2.1.6.1.1 Prerequisites

It is essential that you have the following installed:

- *Python*
- *PIP*
- *OpenSSL for Linux*

8.2.1.6.1.2 Python

The connector requires Python version 3.9 or newer.

To see your current Python version, run the following command:

```
$ python --version
```

8.2.1.6.1.3 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 installing.

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 (1.4MB)
         |████████████████████████████████| 1.4MB 1.6MB/s
Installing collected packages: pip
  Found existing installation: pip 19.1.1
       Uninstalling pip-19.1.1:
         Successfully uninstalled pip-19.1.1
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 `WARNING: pip is configured with locations 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.

8.2.1.6.1.4 OpenSSL for Linux

The Python connector relies on OpenSSL for secure connections to SQreamDB. Some distributions of Python do not include OpenSSL.

To install OpenSSL on RHEL, run the following command:

```
$ sudo yum install -y libffi-devel openssl-devel
```

8.2.1.6.1.5 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.

8.2.1.6.1.6 Installing via PIP without an internet connection

1. To get the `.whl` package file, contact you SQreamDB support representative.
2. Run the following command:

```
tar -xf pysqream_connector_5.2.0.tar.gz
cd pysqream_connector_5.2.0
#Install all packages with --no-index --find-links .
python3 -m pip install *.whl -U --no-index --find-links .
python3.9 -m pip install pysqream-5.2.0.zip -U --no-index --find-links .
python3.9 -m pip install pysqream-sqlalchemy-1.3.zip  -U --no-index --find-links .
```

8.2.1.6.1.7 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
```

8.2.1.6.2 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.

8.2.1.6.2.1 Before You Begin

Download pysqream-sqlalchemy

8.2.1.6.2.2 Limitation

- Does not support the `ARRAY` data type

8.2.1.6.2.3 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',
                               username='<user_name>',
                               password='<password>',
                               host='<host_name>',
                               port=<port_number>,
                               port_ssl=<port_ssl>,
                               database='<database_name>')

engine = sa.create_engine(engine_url,connect_args={"clustered": False, "service": "
↪<service_name>"})
```

8.2.1.6.2.4 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


engine_url = sa.engine.url.URL('sqream',
                               username='sqream',
                               password='12345',
                               host='127.0.0.1',
```

(continues on next page)

(continued from previous page)

```
                                port=3108,
                                database='master')

engine = sa.create_engine(engine_url,connect_args={"clustered": True, "service":
↪"admin"})
table_df = pd.read_sql("select * from nba", con=engine)
```

8.2.1.6.3 API

- *Using the Cursor*
- *Reading Result Metadata*
- *Loading Data into a Table*
- *Using SQLAlchemy ORM to Create and Populate Tables*

8.2.1.6.3.1 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


con = pysqream.connect(host='127.0.0.1',
                       port=3108,
                       database='master',
                       username='rhendricks',
                       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_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()`:

```
# 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, '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', '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 (`[]`).

8.2.1.6.3.2 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',
                       port=3108,
                       database='master',
                       username='rhendricks',
                       password='Tr0ub4dor&3',
                       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']
```

8.2.1.6.3.3 Loading Data into a Table

This example shows how to load 10,000 rows of dummy data to an instance of SQreamDB.

To load data 10,000 rows of dummy data to an instance of SQreamDB:

1. Run the following:

```
import pysqream
import sqlalchemy.orm as orm
from datetime import date, datetime
from time import time


con = pysqream.connect(host='127.0.0.1',
                       port=3108,
                       database='master',
                       username='rhendricks',
```

(continues on next page)

(continued from previous page)

```
                        password='Tr0ub4dor&3',
                        clustered=True)

cur = con.cursor()
```

2. Create a table for loading:

```
create = 'create or 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 (?,?,?,?,?,?,?,?,?,?,?)'
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.close()
```

10. Close the connection:

```
con.close()
```

8.2.1.6.3.4 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 = sa.MetaData()
metadata.bind = engine
```

3. Create a table in the local metadata:

```
employees = sa.Table(
    'employees',
    metadata,
    sa.Column('id', sa.Integer),
    sa.Column('name', sa.TEXT(32)),
    sa.Column('lastname', sa.TEXT(32)),
    sa.Column('salary', sa.Float)
)
```

   The `create_all()` function uses the SQreamDB 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},
                {'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)
```

8.2.1.6.4 Prepared Statements

Prepared statements, also known as parameterized queries, are a safer and more efficient way to execute SQL statements. They prevent SQL injection attacks by separating SQL code from data, and they can improve performance by reusing prepared statements. In SQream, we use `?` as a placeholder for the relevant value in parameterized queries. Prepared statements `INSERT`, `SELECT`, `UPDATE` and `DELETE`

8.2.1.6.4.1 Prepared Statement Limitations

- Prepared Statement do not support the use of keywords_and_identifiers as input parameters.

8.2.1.6.4.2 Prepared Statements code example

```
import pysqream
from datetime import date, datetime
from time import time

# SQreamDB Connection Setting
con = pysqream.connect(host='<your-host-ip>', port=3108, database='master'
                                                                , username='<SQDB␣
↪role name>', password='<SQDB role password>'
                                                                , clustered=True)
cur = con.cursor()

# CREATE
create = 'create or 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)

# DATA
data = (False, 2, 12, 145, 84124234, 3.141, -4.3, "STRING1" , "STRING2", date(2024,␣
↪11, 11), datetime(2024, 11, 11, 11, 11, 11, 11))
row_count = 10**2

# Insert
insert = 'insert into perf values (?,?,?,?,?,?,?,?,?,?,?)'
start = time()

# Prepared Statement
cur.executemany(insert, [data] * row_count)
print (f"Total insert time for {row_count} rows: {time() - start} seconds")


# Results(Table Count)
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]}")


# SELECT
query = "SELECT * FROM perf WHERE s = ?"
params = [("STRING1",)]
```

(continues on next page)

(continued from previous page)

```
# Prepared Statement
cur.execute(query,params)


# Result
rows = cur.fetchall()
print(rows)

for row in rows:
                print(row)


# UPDATE
query = "UPDATE perf SET s = '?' WHERE s = '?'"
params = [("STRING3", "STRING2")]

# Prepared Statement
cur.execute(query,params)

print("Update completed.")


# DELETE
query = "DELETE FROM perf WHERE s = ?"
params = [("STRING1",)]

# Prepared Statement
cur.execute(query,params)

print("Delete completed.")

# Conn Close
cur.close()
con.close()
```

8.2.1.7 Spark

The Spark connector enables reading and writing data to and from SQreamDB and may be used for large-scale data processing.

- *Before You Begin*
- *Configuration*
- *Transferring Data*
- *Data Types and Mapping*
- *Example*

8.2.1.7.1 Before You Begin

To use Spark with SQreamDB, it is essential that you have the following installed:

- SQreamDB version 2022.1.8 or later
- Spark version 3.3.1 or later
- SQreamDB Spark Connector 5.0.0
- *JDBC* version 4.5.6 or later

8.2.1.7.2 Configuration

The Spark JDBC connection properties empower you to customize your Spark connection. These properties facilitate various aspects, including database access, query execution, and result retrieval. Additionally, they provide options for authentication, encryption, and connection pooling.

The following Spark connection properties are supported by SQreamDB:

Parameter	Default	Description
`url`		The JDBC URL to connect to the database.
`dbtable`		The JDBC URL to connect to the database.
`query`		The SQL query to be executed when using the JDBC data source, instead of specifying a table or view name with the dbtable property.
`driver`		The fully qualified class name of the JDBC driver to use when connecting to a relational database.
`numPartitions`		The number of partitions to use when reading data from a data source.
`queryTimeout`	0	The maximum time in seconds for a JDBC query to execute before timing out.
`fetchsize`	0	The number of rows to fetch in a single JDBC fetch operation.
`batchsize`	1000000	The number of rows to write in a single JDBC batch operation when writing to a database.
`sessionInitStatement`		A SQL statement to be executed once at the beginning of a JDBC session, such as to set session-level properties.
`truncate`	`false`	A boolean value indicating whether to truncate an existing table before writing data to it.
`cascadeTruncate`	the default cascading truncate behaviour of the JDBC database in question, specified in the `isCascadeTruncate` in each JDBCDialect	A boolean value indicating whether to recursively truncate child tables when truncating a table.
`createTableOptions`		Additional options to include when creating a new table in a relational database.
`createTableColumnTypes`		A map of column names to column data types to use when creating a new table in a relational database.
`customSchema`		A custom schema to use when reading data from a file format that does not support schema inference, such as CSV or JSON.
`pushDownPredicate`	`true`	A boolean value indicating whether to push down filters to the data source.
`pushDownAggregate`	`false`	A boolean value indicating whether to push down aggregations to the data source.
`pushDownLimit`	`false`	A boolean value indicating whether to push down limits to the data source.
`pushDownTableSample`	`false`	Used to optimize the performance of SQL queries on large tables by pushing down the sampling operation closer to the data source, reducing the amount of data that needs to be processed.
`connectionProvider`		A fully qualified class name of a custom connection provider to use when connecting to a data source.

8.2.1.7.2.1 Connecting Spark to SQreamDB

DataFrames, as Spark objects, play a crucial role in transferring data between different sources. The SQreamDB-Spark Connector facilitates the seamless integration of DataFrames, allowing the insertion of DataFrames into SQreamDB tables. Furthermore, it enables the export of tables or queries as DataFrames, providing compatibility with Spark for versatile data processing.

1. To open the Spark Shell, run the following command under the `Spark/bin` directory:

```
./spark-shell --driver-class-path {driver path}  --jars {Spark-Sqream-Connector.jar␣
↪path}


//Example:

./spark-shell --driver-class-path /home/sqream/sqream-jdbc-4.5.6.jar  --jars Spark-
↪Sqream-Connector-1.0.jar
```

2. To create a SQreamDB session, run the following commands in the Spark Shell:

```
import scala.collection.JavaConverters.mapAsJavaMapConverter
val config = Map("spark.master"->"local").asJava
import com.sqream.driver.SqreamSession;
val sqreamSession=SqreamSession.getSession(config)
```

8.2.1.7.3 Transferring Data

8.2.1.7.3.1 Transferring Data From SQreamDB to Spark

1. Create a mapping of Spark options:

```
val options = Map("query"->"select * from <table_name>", "url"->"jdbc:<jdbc_path>/
↪master;user=<username>;password=<password>;cluster=false").asJava
```

2. Create a Spark DataFrame:

```
val df=sqreamSession.read(options)
```

8.2.1.7.3.2 Transferring Data From Spark to SQreamDB

1. Create a mapping of Spark options, using the `dbtable` Spark option (`query` is not allowed for writing):

```
val options = Map("dbtable"-> <table_name>", "url"->"jdbc:<jdbc_path>/master;user=
↪<username>;password=<password>;cluster=false").asJava
```

2. Create a Spark DataFrame:

```
import org.apache.spark.sql.SaveMode
val df=sqreamSession.write(df, options, SaveMode.Overwrite)
```

8.2.1.7.4 Data Types and Mapping

SQreamDB data types mapped to Spark

SQreamDB	Spark
BIGINT	LONGINT
BOOL	BooleanType
DATE	DateType
DOUBLE	DoubleType
REAL	FloateType
DECIMAL	DeciamlType
INT	Integer
SMALLINT	ShortType
TINYINT	ShortType
DATETIME	TimestampType

Spark data types mapped to SQreamDB

Spark	SQreamDB
BooleanType	BOOL
ByteType	SMALLINT
DateType	DATE
DecimalType	DECIMAL
DoubleType	DOUBLE
FloatType	REAL
IntegerType	INT
LongType	BIGINT
ShortType	SMALLINT
StringType	TEXT
TimestampType	DATETIME

8.2.1.7.5 Example

JAVA

```
import com.sqream.driver.SqreamSession;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SaveMode;

import java.util.HashMap;

public class main {
        public static void main(String[] args) {
                HashMap<String, String> config = new HashMap<>();
                //spark configuration
                //optional configuration here: https://spark.apache.org/docs/latest/
↪configuration.html
                config.put("spark.master", "spark://localhost:7077");
                config.put("spark.dynamicAllocation.enabled", "false");
```

(continues on next page)

(continued from previous page)

```
                config.put("spark.driver.port", "7077");
                config.put("spark.driver.host", "192.168.0.157");
                config.put("spark.driver.bindAddress", "192.168.0.157");

                SqreamSession sqreamSession = SqreamSession.getSession(config);

                //spark properties
                //optional properties here: https://spark.apache.org/docs/latest/sql-
↪data-sources-jdbc.html
                HashMap<String, String> props = new HashMap<>();

                props.put("url", "jdbc:Sqream://192.168.0.157:3108/master;user=sqream;
↪password=1234;cluster=true;");

                //spark partition//
                props.put("dbtable", "public.test_table");
                props.put("partitionColumn","sr_date_sk");
                props.put("numPartitions","2");
                props.put("lowerBound","2450820");
                props.put("upperBound","2452822");


                /*Read from sqream table*/
                Dataset<Row> dataFrame = sqreamSession.read(props);
                dataFrame.show();/*By default, show() displays only the first 20 rows␣
↪of the DataFrame.
                This can be insufficient when working with large datasets. You can␣
↪customize the number of rows displayed by passing an argument to show(n).*//


                /*Add to sqream table*/
                sqreamSession.write(dataFrame, props, SaveMode.Append);

        }
}
```

8.2.1.8 Trino

If you are using Trino for distributed SQL query processing and wish to use it to connect to SQreamDB, follow these instructions.

- *Before You Begin*
- *Installation*
- *Connecting to SQreamDB*
- *Supported Data Types and Mapping*
- *Examples*
- *Limitations*

8.2.1.8.1 Before You Begin

It is essential you have the following installed:

- SQreamDB version 4.1 or later
- Trino version 403 or later
- Trino Connector
- *JDBC* version 4.5.6 or later

8.2.1.8.2 Installation

The Trino Connector must be installed on each cluster node dedicated to Trino.

1. Create a dedicated directory for the Trino Connector.
2. Download the Trino Connector and extract the content of the ZIP file to the dedicated directory.

8.2.1.8.3 Connecting to SQreamDB

Trino uses catalogs for referencing stored objects such as tables, databases, and functions. Each Trino catalog may be configured with access to a single SQreamDB database. If you wish Trino to have access to more than one SQreamDB database or server, you must create additional catalogs.

Catalogs may be created using `properties` files. Start by creating a `sqream.properties` file and placing it under `trino-server/etc/catalog`.

The following is an example of a properties file:

```
connector.name=<name>
connection-url=jdbc:Sqream://<host and port>/<database name>;[<optional parameters>; ...]
connection-user=<user>
connection-password=<password>
```

8.2.1.8.4 Supported Data Types and Mapping

Use the appropriate Trino data type for executing queries. Upon execution, incompatible data types will be converted by Trino to SQreamDB data types.

Trino type	SQreamDB type
BOOLEAN	BOOL
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
REAL	REAL
DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	DATETIME
VARCHAR	TEXT
DECIMAL(P,S)	NUMERIC(P,S)

8.2.1.8.5 Examples

The following is an example of the `SHOW SCHEMAS FROM` statement:

```
SHOW SCHEMAS FROM sqream;
```

The following is an example of the `SHOW TABLES FROM` statement:

```
SHOW TABLES FROM sqream.public;
```

The following is an example of the `DESCRIBE sqream.public.t` statement:

```
DESCRIBE sqream.public.t;
```

8.2.1.8.6 Limitations

The Trino Connector does not support the following SQL statements:

- `GRANT`
- `REVOKE`
- `SHOW GRANTSHOW ROLES`
- `SHOW ROLE GRANTS`

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:

- Overview
- *Avro*
- *CSV*
- *Parquet*
- *ORC*
- *JSON*
- *SQLoader As a Service*

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*
- *Foreign Data Wrapper Best Practice*
- *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 - *Amazon Web Services*
- Using the HDFS data storage system - *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*

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.
- 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*.

9.1.3 Foreign Data Wrapper Best Practice

A recommended approach when working with *Foreign Tables* and Foreign Data Wrapper (FDW) is storing files belonging to distinct file families and files with similar schemas in separate folders.

9.1.3.1 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.

9.1.3.2 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.

9.1.3.2.1 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	TINYI	SMALL-INT	INT	BIG-INT	REAL	DOU-BLE	Text [#f0]_	DATI	DATE-TIME
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.

9.1.3.3 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.

9.1.3.3.1 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:

[1] With UTF8 annotation
[2] With `TIMESTAMP_NANOS` or `TIMESTAMP_MILLIS` annotation
[3] Any microseconds will be rounded down to milliseconds.

SQream DB Type → ORC Source	BOOL	TINYI	SMALL-INT	INT	BIG-INT	REAL	DOU-BLE	TEXT	DATE	DATE-TIME
boolean	Sup-ported	Sup-ported[4]	Sup-ported[Page 1]	Sup-ported[4]	Sup-ported[4]					
tinyint	○[5]	Sup-ported	Sup-ported	Sup-ported	Sup-ported					
smallint	○[5]	○[6]	Sup-ported	Sup-ported	Sup-ported					
int	○[5]	○[6]	○[6]	Sup-ported	Sup-ported					
bigint	○[5]	○[6]	○[6]	○[6]	Sup-ported					
float						Sup-ported	Sup-ported			
double						Sup-ported	Sup-ported			
string / char / varchar								Sup-ported		
date									Sup-ported	Sup-ported
timestamp, timestamp with timezone										Sup-ported

- 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.

9.1.4 Further Reading and Migration Guides

For more information, see the following:

- copy_from
- insert
- *Foreign Tables*

[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

9.2 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 *HDFS Environment*.
- For S3, ensure network access to the S3 endpoint. For more information, see *Amazon Web Services*.

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 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
```

(continues on next page)

(continued from previous page)

```
  avro_fdw
OPTIONS
  (LOCATION = 's3://sqream-docs/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 nba (
  name TEXT(40),
  team TEXT(40),
  number BIGINT,
  position TEXT(2),
  age BIGINT,
  height TEXT(4),
```

(continues on next page)

(continued from previous page)

```
  weight BIGINT,
  college TEXT(40),
  salary FLOAT
)
WRAPPER
  avro_fdw
OPTIONS
  (LOCATION = 's3://sqream-docs/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
  < table_name >
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-docs/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:

```
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
```

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:/sqream-docs/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*

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 *HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Amazon Web Services* 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),
        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
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
        RECORD_DELIMITER = '\r\n',
    OFFSET = 2;
);
```

Repeat steps 3 and 4 for every CSV file you want to import.

9.3.5.1 Loading a standard CSV File From a Local Filesystem

```
COPY
  table_name
FROM
WRAPPER
  csv_fdw
OPTIONS (LOCATION = '/home/rhendricks/file.csv');
```

9.3.5.2 Loading a PSV (pipe separated value) file

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    DELIMITER = '|'
);
```

9.3.5.3 Loading a TSV (tab separated value) file

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    DELIMITER = '\t';
);
```

9.3.5.4 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
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    DELIMITER = E'\021'
);
```

9.3.5.5 Loading a Text File With Multi-Character Delimiters

In the file below, the separator is `'|`.

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
```

(continues on next page)

(continued from previous page)

```
    LOCATION = 's3://sqream-docs/nba.csv',
    DELIMITER = '''|'
);
```

9.3.5.6 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
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    DELIMITER = '|',
    OFFSET  2
);
```

9.3.5.7 Loading Files Formatted for Windows (`\r\n`)

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
        RECORD_DELIMITER = '\r\n',
    DELIMITER = '|'
);
```

9.3.5.8 Loading a File From a Public S3 Bucket

Note: The bucket must be publicly available and objects can be listed

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
```

(continues on next page)

(continued from previous page)

```
    LOCATION = 's3://sqream-docs/nba.csv',
    OFFSET = 2
);
```

9.3.5.9 Loading files from an authenticated S3 bucket

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    OFFSET = 2
    AWS_ID = '12345678',
    AWS_SECRET = 'super_secretive_secret'
);
```

9.3.5.10 Loading files from an HDFS storage

```
COPY
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 'hdfs://hadoop-nn.piedpiper.com/rhendricks/*.csv',
        RECORD DELIMITER = '\r\n',
    OFFSET = 2
);
```

9.3.5.11 Saving rejected rows to a file

See capturing_rejected_rows for more information about the error handling capabilities of `COPY FROM`.

```
COPY
  t
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
  );
```

9.3.5.12 Stopping the load if a certain amount of rows were rejected

```
COPY
  table
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION =  'filename.csv',
    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;
```

9.3.5.13 Load CSV files from a set of directories

Use glob patterns (wildcards) to load multiple files to one table.

```
COPY
  table
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION =  '/path/to/files/2019_08_*/*.csv'
);
```

9.3.5.14 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 (fifth, first, third)
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION =  '/path/to/files/*.csv'
);
```

Note: Any column not specified will revert to its default value or `NULL` value if nullable

9.3.5.15 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
  nba
FROM
WRAPPER
  csv_fdw
OPTIONS
  (
    LOCATION = 's3://sqream-docs/nba.csv',
    DATETIME_FORMAT = '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 ↓	BO	TINYINT	SMALLINT	INT	BIGINT	RE	DO BI	TI	DA	DATE-TIME
BOOLEAN	Supported									
INT16			Supported							
INT32				Supported						
INT64					Supported					
FLOAT						Supported				
DOUBLE							Supported			
BYTE_ARRAY / FIXED_LEN_BYTE_ARRAY								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 *HDFS Environment* guide.
- For S3, ensure network access to the S3 endpoint. For more information, see *Amazon Web Services* 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-docs/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://sqream-docs/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:

```
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
```

(continues on next page)

(continued from previous page)

```
↪          |  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:

Source Data Type / Casting	BOOLEAN	TINYINT	SMALLINT	INT	BIGINT	REAL	DOUBLE	TEXT[1]	DATE	DATETIME
boolean	Supported	Supported[2]	Supported[2]	Supported[2]	Supported[2]					
tinyint	Casting	Supported	Supported	Supported	Supported					
smallint	Casting	Casting[4]	Supported	Supported	Supported					
int	Casting	Casting[4]	Casting[4]	Supported	Supported					
bigint	Casting	Casting[4]	Casting[4]	Casting[4]	Supported					
float						Supported	Supported			
double						Supported	Supported			
string/char/text								Supported		
date								Supported	Supported	Supported[3]
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 *HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Amazon Web Services* 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),
  Salary FLOAT
)
```

(continues on next page)

(continued from previous page)

```
WRAPPER
  orc_fdw
OPTIONS
  (LOCATION = 's3://sqream-docs/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.

```
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,
```

(continues on next page)

(continued from previous page)

```
  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://sqream-docs/users/*.ORC',
    AWS_ID = 'our_aws_id',
    AWS_SECRET = 'our_aws_secret'
  );
```

(continues on next page)

(continued from previous page)

```
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 SQreamDB 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 SQreamDB.

- *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 *HDFS Environment*.
- For S3, ensure network access to the S3 endpoint. For more information, see *Amazon Web Services*.

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

SQreamDB can read the file metadata, enabling the automatic inference of column structure and data types.

```
CREATE FOREIGN TABLE nba
WRAPPER
  json_fdw
OPTIONS
  (LOCATION = 's3://sqream-docs/nba.json');
```

For more information, follow the CREATE FOREIGN TABLE page.

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
  nba
FROM
WRAPPER
  json_fdw
OPTIONS
  (LOCATION = 's3://sqream-docs/nba.json');
```

Note that JSON files generated using the `COPY TO` statement will store objects, and not object arrays.

```
COPY
  nba
TO
WRAPPER
  json_fdw
OPTIONS
(location = 's3://sqream-docs/nba.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 = 'sqream-docs.json');
```

The following is an example of loading data from a JSON file into SQream:

```
WRAPPER
  json_fdw
OPTIONS
  (LOCATION = 'sqream-docs.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 SQLoader As a Service

The **SQLoader** is a Java service that enables you to ingest data into SQreamDB from other DBMS and DBaaS through HTTP requests using network insert.

SQLoader supports ingesting data from the following DBMSs:

- Greenplum
- Microsoft SQL Server
- Oracle (including Oracle Autonomous Database)
- Postgresql
- SAP HANA
- Sybase
- Teradata
- SQreamDB 4.5.15 or later

- *Before You Begin*
- *Installation and Connectivity*
- *SQLoader Service Interface*
- *Creating Summary and Catalog Tables*
- *Data Type Mapping*

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 request runs on a single table, meaning concurrent imports of multiple tables require multiple requests. Additionally, it is important to note that for partitioned tables, each partition consumes a thread. Therefore, for performance efficiency, considering the table's partition count when managing thread allocation is a must.

Compute formula: $\lfloor 0.8 * (TotalMemory - 4) \rfloor$

9.7.2 Installation and Connectivity

9.7.2.1 Getting All Configuration and JAR Files

1. Download the SQLoader binary:
2. Extract the `.tar` file using the following command:

```
tar -xf sqloader_*.tar.gz
```

A folder named `sqloader` with the following files is created:

```
├── sqloader-v1.sh
├── bin
│   ├── sqloader-admin-server-1.1.jar
│   └── sqloader-service-8.2.jar
├── config
        ├── reserved_words.txt
        ├── sqload-jdbc.properties
        └── sqream-mapping.json
```

File Name	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-service-8.2.jar`	The SQLoader service JAR file
`sqloader-admin-server-1.0.jar`	The SQLoader admin server JAR file
`sqloader-v1.sh`	SQLoader service installer bash file

9.7.2.2 Installation

9.7.2.2.1 Deployment Parameters

When using the `sqloader-v1.sh` file (installer), the following flags are already configured.

All deployment flags are not dynamically adjustable at runtime.

Parameter	State	Default	Example	Description
configD:	Optional	config	`java -jar sqloaderServ 2.jar --configDir= path/to/ directory/ >`	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. This flag affects the mapping and reserved words files and does not affect the properties file
hz-Clus ter-Name:	Optional		`java -jar sqloader-ser 2.jar --hzClusterN`	In Hazelcast, a cluster refers to a group of connected Hazelcast instances across different JVMs or machines. By default, these instances connect to the same cluster on the network level, meaning that all SQLoader services that start on a network will connect to each other and share the same queue. An admin can connect to only one Hazelcast cluster at a time. If you start multiple clusters and want to connect them to the admin service, you will need to start multiple admin services, with each service connecting to one of your clusters. It is essential that this flag has the same name used here and across all SQLoader instances.
LOG_I	Optional	logs	`java -jar -DLOG_DIR=/ path/ to/log/ directory sqloader-ser 2.jar`	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
spri: boot admi: clie: url	Optional	http / loca	`java -jar sqloader-ser 2.jar --spring. boot. admin. client. url=http:/ /IP:PORT`	SQLoader admin server connection flag
Xmx	Optional		`java -jar -Xmx<number> sqloader-ser 2.jar`	We recommend using the `Xmx` flag to set the maximum heap memory allocation for the service. If a single service is running on the machine, we suggest allocating 80% of the total memory minus approximately 4GB, which the service typically needs on average. If multiple services are running on the same machine, calculate the recommended heap size for one service and then divide it by the number of services. Compute formula: $\lfloor 0.8 * (TotalMemory - 4) \rfloor$
DE-FAUL'	Manc tory	sqlc prop	`java -jar -DDEFAULT_PR path/ to/file/ sqload-jdbc. properties sqloader-ser 2.jar`	When the service initializes, it looks for the variable DEFAULT_PROPERTIES, which corresponds to the default sqload-jdbc.properties file. Once the service is running with a specified properties file, this setting will remain unchanged as long as the service is operational. To modify it, you must shut down the service, edit the properties file, and then restart the service. Alternatively, you can modify it via a POST request, but this change will only affect the specific load request and not the default setting for all requests.

9.7.2.2.2 Installing the Admin Server and SQLoader Service

1. To install the admin server, run the following command (install it only once on one machine):

```
sudo ./sqloader-v1.sh -admin
```

Output:

```
################################################################################
Welcome to SQloader Admin-Service installation
################################################################################
Please Enter JAVA_HOME PATH
/opt/java
################################################################################
The default PATH to install SQloader Admin Service is /usr/local/sqloader-admin
Do you want to change the default PATH ? (y/N)
################################################################################
The default PATH to SQloader-Admin logs directory is /var/log/sqloader-admin/logs
Do you want to change the default? (y/N)
################################################################################
Please enter HZCLUSTERNAME
sqcluster
################################################################################
SQloader-Admin default port is 7070 , Do you want to change the default port ? (y/N)
################################################################################
JAVA_HOME=/opt/java
BINDIR=/usr/local/sqloader-admin/
LOG_DIR=/var/log/sqloader-admin/
JAR=sqloader-admin-server-1.0.jar
ADMINPORT=7070
HZCLUSTERNAME=sqcluster
################################################################################
############# SQLoader-Admin Service installed successfuly ######################
################################################################################
To Start SQLoader-Admin Service: sudo systemctl start sqloader-admin
To View SQLoader-Admin Service status: sudo systemctl status sqloader-admin
################################################################################
```

2. To start the admin server, run the following command:

```
sudo systemctl start sqloader-admin
```

3. To verify admin server start status, run the following command (optional):

```
sudo systemctl status sqloader-admin
```

4. To install SQLoader service, run the following command (you can install per machine):

```
sudo ./sqloader-v1.sh -service
```

Output:

```
################################################################################
Welocome to SQloader service installation
################################################################################
Please Enter JAVA_HOME Path
/opt/java
```

(continues on next page)

(continued from previous page)

```
################################################################################
The Default PATH to install SQloader Service is /usr/local/sqloader
Do you want to change the default? (y/N)
################################################################################
The default PATH to SQloader Service logs directory is /var/log/sqloader-service
Do you want to change The default? (y/N)
################################################################################
Please enter SQloader Admin IP address
192.168.5.234
################################################################################
Please enter SQloader MEM size in GB
20
################################################################################
Please enter HZCLUSTERNAME
sqcluster
################################################################################
Default CONFDIR is /usr/local/sqloader/config , Do you want to change the default␣
↪CONFDIR ? (y/N)
################################################################################
Default SQloader Admin port is 7070 , Do you want to change the default port ? (y/N)
################################################################################
Default SQloader Service port is 6060 , Do you want to change the default port ? (y/N)
################################################################################
Default sqload-jdbc.properties is /usr/local/sqloader/config, Do you want to change␣
↪the default? (y/N)
Using default sqload-jdbc.properties PATH
/usr/local/sqloader/config
################################################################################
################################################################################
Using /usr/local/sqloader/config/sqload-jdbc.properties
################################################################################
JAVA_HOME=/opt/java
BINDIR=/usr/local/sqloader/bin
LOG_DIR=/var/log/sqloader-service
CONFDIR=/usr/local/sqloader/config
JAR=sqloader-service-8.2.jar
PROPERTIES_FILE=/usr/local/sqloader/config/sqload-jdbc.properties
PORT=6060
ADMINIP=192.168.5.234
ADMINPORT=7070
MEM=20
HZCLUSTERNAME=sqcluster
################################################################################
############# SQLoader Service installed successfuly #######################
################################################################################
To Start SQLoader Service: sudo systemctl start sqloader-service
To View SQLoader Service status: sudo systemctl status sqloader-service
################################################################################
```

5. To start the SQLoader service, run the following command:

```
sudo systemctl start sqloader-service
```

6. To verify SQLoader service start status, run the following command (optional):

```
sudo systemctl status sqloader-service
```

9.7.2.3 Reconfiguration

Admin server

You may reconfigure the admin server even after you have started it.

1. To get the configuration path, run the following command:

```
cat /usr/lib/systemd/system/sqloader-admin.service | grep 'EnvironmentFile'
```

Output:

```
EnvironmentFile=/usr/local/sqloader-admin/config/sqloader_admin.conf
```

2. Restart the admin server:

```
sudo systemctl restart sqloader-admin
```

SQLoader service

You may reconfigure the SQLoader service even after you have started it.

1. To get the configuration path, run the following command:

```
cat /usr/lib/systemd/system/sqloader-service.service | grep 'EnvironmentFile'
```

Output:

```
EnvironmentFile=/usr/local/sqloader/config/sqloader_service.conf
```

2. Restart the SQLoader service:

```
sudo systemctl restart sqloader-service
```

9.7.2.4 Connection String

It is recommended that the `sqload-jdbc.properties` file will contain a connection string.

1. Open the `sqload-jdbc.properties` file.
2. Configure connection parameters for:
   a. The source connection string: Greenplum, Microsoft SQL Server, Oracle, Postgresql, SAP HANA, Sybase or Teradata
   b. The target connection string: SQreamDB
   c. The *catalog* connection string: Greenplum, Microsoft SQL Server, Oracle, Postgresql, SAP HANA, SQreamDB, Sybase, or Teradata

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

```
# Postgresql, Oracle, Teradata, SAP HANA, Microsoft SQL Server, Sybase and SQreamDB Connection Strings
# (only one source connection string should be specified)

# postgres (and also Greenplum)
connectionStringSource=jdbc:postgresql://<HostIp:port>/<database_name>?user=<user_name>&password=<password>&ssl=<true/false>

# oracle
connectionStringSource=jdbc:oracle:thin:@//<HostIp:port>/<database_name>?user=<user_name>&password=<password>&ssl=<true/false>

# Oracle Autonomous Database

connectionStringSource=jdbc:oracle:thin:@<database_name>?tns_admin=<path_to_oracle_wallet>&user=<user>&password=<password>

# teradata
connectionStringSource=jdbc:teradata://<HostIp>/DATABASE=<database_name>,DBS_PORT=<port>,user=<user_name>,password=<password>

# sap hana
connectionStringSource=jdbc:sap://<HostIp>:<port>/?user=<user_name>&password=<password>

# microsoft sql server
connectionStringSource=jdbc:sqlserver://<HostIp>:<port>;databaseName=<database_name>;user=<user_name>;password=<password>;encrypt=<true/false>;trustServerCertificate=<true/false>

# sybase
connectionStringSource=jdbc:sybase:Tds:<HostIp>:<port>/<database_name>?user=<user_name>&password=<password>

# sqream
connectionStringSqream=jdbc:Sqream://<HostIp:port>/<database_name>;cluster=<true/false>;user=<user_name>;password=<password>



# Catalog Database Parameters

# Connection string (only one catalog connection string should be specified)
# Catalog database connection string on Oracle:
connectionStringCatalog=jdbc:oracle:thin:@//<HostIp:port>/<database_name>?user=<user_name>&password=<password>

# Catalog database connection string on SQreamDB:
connectionStringCatalog=jdbc:Sqream://<HostIp:port>/<database_name>;cluster=<true/false>;user=<user_name>;password=<password>



# CDC and Incremental Parameters
cdcCatalogTable=public.CDC_TABLES
```

(continues on next page)

(continued from previous page)

```
cdcTrackingTable=public.CDC_TRACKING
cdcPrimaryKeyTable=public.CDC_TABLE_PRIMARY_KEYS

# Summary table
loadSummaryTable=public.SQLOAD_SUMMARY



# OPTIONAL - Data transfer options
filter=1=1
count=true
limit=2000
threadCount=1
rowid=false
batchSize=500
fetchSize=100000
chunkSize=0
caseSensitive=false
truncate=true
drop=true
loadTypeName=full
cdcDelete=true
usePartitions=false
lockCheck=false
lockTable=true
loadDttm=false
useDbmsLob=false

.. more flags
```

9.7.3 SQLoader Service Interface

The SQLoader service automatically detects the IP addresses of incoming HTTP requests, even if the request originates from the same IP address as the one hosting the service. If you are accessing the service using a proxy server, you can include the client IP address in the request itself by using the `X-Forwarded-For` HTTP header, as in the following example:

```
curl -X POST -H 'X-Forwarded-For: 192.168.1.2' -H 'Content-Type: application/json' --
↪data '{"loadTypeName": "inc", "sourceSchema": "QA", "sourceTable": "MY_TABLE",
↪"sqreamTable": "MY_TABLE", "sqreamSchema": "QA"}' http://MyPc:6060/load
```

9.7.3.1 Supported HTTP Requests

Reques Type	Reques Nam	cURL Command	Description	Example
POST	load	`curl --header "Content-Type: application/json" --request POST --data '{}' http://127.0.0.1:6060/load`	Sends a request to the service and returns immediately. This HTTP request is utilized within a load-balancing queue shared across multiple instances. This setup ensures efficient resource utilization by distributing incoming load requests evenly across all available instances. Additionally, the system incorporates *high availability* mechanisms to recover failed jobs in case an instance crashes, ensuring continuous operation and reliability even during instance failures. Note that if all instances crash, at least one instance must remain operational to recover and execute pending jobs.	`curl --header "Content-Type: application/json" --request POST --data '{"sourceTable": "AVIV_INC", "sqreamTable": "t_inc", "limit":2000, "loadType-Name":"full"}' http://127.0.0.1:6060/load`
POST	syn- cLo	`curl --header "Content-Type: application/json" --request POST --data '{}' http://127.0.0.1:6060/syncLoad`	Sends a request to the service and returns once the request is complete. There's no load-balancing queue shared across multiple instances; therefore, it's advised that `syncLoad` requests be monitored by the user and not heavily sent. Monitor using the `getActiveLoads` cURL.	`curl --header "Content-Type: application/json" --request POST --data '{"sourceTable": "AVIV_INC", "sqreamTable": "t_inc", "limit":2000, "loadType-Name":"full"}' http://127.0.0.1:6060/syncLoad`
POST	fil- ter- Logs	`curl --header "Content-Type: application/json" --request POST --data '{"requestId": "output-FilePath": ""}' http://127.0.0.1:6060/filterLogs`	Retrieves logs for a specific request ID	`curl --header "Content-Type: application/json" --request POST --data '{"requestId":"request-1-6a288 "outputFilePath": "/home/avivs/sqloader_request.log"}' http://127.0.0.1:6060/filterLogs`
GET	get tive	`curl --header "Content-Type: application/json" --request GET http://127.0.0.1:6060/getActiveLoads`	Returns a list of all active loads currently running across all services	
GET	can- cel-	`curl --request`	Cancels an active request by request ID	`curl --request GET http://`

9.7.3.2 High Availability

SQLoader as a service supports high availability for asynchronous load requests only. When a service crashes, another service will take over the tasks and execute them from the beginning. However, there are some limited cases where high availability will not provide coverage:

- **At least one service must remain operational**: After a crash, at least one service must be up and running to ensure that tasks can be recovered and executed.
- **Limitations for specific tasks**: When any of the following is configured:
  - A task involving a `clustered` flag must be set to `true` to enable high availability.
  - A task involving a full load with `truncate=false` and `drop=false` will not rerun to prevent data duplication. In this type of load, data is inserted directly into the target table rather than a temporary table, making it impossible to determine if any data was inserted before the crash.

This setup ensures that asynchronous load requests are handled reliably, even in the event of service failures.

9.7.3.3 Log Rotation

Log rotation is based on time and size. At midnight (00:00) or when the file reaches 100MB, rotation occurs. Rotation means the log file `SQLoader_service.log` is renamed to `SQLoader_service_%d_%i.log` (%d=date, %i=rotation number), and a new, empty `SQLoader_service.log` file is created for the SQLoader service to continue writing to.

9.7.3.3.1 Log Automatic cleanup

The maximum number of archived log files to keep is set to 360, so Logback will retain the latest 360 log files in the logs directory. Additionally, the total file size in the directory is limited to 50 GB. If the total size of archived log files exceeds this limit, older log files will be deleted to make room for new ones.

9.7.3.4 SQLoader Request Parameters

Mandatory flags must be configured using HTTP flags or the `properties` file.

HTTP Parameter	State	Default	Description
`clustered`	Optional	`true`	This flag is relevant only f
`configFile`	Optional	`sqload-jdbc.properties`	Defines the path to the co
`connectionStringSqream`	Mandatory		JDBC connection string to
`connectionStringSource`	Mandatory		JDBC connection string to
`connectionStringCatalog`	Mandatory		JDBC connection string to
`cdcCatalogTable`	Optional		Part of the schema within
`cdcTrackingTable`	Optional		Part of the schema within
`cdcPrimaryKeyTable`	Optional		Part of the schema within
`loadSummaryTable`	Mandatory		Part of the schema within
`batchSize`	Optional	`10.000`	The number of records to
`caseSensitive`	Optional	`false`	If `true`, keeps table nam
`checkCdcChain`	Optional	`false`	Check CDC chain betwee
`chunkSize`	Optional	`0`	The number of records re
`columnListFilePath`	Optional		The name of the file that c
`columns`	Optional	All columns	The name or names of col

HTTP Parameter	State	Default	Description
count	Optional	true	Defines whether or not tab
cdcDelete	Optional	true	Defines whether or not loa
drop	Optional	false	Defines whether or not a r
fetchSize	Optional	100000	The number of records to
filter	Optional	1=1	Defines whether or not on
h, help	Optional		Displays the help menu an
limit	Optional	0 (no limit)	Limits the number of row
loadDttm	Optional	true	Add an additional loadD
loadDttmColumnName	Optional	sq_load_dttm	Specifies the name of the
loadTypeName	Optional	full	Defines a loading type tha
lockCheck	Optional	true	Defines whether or not SQ
lockTable	Optional	true	Defines whether or not SQ
partitionName	Optional		Specifies the number of ta
port	Optional	6060	
rowid	Optional	false	Defines whether or not SQ
sourceDatabaseName	Optional	ORCL	Defines the source databas
splitByColumn	Optional		Column name for split (re
sourceSchema	Mandatory		Source schema name to lo
sourceTable	Mandatory		Source table name to load
sqreamSchema	Optional	The schema name defined in the sourceSchema flag	Target schema name to lo
sqreamTable	Optional	The table name defined in the sourceTable flag	Target table name to load
threadCount	Optional	1	Number of threads to use
truncate	Optional	false	Truncate target table befo
typeMappingPath	Optional	config/sqream-mapping.json	A mapping file that conve
useDbmsLob	Optional	true	Defines whether or not SQ
usePartitions	Optional	true	Defines whether or not SQ
validateSourceTable	Optional	true	Allows control over the va

9.7.3.4.1 Using the `loadTypeName` Parameter

Using the `loadTypeName` parameter, you can define how you wish records' changes to be made to data in order to track inserts, updates, and deletes for data synchronization and auditing purposes.

Loading Type	Parameter Option	Supported Databases	Description
Full Table	full	All supported DBs	The entire data of the source table is loaded into SQreamDB
Change Data Capture (CDC)	cdc	Oracle	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. SQLoader recognizes tables by table name and metadata.
Incremental	inc	Oracle, PostgreSQL, SQreamDB	Only changes made to the source table data since last load will be loaded into SQreamDB. Changes include transactions of INSERT statement. SQLoader recognizes the table by table name and metadata.

9.7.3.5 Using the SQLoader Service Web Interface

The SQLoader Admin Server is a web-based administration tool specifically designed to manage and monitor the SQLoader service. It provides a user-friendly interface for monitoring data loading processes, managing configurations, and troubleshooting issues related to data loading into SQreamDB.

9.7.3.5.1 SQLoader Service Web Interface Features

- Monitor Services:
  - Health Checks: Monitor the health status of services to ensure they are functioning properly.
  - Metrics: Monitor real-time performance metrics, including CPU usage, memory usage, and response times.
  - Logging: View logs generated by services for troubleshooting and debugging purposes, and dynamically modify log levels during runtime to adjust verbosity for troubleshooting or performance monitoring.
- Manage Active Load Requests:
  - View a list of currently active data loading requests, including their status, progress, and relevant metadata.

9.7.4 Creating Summary and Catalog Tables

- *Creating a Summary Table*
- *Creating Catalog Tables*

The summary and catalog tables are pre-aggregated tables that store summarized or aggregated data.

9.7.4.1 Creating a Summary Table

The summary table is part of the schema within the database catalog.

The following summary table DDL uses Oracle syntax.

Note: If you are migrating from SQLoader as a process to **SQLoader as a service**, as described on this page, it is highly recommended that you add the following column to your existing summary table instead of re-creating it.

```
request_id            varchar2(200) default NULL,
client_ip             varchar2(200) default NULL,
requested_host        varchar2(200) default NULL,
acquired_host         varchar2(200) default NULL
```

```
# Use this DDL to create summary tables on Oracle database
      create table sqload_summary (
        db_name               varchar2(200 byte),
        schema_name           varchar2(200 byte),
        table_name            varchar2(200 byte),
        table_name_full       varchar2(200 byte),
        load_type             varchar2(200 byte),
        updated_dttm_from     date,
        updated_dttm_to       date,
```

(continues on next page)

(continued from previous page)

```
        last_val_int          number(22,0),
        last_val_ts           date,
        start_time            timestamp (6),
        finish_time           timestamp (6),
        elapsed_sec           number(*,0),
        row_count             number(*,0),
        sql_filter            varchar2(800 byte),
        partition             varchar2(200 byte),
        stmt_type             varchar2(200 byte),
        status                varchar2(200 byte),
        log_file              varchar2(200 byte),
        db_url                varchar2(400 byte),
        partition_count       number(*,0) default 0,
        thread_count          number(*,0) default 1,
        elapsed_ms            number(*,0) default 0,
        status_code           number(*,0) 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,
        request_id            varchar2(200) default NULL,
        client_ip             varchar2(200) default NULL,
        requested_host        varchar2(200) default NULL,
        acquired_host         varchar2(200) default NULL
      );

      create index sqload_summary_idx1 on sqload_summary (db_name,schema_name,table_
↪name);
      create index sqload_summary_idx2 on sqload_summary (start_time);
      create index sqload_summary_idx3 on sqload_summary (finish_time);
```

```
# Use this DDL to create summary tables SQDB databases
      CREATE TABLE sqload_summary (
       DB_NAME TEXT,
       SCHEMA_NAME TEXT,
       TABLE_NAME TEXT,
       TABLE_NAME_FULL TEXT,
       LOAD_TYPE TEXT,
       UPDATED_DTTM_FROM DATE ,
       UPDATED_DTTM_TO DATE ,
       LAST_VAL_INT NUMBER,
       LAST_VAL_TS DATETIME ,
       LAST_VAL_DT2 DATETIME2 ,
       START_TIME DATETIME ,
       FINISH_TIME DATETIME ,
       ELAPSED_SEC NUMBER ,
       ROW_COUNT NUMBER ,
       SQL_FILTER TEXT,
       PARTITION TEXT,
       STMT_TYPE TEXT,
       STATUS TEXT,
       LOG_FILE TEXT,
       DB_URL TEXT,
```

(continues on next page)

(continued from previous page)

```
      PARTITION_COUNT NUMBER  DEFAULT 0,
      THREAD_COUNT NUMBER  DEFAULT 1,
      ELAPSED_MS NUMBER  DEFAULT 0,
      STATUS_CODE NUMBER  DEFAULT 0,
      ELAPSED_SOURCE_MS NUMBER DEFAULT NULL,
      ELAPSED_SOURCE_SEC NUMBER DEFAULT NULL,
      ELAPSED_TARGET_MS NUMBER DEFAULT NULL,
      ELAPSED_TARGET_SEC NUMBER DEFAULT NULL,
      TARGET_DB_URL TEXT DEFAULT NULL,
      SQLOADER_VERSION TEXT DEFAULT NULL,
      CLIENT_IP TEXT DEFAULT NULL,
      REQUESTED_HOST TEXT DEFAULT NULL,
      ACQUIRED_HOST TEXT DEFAULT NULL,
      REQUEST_ID TEXT  DEFAULT NULL
      );
```

9.7.4.2 Creating Catalog Tables

CDC (Change Data Capture) and Incremental tables are database tables that record changes made to data in order to track inserts, updates, and deletes for data synchronization and auditing purposes.

See *Using the loadTypeName Parameter*

```
#To be used for Oracle
    create table cdc_tables (
      db_name               varchar2(200 byte),
      schema_name           varchar2(200 byte),
      table_name            varchar2(200 byte),
      table_name_full       varchar2(200 byte),
      table_name_cdc        varchar2(200 byte),
      inc_column_name       varchar2(200 byte),
      inc_column_type       varchar2(200 byte),
      load_type             varchar2(200 byte),
      freq_type             varchar2(200 byte),
      freq_interval         number(22,0),
      is_active             number(*,0) default 0,
      status_load           number(*,0));

    create index cdc_tables_idx1 on cdc_tables (db_name,table_name_full);

    create table cdc_table_primary_keys (
      db_name               varchar2(200 byte),
      schema_name           varchar2(200 byte),
      table_name            varchar2(200 byte),
      table_name_full       varchar2(200 byte),
      constraint_name       varchar2(200 byte),
      column_name           varchar2(200 byte),
      is_nullable           number(*,0));

    create index cdc_table_primary_keys_idx1 on cdc_table_primary_keys (db_name,table_
↪name_full);


    create table cdc_tracking (
      db_name               varchar2(200 byte),
```

(continues on next page)

(continued from previous page)

```
      schema_name           varchar2(200 byte),
      table_name            varchar2(200 byte),
      table_name_full       varchar2(200 byte),
      last_updated_dttm     date,
      last_val_int          number(22,0) default 0,
      last_val_ts           timestamp (6),
      last_val_dt           date);

    create index cdc_tracking_idx1 on cdc_tracking (db_name,table_name_full);
```

```
#To be used for SQDB
CREATE TABLE cdc_tracking (
  DB_NAME TEXT,
  SCHEMA_NAME TEXT,
  TABLE_NAME TEXT,
  TABLE_NAME_FULL TEXT,
  LAST_UPDATED_DTTM DATE ,
  LAST_VAL_INT NUMBER DEFAULT 0,
  LAST_VAL_TS DATETIME,
  LAST_VAL_DT DATETIME,
  LAST_VAL_DT2 DATETIME2
);

CREATE TABLE public.CDC_TABLES (
  DB_NAME TEXT(200),
  SCHEMA_NAME TEXT(200),
  TABLE_NAME TEXT(200),
  TABLE_NAME_FULL TEXT(200),
  TABLE_NAME_CDC TEXT(200),
  INC_COLUMN_NAME TEXT(200),
  INC_COLUMN_TYPE TEXT(200),
  LOAD_TYPE TEXT(200),
  FREQ_TYPE TEXT(200),
  FREQ_INTERVAL BIGINT,
  IS_ACTIVE INT DEFAULT 0,
  STATUS_LOAD INT DEFAULT 0,
  INC_GAP_VALUE INT DEFAULT 0
);

CREATE TABLE public.CDC_TABLE_PRIMARY_KEYS (
  DB_NAME TEXT(200),
  SCHEMA_NAME TEXT(200),
  TABLE_NAME TEXT(200),
  TABLE_NAME_FULL TEXT(200),
  CONSTRAINT_NAME TEXT(200),
  COLUMN_NAME TEXT(200),
  IS_NULLABLE INT DEFAULT 0
);
```

9.7.5 Data Type Mapping

- *Automatic Mapping*
- *Manually Adjusting Mapping*

9.7.5.1 Automatic Mapping

The **SQLoader** automatically maps data types used in Greenplum, Microsoft SQL Server, Oracle, Postgresql, Sybase, SAP HANA, and Teradata tables that are loaded into SQreamDB.

9.7.5.1.1 Greenplum

Greenplum 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

9.7.5.1.2 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

9.7.5.1.3 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

9.7.5.1.4 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

9.7.5.1.5 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

9.7.5.1.6 Sybase

Sybase Type	SQreamDB Type
CHAR, VARCHAR, LONG VARCHAR, CHARACTER, TEXT	TEXT
TINYINT	TINYINT
SMALLINT	SMALLINT
INT, INTEGER	INT
BIGINT	BIGINT
DECIMAL, NUMERIC	NUMERIC
NUMERIC(126,38)	NUMERIC(38,10)
FLOAT, DOUBLE	DOUBLE
DATE	DATE
DATETIME, TIMESTAMP, TIME	DATETIME
BIT	BOOL
VARBINARY, BINARY, LONG BINARY	TEXT

9.7.5.1.7 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

9.7.5.2 Manually Adjusting Mapping

You have the possibility to adjust the mapping process according to your specific needs, using any of the following methods.

9.7.5.2.1 `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
        }
}
```

9.7.5.2.1.1 Preparing Oracle for Data Migration

The preparation of incremental and Change Data Capture (CDC) tables is essential for efficiently tracking and managing changes to data over time, enabling streamlined data synchronization, and replication.

9.7.5.2.1.2 Preparing CDC Tables

1. Prepare the data table:

```
-- Drop the existing table if it exists
DROP TABLE cdc_example;

-- Create the main data table
CREATE TABLE cdc_example (
    id NUMBER(8) PRIMARY KEY,
    id_name VARCHAR(8),
    dttm TIMESTAMP,
    f_col FLOAT
);

-- Insert initial data into the table
INSERT INTO cdc_example (id, id_name, dttm, f_col) VALUES (-1, 'A', CURRENT_
↪TIMESTAMP, 0);

-- Verify the data in the table
SELECT * FROM cdc_example ORDER BY id DESC;
```

2. Prepare the CDC catalog:

```
-- Drop the CDC table if it exists
DROP TABLE cdc_example_cdc;

-- Create the CDC table to store change data
CREATE TABLE cdc_example_cdc (
    id NUMBER(8),
    id_name VARCHAR(8),
    row_id ROWID,
    updated_dttm DATE,
    type VARCHAR2(1)
);

-- Insert record to CDC_TABLES in the catalog
INSERT INTO public.CDC_TABLES (
    DB_NAME,
    SCHEMA_NAME,
    TABLE_NAME,
    TABLE_NAME_FULL,
    TABLE_NAME_CDC,
    INC_COLUMN_NAME,
    INC_COLUMN_TYPE,
    LOAD_TYPE,
    FREQ_TYPE,
    FREQ_INTERVAL,
    IS_ACTIVE,
    STATUS_LOAD,
    INC_GAP_VALUE
```

(continues on next page)

(continued from previous page)

```
) VALUES (
    'ORCL',
    'QA',
    'CDC_EXAMPLE',
    'QA.CDC_EXAMPLE',
    'QA.CDC_EXAMPLE_CDC',
    NULL,
    NULL,
    'CDC',
    NULL,
    NULL,
    1,
    0,
    0
);

-- Insert record to primary keys table in the catalog
INSERT INTO public.CDC_TABLE_PRIMARY_KEYS (
    DB_NAME,
    SCHEMA_NAME,
    TABLE_NAME,
    TABLE_NAME_FULL,
    CONSTRAINT_NAME,
    COLUMN_NAME,
    IS_NULLABLE
) VALUES (
    'ORCL',
    'QA',
    'CDC_EXAMPLE',
    'QA.CDC_EXAMPLE',
    NULL,
    'ID',
   0
);
```

3. Create trigger on data table:

```
-- Create a trigger on the data table to track changes and populate the CDC table
CREATE OR REPLACE TRIGGER cdc_example_tracking
AFTER UPDATE OR INSERT OR DELETE ON cdc_example
FOR EACH ROW
DECLARE
    l_xtn VARCHAR2(1);
    l_id INTEGER;
    l_id_name VARCHAR2(1);
    r_rowid ROWID;
BEGIN
    l_xtn := CASE
                WHEN UPDATING THEN 'U'
                WHEN INSERTING THEN 'I'
                WHEN DELETING THEN 'D'
             END;

        l_id_name := CASE
                     WHEN UPDATING THEN :NEW.id_name
                     WHEN INSERTING THEN :NEW.id_name
                     WHEN DELETING THEN :OLD.id_name
```

(continues on next page)

(continued from previous page)

```
                END;

        l_id := CASE
                WHEN UPDATING THEN :NEW.id
                WHEN INSERTING THEN :NEW.id
                WHEN DELETING THEN :OLD.id
            END;

        r_rowid := CASE
                   WHEN UPDATING THEN :NEW.rowid
                   WHEN INSERTING THEN :NEW.rowid
                   WHEN DELETING THEN :OLD.rowid
               END;

        INSERT INTO cdc_example_cdc (
        id,
        id_name,
        row_id,
        updated_dttm,
        type
        ) VALUES (
        l_id,
        l_id_name,
        r_rowid,
        SYSDATE,
        l_xtn
   );
END;
```

9.7.5.2.1.3 Preparing Incremental Table

1. Prepare the data table:

```
-- Create the data table for incremental loading
CREATE TABLE inc_example (
    ID INT PRIMARY KEY,
    name VARCHAR(8)
);

-- Insert initial data into the table
INSERT INTO inc_example (ID, name) VALUES (1, 'A');

-- Verify the data in the table
SELECT * FROM inc_example;
```

2. Prepare the CDC catalog:

```
-- Insert record into CDC_TABLES in the catalog
INSERT INTO public.CDC_TABLES (
    DB_NAME,
    SCHEMA_NAME,
    TABLE_NAME,
    TABLE_NAME_FULL,
    INC_COLUMN_NAME,
    INC_COLUMN_TYPE,
```

(continues on next page)

(continued from previous page)

```
    LOAD_TYPE,
    IS_ACTIVE,
    STATUS_LOAD
) VALUES (
    'ORCL',
    'QA',
    'INC_EXAMPLE',
    'QA.INC_EXAMPLE',
    'ID',
    'INT',
    'INC',
    1,
    0
);

-- Insert record into primary keys table in the catalog
INSERT INTO public.CDC_TABLE_PRIMARY_KEYS (
    DB_NAME,
    SCHEMA_NAME,
    TABLE_NAME,
    TABLE_NAME_FULL,
    COLUMN_NAME,
    IS_NULLABLE
) VALUES (
    'ORCL',
    'QA',
    'INC_EXAMPLE',
    'QA.INC_EXAMPLE',
    'ID',
    0
);
```

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
```

```
#PATH=$PATH:$HOME/.local/bin:$HOME/bin

#export PATH

# PS1
#MYIP=$(curl -s -XGET "http://ip-api.com/json" | python -c 'import json,sys;␣
↪jstr=json.load(sys.stdin); print jstr["query"]')
#PS1="\[\e[01;32m\]\D{%F %T} \[\e[01;33m\]\u@\[\e[01;36m\]$MYIP \[\e[01;31m\]\w\[\
↪e[37;36m\]\$ \[\e[1;37m\]"

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
```

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
```

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
```

   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 a 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 the next step.
    - 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.

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,
   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 Query Healer

The **Query Healer** periodically examines the progress of running statements, creating a log entry for all statements exceeding a defined time period.

11.1.1 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 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.

11.1.2 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.

11.1.3 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.

11.2 Compression

SQreamDB uses a variety of compression and encoding methods to optimize query performance and to save disk space.

- *Encoding*
- *Lossless Compression*
- *Best Practices*

11.2.1 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.

11.2.2 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.

11.2.2.1 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.

11.2.2.2 Compression Methods

The following table shows the supported compression methods:

Compression Metho	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
`SNAPP`	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
`SEQUENC`	`Integer`, `date`, and `timestamp`	Optimized RLE + Delta type for built-in identity columns.	GPU

11.2.2.3 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*

11.2.2.3.1 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"')
);
```

11.2.2.3.2 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
);
```

11.2.2.3.3 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.

11.2.2.4 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*

11.2.2.4.1 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;
```

11.2.2.4.2 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
```

11.2.2.4.3 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`).

11.2.3 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*

11.2.3.1 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.

11.2.3.2 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.

11.2.3.3 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.

11.3 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.

- *Before You Begin*
- *SQreamDB's UDF Support*

- *Working with Existing UDFs*
- *Permissions and Sharing*
- *Example*
- *Best Practices*

11.3.1 Before You Begin

- Ensure you have Python 3.9 or newer installed

Note: This feature is deprecated on Q3 2025, and would be replaced by an enhanced implementation. Please consult SQream support for usage.

11.3.2 SQreamDB's UDF Support

11.3.2.1 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.

11.3.2.2 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();
"Python version: 3.11.7 (main, Dec 22 2024, 18:29:20) [GCC 11.1.0]. Path: /usr/local"
```

11.3.2.3 Using Modules

To import a Python module, use the standard `import` syntax in the first lines of the user-defined function.

11.3.3 Working with Existing UDFs

11.3.3.1 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
```

11.3.3.2 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.

11.3.3.3 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.

11.3.4 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*.

11.3.5 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!"
```

11.3.6 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.

11.4 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.

11.4.1 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.

11.4.2 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.

11.4.2.1 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.

11.4.2.2 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
```

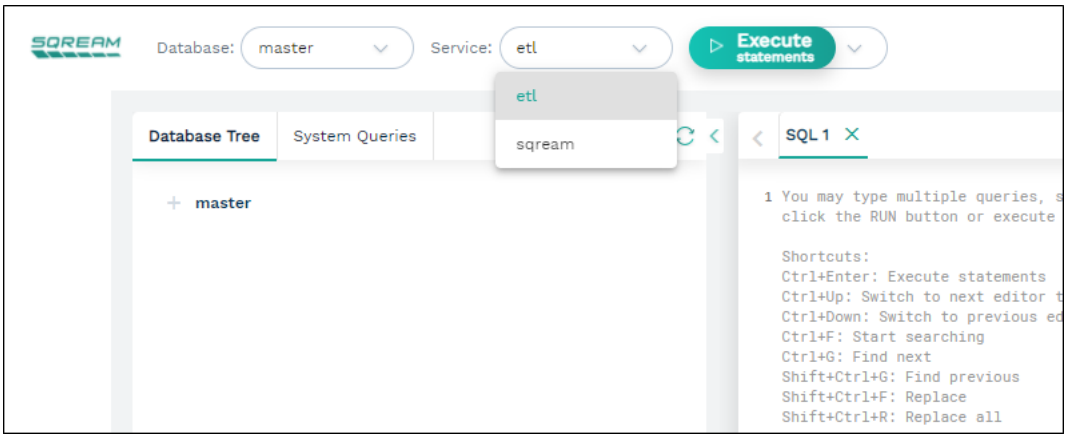
11.4.3 Configuring a Client Connection to a Specific Service

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*

11.4.3.1 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*.

11.4.3.2 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.

11.4.3.3 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.

11.4.3.4 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.

11.4.3.5 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.

11.4.3.6 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.

11.5 Concurrency and Locks

Locks are used in SQreamDB 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).

11.5.1 Locking Modes

SQreamDB 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.

11.5.2 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 `stateLockTimeout`.

11.5.3 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, SQreamDB applies locks during the table creation process to prevent the table from being altered during it's creation.

```
SELECT SHOW_LOCKS();
statement_id | statement_string                                                
```

11.6 Data Encryption

The **Data Encryption** page describes the following:

11.6.1 Overview

Data Encryption helps protect sensitive data at rest by concealing it from unauthorized users in the event of a breach. This is achieved by scrambling the content into an unreadable format based on encryption and decryption keys. Typically speaking, this data pertains to **PII (Personally Identifiable Information)**, which is sensitive information such as credit card numbers and other information related to an identifiable person.

Users encrypt their data on a column basis by specifying `column_name` in the encryption syntax.

The demand for confidentiality has steadily increased to protect the growing volumes of private data stored on computer systems and transmitted over the internet. To this end, regulatory bodies such as the **General Data Protection Regulation (GDPR)** have produced requirements to standardize and enforce compliance aimed at protecting customer data.

SQream enables customers to implement a-Symmetric Encryption solution using Secret Keys that they provide and manage themselves. The chosen encryption algorithm is AES-256, known for its strength and security. It is crucial to ensure that the Secret Key length is precisely 256 bits or 32 bytes.

For more information on the encryption syntax, see *Syntax*.

For more information on GDPR compliance requirements, see the GDPR checklist.

11.6.2 Encryption Methods

Data exists in one of following states and determines the encryption method:

11.6.2.1 Encrypting Data in Transit

Data in transit refers to data you use on a regular basis, usually stored on a database and accessed through applications or programs. This data is typically transferred between several physical or remote locations through email or uploading documents to the cloud. This type of data must therefore be protected while **in transit**. SQream encrypts data in transit using SSL when, for example, users insert data files from external repositories over a JDBC or ODBC connection.

For more information, see Use TLS/SSL When Possible.

11.6.2.2 Encrypting Data at Rest

Data at rest refers to data stored on your hard drive or on the cloud. Because this data can be potentially intercepted **physically**, it requires a form of encryption that protects your data wherever you store it. SQream facilitates encryption by letting you encrypt any column.

11.6.3 Data Types

Typically speaking, sensitive pertains to **PII (Personally Identifiable Information)**, which is sensitive information such as credit card numbers and other information related to an identifiable person.

SQream's data encryption feature supports encrypting column-based data belonging to the following data types:

- `INT`
- `BIGINT`
- `TEXT`

For more information on the above data types, see *Supported Data Types*.

11.6.4 Syntax

Encrypting columns in a new table

```
CREATE TABLE <table name>  (
     <column_name> <data_type> ENCRYPT,
     <column_name> <data_type> NULL ENCRYPT,
             <column_name> <data_type> NOT NULL ENCRYPT
             );
```

Adding an encrypted column to an existing table

```
ALTER TABLE <table_name> ADD COLUMN <column_name> <data_type> ENCRYPT;
```

Encryption methods

```
ENCRYPT ( <column name to encrypt> , <Secret Key of exactly 256-bit (32-byte) length>␣
↪)
```

Decryption method

```
DECRYPT ( <column name to decrypt> , <Secret Key of exactly 256-bit (32-byte) length>␣
↪)
```

11.6.5 Examples

Encrypting a new table

```
CREATE TABLE client_name  (
     id BIGINT NOT NULL ENCRYPT,
     first_name TEXT ENCRYPT,
     last_name TEXT,
     salary INT ENCRYPT);
```

Inserting encrypt player salary (`INT` data type)

```
INSERT INTO NBA (player_name, team_name, jersey_number, position, age, height, weight,
↪ college, salary)
VALUES ('Jayson Christopher Tatum', 'Boston Celtics', 0, 'SF', 25, '6-8', 210 , 'Duke
↪', ENCRYPT ( 32600060 ,
↪'6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f'));
```

Similar example using `COPY FROM`

```
COPY NBA
(
player_name, team_name, jersey_number, position, age, height, weight, college,
ENCRYPT (salary, '6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f')
)
FROM WRAPPER csv_fdw
OPTIONS
(location = '/tmp/source_file.csv', quote='@');
```

Query the encrypted data

```
SELECT player_name, DECRYPT( salary,
↪'6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f') FROM NBA
WHERE player_name ='Jayson Christopher Tatum';

player_name              |salary    |
-------------------------+----------+
Jayson Christopher Tatum|1500000   |
```

Query the encrypted data using `WHERE` clause on an encrypted column

```
SELECT player_name, DECRYPT( salary,
↪'6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f')
FROM NBA
WHERE DECRYPT( salary,
↪'6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f') > 1000000;

player_name              |salary    |
-------------------------+----------+
Jayson Christopher Tatum|1500000   |
------------------------+----------+
Marcus Smart             |1350000   |
```

Example of `COPY TO` using `DECRYPT`

```
COPY
  (SELECT player_name, DECRYPT( salary,
↪'6a8431f6e9c2777ee356c0b8aa3c12c0c63bdf366ac3342c4c9184b51697b47f')
  FROM NBA
  WHERE player_name ='Jayson Christopher Tatum')
TO WRAPPER parquet_fdw
OPTIONS (LOCATION = '/tmp/file.parquet');
```

11.6.6 Limitations

- The following functionality is not supported by the encryption feature: `Catalog queries`, `Utility commands`, `Foreign Tables`, `Create AS SELECT`.
- A single encryption key must be used per column - using a different key would result in an error.
- Compression of encrypted columns is limited to the following types: `Flat`, `LZ4`, `PD4`, `DICT`, `RLE`.
- This feature is not backward compatible with previous versions of SQreamDB.
- The encryption feature affect performance and compression.

11.6.7 Permissions

The Data Encryption feature does not require a specific permission, users with relevant **TABLE** and **COLUMN** permissions may utilize it.

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
key-words_and_ident	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_expressior	Scalar expressions are single-value computations that operate on one or more values to produce a single result.
cross_database_c	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.
com-mon_table_expre	Common Table Expressions (CTEs) are named temporary result sets that simplify complex queries by allowing the definition of subqueries for better readability and reusability.
win-dow_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.
sqream_scripting	Metalanguage scripting enhances your interaction with SQL by providing conventions which allow dynamic generation, management, and automation of SQL code.
pivot_unpivot	convert row-level data into columnar representation.

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
DROP SAVED QUERY	Drops a saved query
DUMP DATABASE DDL	View the `CREATE TABLE` statement for an current database
EXECUTE SAVED QUERY	Executes a previously saved query
EXPLAIN	Returns a static query plan, which can be used to debug query plans
export_open_snapshots	Lists and saves information about all currently open snapshots to a specified file
get_chunk_info	Retrieves information of specific chunks
GET DDL	View the `CREATE TABLE` statement for a table
get_extent_info	Retrieves information of specific extents
GET FUNCTION DDL	View the `CREATE FUNCTION` statement for a UDF
GET LICENSE INFO	View a user's license information
GLOBAL GRACEFUL SHUTDOWN	Graceful shutdown of all servers in the cluster
GPU METRICS	Monitor license quota usage by reviewing monthly or daily GPU usage
get_open_snapshots	Lists information about all currently open snapshots
GET TOTAL CHUNKS SIZE	Returns the total size of all data chunks saved in the system
GET VIEW DDL	View the `CREATE VIEW` statement for a view
HEALTH CHECK MONITORING	Returns system health monitoring logs
LDAP GET ATTR	Enables you to specify the LDAP attributes you want the SQreamDB role catalog table to show
LIST SAVED QUERIES	Lists previously saved query names, one per row
RECHUNK	Enables you to merge small data chunks into larger ones
RECOMPILE SAVED QUERY	Recompiles a saved query that has been invalidated due to a schema change
RECOMPILE VIEW	Recreate a view after schema changes

continues on next page

Table 1 – continued from previous page

Command	Usage
REMOVE LOCK	Clears locks
REMOVE STATEMENT LOCKS	Clears all locks in the system
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
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
SHOW VERSION	Returns the system version for SQream DB
swap_table_names	Swaps the names of two tables contained within a schema

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_instance	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_dd	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_permission	Returns a list of permissions required to run a statement or query
grant	Grant permissions to a role
grant_usage_on_service_t	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
is_table_exists	Checks if the mentioned table exists in the mentioned schema
is_view_exists	Checks if the mentioned view exists in the mentioned schema

12.1.3.1.1.3 Conversion

The following table shows the **conversion** functions:

Function	Description
from_unix	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 2: 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 3: 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 ($\sqrt{x}$)
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
concat_function	Concatenates multiple 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
- current_timestamp2
- 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
`databas`	Shows the name of the database containing the table.
`ta-ble_id`	Shows the ID of the table containing the column.
`schema_`	Shows the name of the schema containing the table.
`ta-ble_nam`	Shows the name of the table containing the column.
`clus-ter-ing_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_	Shows the name of the database containing the table.
schema_n	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_nam	Shows the column's data type. For more information see *Supported Data Types*.
column_size	Shows the maximum length in bytes.
has_defa	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_va	Shows the column's default value. For more information, see Default Value Constraints.
compression_str	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
databas	Shows the database's unique ID.
databas	Shows the database's name.
default_d	Reserved for internal use.
default_p	Reserved for internal use.
rechunk	Reserved for internal use.
storage_sub	Reserved for internal use.
compression_ch	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_na`	Shows the flag name
`value`	Shows the current flag configured value
`de-fault_v`	Shows the flag default value
`scope`	Shows whether flag configuration is session-based or cluster-based
`de-scrip-tion`	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
`per-mis-sion_ty`	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
`databas`	Shows the database that the default permission rule applies to.
`schema_`	Shows the schema that the rule applies to, or `NULL` if the `ALTER` statement does not specify a schema.
`modi-fier_rc`	Shows the role to apply the rule to.
`get-ter_rol`	Shows the role that the permission is granted to.
`per-mis-sion_ty`	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
`databas`	Shows the database that the default permission rule applies to.
`modi-fier_rc`	Shows the role to apply the rule to.
`get-ter_rol`	Shows the role that the permission is granted to.
`per-mis-sion_ty`	Shows the type of permission granted.
`get-ter_rol`	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
`databas`	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_ty`	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
`databas`	Shows the name of the database the permission applies to
`role_id`	Shows the ID of the role granted permissions.
`permission_ty`	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
`databas`	Shows the name of the database containing the schema.
`schema_`	Shows the ID of the schema the permission applies to.
`role_id`	Shows the ID of the role granted permissions.
`permission_ty`	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_par`	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_pas`	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_rol`	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_`	Shows the schema's unique ID.
`schema_`	Shows the schema's name.
`schema_`	Shows the name of the role that owns the schema.
`rechun-ker_ign`	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
`databas`	Shows the name of the database containing the table.
`ta-ble_id`	Shows the table's database-unique ID.
`schema_`	Shows the name of the schema containing the table.
`ta-ble_nam`	Shows the name of the table.
`row_cou`	Identifies whether the `row_count` can be used.
`row_cou`	Shows the number of rows in the table.
`rechun-ker_ign`	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
`databa`	Shows the name of the database containing the table.
`table_id`	Shows the table's database-unique ID.
`schema`	Shows the name of the schema containing the table.
`table_na`	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_sc`	Shows the name of the schema containing the view.
`view_na`	Shows the name of the view.
`view_da`	Reserved for internal use.
`view_qu`	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
`databas`	Shows the name of the database containing the view.
`function_id`	Shows the UDF's database-unique ID.
`function_na`	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
`databas`	Shows the name of the databse containing the extent.
`ta-ble_id`	Shows the ID of the table containing the extent.
`col-umn_id`	Shows the ID of the column containing the extent.
`ex-tent_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
`databas`	Shows the name of the databse containing the extent.
`ta-ble_id`	Shows the ID of the table containing the extent.
`col-umn_id`	Shows the ID of the column containing the extent.
`chunk_i`	Shows the chunk ID.
`ex-tent_id`	Shows the extent ID.
`com-pressed`	Shows the compressed chunk size in bytes.
`un-com-pressed`	Shows the uncompressed chunk size in bytes.
`com-pres-sion_ty`	Shows the chunk's actual compression scheme.
`long_mi`	Shows the minimum numeric value in the chunk (if one exists).
`long_ma`	Shows the maximum numeric value in the chunk (if one exists).
`string_`	Shows the minimum text value in the chunk (if one exists).
`string_`	Shows the maximum text value in the chunk (if one exists).
`off-set_in_`	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
`databas`	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_nu`	Shows the amount of rows in the chunk.
`deletion_st`	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
`databas`	Shows the name of the databse containing the predicate.
`table_id`	Shows the ID of the table containing the predicate.
`max_chu`	Reserved for internal use, this is a placeholder marker for the highest `chunk_id` logged during the `DELETE` operation.
`delete_`	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 4: User CLIs

Command	Usage
sqream sql	Built-in SQL client

Table 5: 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 6: 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_logs`	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 Multi-Platform Sqream SQL

SQreamDB comes with a built-in client for executing SQL statements either interactively or from the command-line.

- *Before You Begin*
- *Installing Sqream SQL*
- *Using Sqream SQL*
- *Examples*
- *Operations and Flag References*

12.3.3.1 Before You Begin

Sqream SQL requires Java 8

12.3.3.2 Installing Sqream SQL

If you have a SQreamDB installation on your server, `sqream sql` can be found in the `bin` directory of your SQreamDB installation, under the name `sqream`.

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=> _
```

12.3.3.3 Using Sqream SQL

By default, sqream sql runs in interactive mode. You can issue commands or SQL statements.

12.3.3.3.1 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 ␣
```

(continues on next page)

(continued from previous page)

```
↪ ,\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
```

12.3.3.3.2 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 (>).

12.3.3.3.3 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

12.3.3.4 Examples

12.3.3.4.1 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=>_
```

12.3.3.4.2 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

farm=> select * from animals;
1,goat                          ,0
4,bull                          ,1
2 rows
time: 0.029299s
```

12.3.3.4.3 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
```

12.3.3.4.4 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

12.3.3.4.4.1 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
```

12.3.3.4.4.2 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
```

12.3.3.4.5 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
```

12.3.3.4.6 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
```

12.3.3.4.7 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=>_
```

12.3.3.5 Operations and Flag References

12.3.3.5.1 Command Line Arguments

Sqream SQL supports the following command line arguments:

Argument	Default	Description
`-c` or `--comman`	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 SQreamDB worker.
`--port`	`5000`	Sets the connection port.
`--databa` or `-d`	None	Specifies the database name for queries and statements in this session.
`--usernai`	None	Username to connect to the specified database.
`--passwo`	None	Specify the password using the command line argument. If not specified, the client will prompt the user for the password.
`--cluste`	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 SQreamDB worker.
`--servic`	`sqrea`	*Service name (queue)* that statements will file into.
`--result`	False	Outputs results only, without timing information and row counts
`--no-his`	False	When set, prevents command history from being saved in `~/.sqream/clientcmdhist`
`--delimi`	`,`	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

12.3.3.5.1.1 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)

12.3.3.5.2 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

12.3.3.5.3 Basic Commands

Command	Description
`Ctrl-l`	Clear the screen.
`Ctrl-c`	Terminate the current command.
`Ctrl-z`	Suspend/stop the command.
`Ctrl-d`	Quit SQream SQL

12.3.3.5.4 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.

12.3.3.5.5 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.

12.3.4 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_se`	`3105`	The metadata server listening port
`--metadata_se`	`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_int`	`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
`--log_path`	`./server_picker_logs`	Configures the default location for the log file

12.3.4.1.2 Example

```
server_picker --metadata_server_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 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.

12.3.6.1 Running upgrade_storage

`upgrade_storage` can be found in the `bin` directory of your SQream DB installation.

12.3.6.1.1 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_predicate`	Option	Allows the upgrade process to proceed even if there are predicates marked for deletion.

12.3.6.1.2 Syntax

```
$ upgrade_storage <storage path> [--check_predicates=0]
```

```
$ upgrade_storage <storage path> [--storage_version]
```

12.3.6.2 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.

12.3.6.3 Examples

12.3.6.3.1 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
upgrade_storage to 25
upgrade_storage to 25 - Done
upgrade_v26
upgrade_storage to 26
upgrade_storage to 26 - Done
```

(continues on next page)

(continued from previous page)

```
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, - 9,223,372,036,	8 bytes	36124441255243	
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	0.12324567890123456789	DECIMAL, NUMBER
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	Date and time in nanosecond precision including UTC offset	16 bytes	'1955-11-05 01:24:00.000000000 +00:00'	NA

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 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])
```

13.1.2.2 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']),`

13.1.2.3 Examples

- *ARRAY Statements*
- *Ingesting Arrayed Data from External Files*

13.1.2.3.1 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],
    ARRAY [0.4354,0.5365435,3.6456]
  );
```

Converting arrayed elements into a set of rows:

```
SELECT
  UNNEST(clmn1) FROM my_array;
```

```
 clmn1  |
--------+
 1      |
 2      |
 3      |
```

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]
```

13.1.2.3.2 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",
    "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;
```

Output:

```
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]
```

13.1.2.4 Limitations

13.1.2.4.1 Casting Limitations

13.1.2.4.1.1 NUMERIC

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;
```

13.1.2.4.1.2 TEXT

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;
```

13.1.2.4.2 Connectors

13.1.2.4.2.1 .NET and ODBC

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.

13.1.2.4.2.2 Pysqream

Please note that SQLAlchemy does not support the `ARRAY` data type.

13.1.2.4.3 Functions

13.1.2.4.3.1 || (Concatenate)

Using the `||` (Concatenate) function with two different data types requires explicit casting.

```
SELECT (clmn1, 4::tinyint) || (clmn2, 5::tinyint) FROM my_array;
```

13.1.2.4.3.2 UNNEST

The `UNNEST` function is computationally intensive; therefore, it is recommended to use up to 10 `UNNEST` clauses per statement as a best practice. To improve performance, consider filtering the results before applying `UNNEST` to reduce data volume and optimize query runtime.

13.1.2.4.3.3 Window

Window functions are not supported.

13.2 Casts and Conversions

SQreamDB 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 SQreamDB.

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)
```

13.2.1 Conversion Methods

SQreamDB supports the following data conversion methods:

- `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.

13.2.2 Supported Casts

The listed table of supported casts also applies to the *Array* data type. For instance, you can cast a `NUMERIC[]` array to a `TEXT[]` array.

FROM / TO	BOO	TINYINT/SMALLINT/I	REAL/FL(	NU-MERIC	DATE/DATETIME/D/	TEXTME2
BOOL	N/A	✓	✗	✗	✗	✓
TINYINT/SMALLINT/I	✓	N/A	✓	✓	✗	✓
REAL/FLOAT	✗	✓	N/A	✓	✗	✓
NUMERIC	✗	✓	✓	✓	✗	✓
DATE/DATETIME/DA	✗	✗	✗	✗	✓	✓
TEXT	✓	✓	✓	✓	✓	N/A

13.2.2.1 Value Dependent Conversions

Conversions between certain data types may be value-dependent, as the outcome can vary based on the specific values being converted and their compatibility with the target data type's range or precision.

For example:

```
CREATE OR REPLACE TABLE t(xint INT, xtext TEXT);
INSERT INTO t VALUES(1234567, 'abc');

-- yields cast overflow:
SELECT xint::TINYINT FROM t;

-- yields Unsupported conversion attempt from string to number - not all strings are␣
↪numbers:
SELECT xtext::INT FROM t;
```

(continues on next page)

(continued from previous page)

```
CREATE OR REPLACE TABLE t(xint INT, xtext TEXT);
INSERT INTO t VALUES(12, '12');

-- yields 12 in both cases:
SELECT xint::TINYINT FROM t;

SELECT xtext::INT FROM t;
```

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 DATE 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, while a `DATETIME` column is 8 bytes.

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'

13.3.7 DateTime2

DATETIME2 is a type designed for storing year, month, day, hour, minute, seconds, milliseconds, microseconds, nanoseconds and UTC offset with 1 nanosecond precision.

13.3.7.1 Aliases

NA

13.3.7.2 Syntax

DATETIME2 values are formatted as string literals.

The following is an example of the DATETIME2 syntax:

```
``1955-11-05 01:24:00.000000000 +00:00``
```

DATETIME2 values are formatted as string literals conforming to ISO 8601.

SQream attempts to guess if the string literal is a datetime2 based on context, for example when used in date-specific functions.

13.3.7.3 Size

A DATETIME2 column is 16 bytes.

However, the size of these values is compressed by SQream DB.

13.3.7.4 Date Examples

The following is an example of the Date syntax:

```
CREATE TABLE important_dates (a DATETIME2);

INSERT INTO important_dates VALUES ('1955-11-05 01:24:00.000000000 -08:00');

SELECT * FROM important_dates;
```

The following is an example of the correct output:

```
1955-11-05 01:24:00.000 -0800
```

13.3.7.5 Date Casts and Conversions

The following table shows the possible `DATETIME2` value conversions:

Type	Details
`TEXT`	`'1997-01-01'` → `'1997-01-01 00:00:00.000000 +00:00'`, `'1955-11-05 01:24'` → `'1955-11-05 01:24:00.000000 +00:00'`
`DATE`	`'1997-01-01'` → `'1997-01-01 00:00:00.000000 +00:00'`
`DATETIME`	`'1955-11-05 01:24'` → `'1955-11-05 01:24:00.000000 +00:00'`

CHAPTER

FOURTEEN

RELEASE NOTES

Version 4.11 - April 9th, 2025

Version 4.10 - January 20, 2025

Version 4.9 - November 28, 2024

Version 4.8 - October 06, 2024

- Prepared statements are now supported by our Python and JDBC client drivers.
- PIVOT and UNPIVOT.
- Window funtion alias allows to specify a parameter within the window function definition. This eliminates the need to repeatedly input the same SQL code in queries that use multiple window functions with identical definitions.
- CONCAT function concatenates one or more strings, or concatenates one or more binary values.

Version 4.7 - September 01, 2024

- *AWS private cloud deployment* is now available for SQreamDB on AWS Marketplace.
- Execute a single SQL statement across your SQreamDB cluster using the new Cross-Database syntax.
- Safely cast data types with the new IsCastable function.
- Automatically delete source files being copied into SQreamDB using the copy_from command.

Version 4.6 - August 20, 2024

- You can now sign in to SQreamDB Studio using your universal *Single Sign-On (SSO)* provider authentication
- Announcing a new *Activity Report* reflecting your storage and resource usage
- Announcing a new Java-based cross-platform SQream SQL CLI
- `TOP` clause enhancements
- *Saved Query* command permission enhancements

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, 2023

- 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 SQDB release policy

14.1.1 Release Cadence

- **Major Versions** Major product versions are released when significant architectural changes, or substantial improvements are ready for deployment. There is no fixed release cadence for major versions.
- **Minor Versions** SQream releases minor product versions once per quarter. Assuring our customers benefit new features and improvements.
- **Patches** Patches and hotfixes will be releases as needed to address specific issues.

14.1.2 Transition to Maintenance Mode

- Upon release of a new version, the previous versions immediately transitions into Maintenance Mode.
- **Minimum Full Support Period** Each minor version will receive full support for a minimum of three months from its release date before transitioning to maintenance mode. Major versions will also have a minimum of 3 months of full support.
- During Maintenance Mode, support will be limited to high-priority issues and showstopper bugs that impact critical functionality.

14.1.3 End of Support Timeline

End of Support (EOS) for each version is scheduled one year after the version's release date. After the EOS date, no further updates or bug fixes will be provided for that version, and customers will be encouraged to upgrade to a supported version to receive continued updates and support.

14.1.4 SQDB Releases Timeline

Release	Release Date	Maintenance Mode	End of Support
4.11	April 9th 2025	TBD	April 9th 2026
4.10	January 20th 2025	April 9th 2025	January 20th 2026
4.9	November 29th 2024	February 29th 2025	November 29th 2025
4.8	October 6th 2024	January 6th 2025	October 6th 2025
4.5	December 5th 2023	March 5th 2024	September 30th 2025
4.3	June 11th 2023	September 11th 2023	June 11th 2024

14.2 4.x Release Notes

14.2.1 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*

14.2.1.1 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

14.2.1.2 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.

14.2.1.3 SQreamDB Studio Updates and Improvements

SQream Studio version 5.6.0 has been released.

14.2.1.4 Known Issues

- Percentile is not supported for Window Functions.
- Performance degradation when using `VARCHAR` partition key in a Window Functions expression
- In SQreamDB minor versions 4.3.9 and 4.3.10, granting permissions through the Acceleration Studio might result in an error, even though the permission has been successfully granted.

14.2.1.5 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

14.2.1.6 Configuration Adjustments

► You may now configure the object access style and your endpoint URL with Virtual Private Cloud (VPC) when using AWS S3.

Visit *Amazon Web Services* to learn more about how and when you should use these two new parameters:

- `AwsEndpointOverride`
- `AwsObjectAccessStyle`

14.2.1.7 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.

14.2.1.8 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.

14.2.2 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*

14.2.2.1 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 5.3.1 • ODBC 4.4.4 • NodeJS • .NET 5.0.0 • Pysqream 5.1.0 (compatible with v4.5.13 or later) • Spark 5.0.0 • SQLoader As A Service 8.1 (compatible with v4.6.1 or later) • SQLoader As A Process 7.13 (compatible with v4.5.13 or later)

14.2.2.2 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

► The new swap_table_names utility function enables you to swap the names of two tables within a schema.

14.2.2.3 Known Issues

- Percentile is not supported for Window Functions

14.2.2.4 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

14.2.2.5 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.

14.2.2.6 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.

14.2.3 Release Notes 4.8

The 4.8 release notes were released on October 6th, 2024

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.8 resolved Issues*
- *Deprecations*
- *Upgrading to Version 4.8*

14.2.3.1 Compatibility Matrix

System Requirement	Details
Supported OS	RHEL 8.x
supported Nvidia driver	CUDA version 12.x
Storage version	51
Driver compatibility	• JDBC 5.4.0 • ODBC 4.4.4 • NodeJS 4.2.4 • .NET 5.0.0 • Pysqream 5.3.0 • SQLAlchemy 1.4 • Spark 5.0.0 • SQLoader As A Service 8.2

14.2.3.2 New Features and Enhancements

► Prepared statements, also known as parameterized queries, are a safer and more efficient way to execute SQL statements. They prevent SQL injection attacks by separating SQL code from data, and they can improve performance by reusing prepared statements. These are now supported by our Python and JDBC client drivers.

► PIVOT allows to convert row-level data into columnar representation. This technique is particularly useful when you need to summarize and visualize data. UNPIVOT does the opposite by transforming columnar data into rows. This operation is invaluable for scenarios where you wish to explore data in a more granular manner.

► Window funtion alias allows to specify a parameter within the window function definition. This eliminates the need to repeatedly input the same SQL code in queries that use multiple window functions with identical definitions.

► CONCAT function concatenates one or more strings, or concatenates one or more binary values.

14.2.3.3 Known Issues

Percentile is not supported for Window Functions

14.2.3.4 Version 4.8 resolved Issues

SQ No.	Description
SQ-12365	SQream CLI - Comment is not ignored as expected
SQ-17520	SQLoader - DELETE issue following CDC process

14.2.3.5 Deprecations

► Haskell CLI

Starting October 2024, support for the Haskell CLI will be discontinued, and it will be replaced by a JAVA CLI that is compatible with both SQreamDB and BLUE.

► 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.

14.2.3.6 Upgrading to Version 4.8

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>
```

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.

14.2.4 Release Notes 4.9

The 4.9 release notes were released on Novemebr 28th, 2024

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.9 resolved Issues*
- *Deprecations*
- *Upgrading to Version 4.9*

14.2.4.1 Compatibility Matrix

System Requirement	Details
Supported OS	RHEL 8.9
supported Nvidia driver	CUDA version 12.x
Storage version	57
Driver compatibility	• JDBC 5.4.2 • ODBC 4.4.4 • NodeJS 4.2.4 • .NET 5.0.0 • Pysqream 5.3.0 • SQLAlchemy 1.4 • Spark 5.0.0 • SQLoader As A Service 8.3

14.2.4.2 New Features and Enhancements

This release does not include any new features or enhancements

14.2.4.3 Known Issues

Percentile is not supported for Window Functions

14.2.4.4 Version 4.9 resolved Issues

SQ No.	Description
SQ-19055	Illegal memory access was encountered error
SQ-19053	Workers connectivity issues
SQ-19051	Compression related metadata issue
SQ-18745	Cannot grant select to a role on a table
SQ-16877	Query run time is longer than expected

14.2.4.5 Deprecations

► Haskell CLI

Starting October 2024, support for the Haskell CLI is discontinued, and it is replaced by the *Multi Platform CLI* that is compatible with the Haskell CLI with the added value of `Table-View` and cross platform compatability.

► 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.

14.2.4.6 Upgrading to Version 4.9

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>
```

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.

14.2.5 Release Notes 4.10

The 4.10 release notes were released on January 20th, 2025

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.10 resolved Issues*
- *Deprecations*
- *Upgrading to Version 4.10*

14.2.5.1 Compatibility Matrix

System Requirement	Details
Supported OS	RHEL 8.9
supported Nvidia driver	CUDA version 12.3.2
Storage version	58
Driver compatibility	• JDBC 5.4.2 • ODBC 4.4.4 • NodeJS 4.2.4 • .NET 5.0.0 • Pysqream 5.3.0 • SQLAlchemy 1.4 • Spark 5.0.0 • SQLoader As A Service 8.3.1 • Java CLI 2.2

14.2.5.2 New Features and Enhancements

► Alter Default Permissions is now enhanced to include support for `USER DEFINED FUNCTIONS`.

► Enahnced Regular Expresseion (RegEx) support.

► Python version for *Python User Defined Functions* has been upgraded to Python 3.11.

14.2.5.3 Known Issues

Percentile is not supported for Window Functions

14.2.5.4 Version 4.10 resolved Issues

SQ No.	Description
SQ-18800	Running Create Table As Select from a table with delete predicates causes worker abnormal behavior
SQ-18797	Worker failure leaves orphan locks
SQ-18555	Enhance set parameter command efficiency
SQ-18504	Access Control Permissions - Functions
SQ-18253	A network insert query fails
SQ-16436	Right function error
SQ-14398	Improve orphan locks handling

14.2.5.5 Deprecations

► Haskell CLI

Starting October 2024, support for the Haskell CLI is discontinued, and it is replaced by the *Multi Platform CLI* that is compatible with the Haskell CLI with the added value of `Table-View` and cross platform compatability.

► 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.

► DateTime2 Alias

Starting April 2025, The alias `DateTime2` for the `DateTime` Data Type is being deprecated to simplify our data model and make way for the introduction of a new data type named `DateTime2`. While `DateTime2` will remain functional as an alias until April 2025, we recommend updating your code to use `DateTime` directly to ensure compatibility.

14.2.5.6 Upgrading to Version 4.10

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>
```

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.

14.2.6 Release Notes 4.11

The 4.11 release notes were released on April 9th, 2025

- *Compatibility Matrix*
- *New Features and Enhancements*
- *Known Issues*
- *Version 4.11 resolved Issues*
- *Deprecation*
- *Upgrading to Version 4.11*

14.2.6.1 Compatibility Matrix

System Requirement	Details
Supported OS	RHEL 8.9
supported Nvidia driver	CUDA version 12.3.2
Storage version	59

14.2.6.2 New Features and Enhancements

► SQDB's new `DATETIME2` data type delivers nanosecond precision and timezone notation, for reliable and accurate time-based data.

► Array decompress to run on GPU (Performance improvement).

14.2.6.3 Known Issues

Percentile is not supported for Window Functions

14.2.6.4 Version 4.11 resolved Issues

SQ No.	Description
SQ-18621	`DELETE` statement causes Worker stability issue
SQ-19275	Compression issue
SQ-19415	Problem granting column DDL privilege to new roles
SQ-19485	Issue using `COPY FROM` JSON file containing `ARRAY` data
SQ-19534	Issue with `ARRAY` containing large `TEXT`
SQ-19541	Issue using `DISTINCT` clause on `ARRAY`
SQ-19571	`LZ4` is not permitted for `FLOAT` cell of `ARRAY`
SQ-19635	`DICT` Compression issue
SQ-19869	`STDDEV` functions slow compilation time
SQ-19879	Issue with Saved Queries Default permissions during database creation

14.2.6.5 Deprecation

► Haskell CLI

Starting October 2024, support for the Haskell CLI is discontinued, and it is replaced by the *Multi Platform CLI* that is compatible with the Haskell CLI with the added value of `Table-View` and cross platform compatability.

► CentOS Linux 7.x

- As of June 2024, CentOS Linux 7.x has reached its End of Life and is no longer supported by SQreamDB.

► DateTime2 Alias

As of April 2025, the alias `DateTime2` for the `DateTime` Data Type has been deprecated to simplify our data model and make way for the introduction of a new data type named `DateTime2`.

14.2.6.6 Upgrading to Version 4.11

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>
```

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.

CHAPTER

FIFTEEN

TROUBLESHOOTING

The **Troubleshooting** page describes solutions to the following issues:

15.1 Remedying Slow Queries

This page describes how to troubleshoot the causes of slow queries.

Slow queries may be the result of various factors, including inefficient query practices, suboptimal table designs, or issues with system resources. If you're experiencing sluggish query performance, it's essential to diagnose and address the underlying causes promptly.

Step 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.

Step 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.

Step 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.

Step 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.

Step 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.

Step 6: Check if there are long running statements

1. Identify any currently running statements (use `SELECT show_server_status()` to monitor)
2. If there are more statements than available resources, some statements may be in an `In queue` mode.
3. If there is a statement that has been running for too long and is blocking the queue, consider stopping it (use `SELECT stop_statement(<statement id>)`).

If the statement does not stop correctly, contact SQream Support.

If there are no long running statements or this does not help, continue to step 7.

Step 7: Check if there are active locks

1. Use `SELECT show_locks()` to list any outstanding locks.
2. If a statement is locking some objects, consider waiting for that statement to end or stop it.
3. If after a statement is completed the locks don't free up, refer to the *Concurrency and Locks* guide.

If performance does not improve after the locks are released, continue to step 8.

Step 8: Check free memory across hosts

1. Check free memory across the hosts by running `$ free -th` from the terminal.
2. If the machine has less than 5% free memory, consider **lowering** the `limitQueryMemoryGB` and `spoolMemoryGB` settings. Refer to the *Spooling Configuration* guide.
3. If the machine has a lot of free memory, consider **increasing** the `limitQueryMemoryGB` and `spoolMemoryGB` settings.

If performance does not improve, contact SQream Support.

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 *Foreign 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 = '|',
```

(continues on next page)

(continued from previous page)

```
    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.