SQream Connector Native Java 1.0.0

SQream Technologies

2018-06-28

Table of Contents

The SQream Native Java Connector - Overview

- This guide describes the implementation of the SQream Native Java connector and is designed for SQream DB administrators and developers.
- The SQream Native Java connector gives structures to initialize a connection, run SQL queries through the connection (statements), and enables network streaming (insert, select).
- SQream connector protocol version: 5

1. API Reference

To use the functions include the connector jar and import: "import com.sqream.connector;

1.1. Connection

Table 1. Initializing and closing connections

Function	Description

`ConnectionHandle ('ip', port, 'database', 'username', 'password', useSsl)`	`Creates a connection handle with all the connection informations and opens a socket to the ip, this object can be held to keep a connection alive to the sqream server.`
	`ip - IP address as a string.`
	`port - port number SQream is listening on.`
	`database - name of database to connect to.`
	`username, password - connection credentials. default is 'sqream' for both.`
	`useSsl - True / false. If true, connect to SQream using SSL port.`

`ConnectionHandle.connect()`	`Connects the handle to the sqream, accessing its database.`
`ConnectionHandle.close()`	`Closes the connection handle.`

1.2. Statement

Table 2. Statement execution

Function	Description
`StatementHandle(ConnectionHandle, 'statement')`	`Creates a statement object which purpose is to operate the different messages of the protocol in order to execute a query and its different functionalities.`
`StatementHandle.prepare()`	`Prepares the statement of the current StatementHandle.`
`StatementHandle.execute()`	`Executes the statement of the current StatementHandle. Comes after prepare().`
`StatementHandle.nextRow()`	`On an insert query - start setting the next row for insertion. SQream does not support partial inserts. On a select query - move to next row index to start selecting items from various columns using get() functions`

Function	Description
StatementHandle.close()	Closes the StatementHandle.

1.3. High level protocol functions

Table 3. Retrieve results from a select query by column index

Function	Description
isNull(int col_id)	Check whether the value in column index col_id is a null
getBool(int col_id)	Get Boolean value from column index col_id at the current row
getUbyte(int col_id)	Get UByte value from column index col_id at the current row
getShort(int col_id)	Get Short value from column index col_id at the current row
getInt(int col_id)	Get Int value from column index col_id at the current row
getLong(int col_id)	Get Long value from column index col_id at the current row
getFloat(int col_id)	Get Float value from column index col_id at the current row
getDouble(int col_id)	Get Double value from column index col_id at the current row
getDate(int col_id)	Get Date value from column index col_id at the current row
getDatetime(int col_id)	Get Datetime value from column index col_id at the current row
getVarchar(int col_id)	Get Varchar value from column index col_id at the current row
getNvarchar(int col_id)	Get Nvarchar value from column index col_id at the current row

Table 4. Retrieve results from a select query by column name

Function	Description
isNull(String col_name)	Check whether the value in column named col_name is a null
getBool(String col_name)	Get Boolean value from column named col_name at the current row
getUbyte(String col_name)	Get UByte value from column named col_name at the current row
getShort(String col_name)	Get Short value from column named col_name at the current row
getInt(String col_name)	Get Int value from column named col_name at the current row
getLong(String col_name)	Get Long value from column named col_name at the current row
getFloat(String col_name)	Get Float value from column named col_name at the current row
getDouble(String col_name)	Get Double value from column named col_name at the current row
getDate(String col_name)	Get Date value from column named col_name at the current row
getDatetime(String col_name)	Get Datetime value from column named col_name at the current row
getVarchar(String col_name)	Get Varchar value from column named col_name at the current row
getNvarchar(String col_name)	Get Nvarchar value from column named col_name at the current row

Table 5. Set data by index following a bulk insert query

Function	Description
setNull(int col)	Set column at index col in the current row to null
setBool(int col, boolean val)	Set column at index col of type Boolean in the current row
setUbyte(int col, byte val)	Set column at index col of type UByte in the current row - unsignted bytes only
setShort(int col, short val)	Set column at index col of type Short in the current row
setInt(int col, int val)	Set column at index col of type Int in the current row
setLong(int col, long val)	Set column at index col of type Long in the current row
setFloat(int col, float val)	Set column at index col of type Float in the current row
setDouble(int col, double val)	Set column at index col of type Double in the current row
setDate(int col, Date val)	Set column at index col of type Date in the current row
setDatetime(int col, Timestamp val)	Set column at index col of type Datetime in the current row
setVarchar(int col, String val)	Set column at index col of type Varchar in the current row
setNvarchar(int col, String val)	Set column at index col of type Nvarchar in the current row

2. Code Samples

2.1. Import and establish a connection

Example

```
import com.sqream.connector;

class Test {

    // Connection parameters: IP, Port, Database, Username, Password
    ConnectionHandle Client = new ConnectionHandle ("127.0.0.1", 5000, "master",
"sqream", "sqream", false);
    Client = Client.connect();
```

2.2. Run a query - Create a table

Example

```
    String statement = "create or replace table table_name (int_column int)";
    StatementHandle stmt = new StatementHandle(Client, statement);
    stmt.prepare();
    stmt.execute();
    stmt.close();
```

2.3. Run a query - Insert values into table

Example

```
    String statement = "insert into table_name(int_column) values (5), (6), (7), (8)";
    StatementHandle stmt = new StatementHandle(Client, statement);
    stmt.prepare();
    stmt.execute();
    stmt.close();
```

2.4. Run a query - Get column values from table

Example

```
    // Retrieve data
    String statement = "select int_column from table_name";
    StatementHandle stmt = new StatementHandle(Client, statement);
    stmt.prepare();
    stmt.execute();

    // Pull out the actual data
    while (stmt.nextRow())
        System.out.println("Number recieved: " + stmt.getInt(1));

    stmt.close();


    // After running all statements
    //  ---------------------------
    client.close()
}
```

2.5. Run a query - Use bulk insert to insert large amounts of data in a programmatic way

Example

```
/* Example of classic Set data loop, using network streaming (also called Network
Insert) */
// here we create the according table by executing a
// "create or replace table table_name (int_column int, varchar_column varchar(10))"
statement

int[] row1 = {1,2,3};
String[] row2 = {"s1","s2","s3"};
int length_of_arrays = 3;

// each interrogation symbol represent a column to which the network insertion can
push
String statement = "insert into table_name(int_column, varchar_column) values(?, ?)";
StatementHandle stmt = new StatementHandle(Client, statement);
stmt.execute();
for (int idx = 0; idx < length_of_arrays; idx ++) {
    stmt.setInt(1, row1[idx])      // put a value at column 1 of the table
    stmt.setVarchar(2, row2[idx])  // put a value at column 2 of the table
}

stmt.close();
client.close();
```

2.6. Run a query - Starting and finishing

Example

```
/*  Initialization - Termination Example     */
import com.sqream.connector;

    class Query {
        // arg types are: string, integer, string, string, string, boolean, integer
        ConnectionHandle Client = new ConnectionHandle ('127.0.0.1', 5000, 'master',
'sqream', 'sqream', false);
        Client = Client.connect();
        String statement = "sql statement";
        StatementHandle stmt = new StatementHandle(Client, statement);
        .
        .

        // closes the statement (to do after execute + necessary fetch/put to close
the
        // statement and be able to open another one through prepare())
        stmt.close();

        // closes the connection completely, destroying
         the socket, a call to "connect(..)"
        // needs to be done do continue
        client.close();
    }
```

Copyright