SQream Connector Native C++ 1.0.0

SQream Technologies

2018-06-28

Table of Contents

The SQream Native C++ Connector - Overview

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

1. API Reference

To use the functions include the connector main .h file ("SQream-cpp-connector.h"). This will give the application access to the interfaces. It is also needed later to possess the .so of the library in a way that the using program can compile and link to it.

The connector functions are situated in the name space "sqream::driver" and for the rest of the documentation all the functions are assumed to be called using this name space.

1.1. Connection

Table 1. Initializing and closing connections

Function	Description
`driver()`	The constructor creating the handle to the connection
`connect(const std::string &ipv4,int port,bool ssl,const std::string &username,const std::string &password,const std::string &database)`	Connects the handle to the sqream, accessing its database ipv4 - IP address as a string. port - port number SQream is listening on. ssl - True / false. If true, connect to SQream using SSL port. username, password - connection credentials. default is 'sqream' for both. database - name of database to connect to.
`disconnect()`	Close the connection handle and reset the driver

1.2. Statement

Table 2. Statement execution

Function	Description
`new_query(const std::string &sql_query)`	Starts a new statement resetting any statement related datas and follows it by Prepare
`execute_query()`	Executes the statement of the current statement. Comes after prepare().

Function	Description
next_query_row()	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
finish_query()	Closes the statement

1.3. Helper functions

Table 3. Additional functions in order to accomplish a broader amount of operations or give access to some internal numbers

Function	Description
new_query_execute(driver *drv,std::string sql_query)	Operates the protocol until the statement is executed, this function is a shortcut and set_types or get_types functions can be executed right after
void run_direct_query(driver *drv,std::string sql_query)	Executes a query from start to end (closes the statement). This function is a shortcut for when you don't need to see any input/output like a DDL type of query ("create table" etc)
std::vector<column> get_metadata(driver *drv)	Returns the metadata of the current statement if available (after a Prepare) or else an empty vector

1.4. High level protocol functions

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

Function	Description
is_null(size_t col_id)	Check whether the value in column index col_id is a null
get_bool(size_t col_id)	Get bool value from column index col_id at the current row
get_ubyte(size_t col_id)	Get uint8_t value from column index col_id at the current row
get_short(size_t col_id)	Get int16_t value from column index col_id at the current row
get_int(size_t col_id)	Get int32_t value from column index col_id at the current row
get_long(size_t col_id)	Get int64_t value from column index col_id at the current row
get_float(size_t col_id)	Get float value from column index col_id at the current row
get_double(size_t col_id)	Get double value from column index col_id at the current row
get_date(size_t col_id)	Get uint32_t value from column index col_id at the current row
get_datetime(size_t col_id)	Get uint64_t value from column index col_id at the current row
get_varchar(size_t col_id)	Get string value from column index col_id at the current row
get_nvarchar(size_t col_id)	Get string value from column index col_id at the current row

Table 5. Retrieve results from a select query by column name

Function	Description
is_null(string col_name)	Check whether the value in column named col_name is a null
get_bool(string col_name)	Get Boolean value from column named col_name at the current row
get_ubyte(String col_name)	Get UByte value from column named col_name at the current row
get_short(string col_name)	Get Short value from column named col_name at the current row
get_int(string col_name)	Get Int value from column named col_name at the current row
get_long(string col_name)	Get Long value from column named col_name at the current row
get_float(string col_name)	Get Float value from column named col_name at the current row
get_double(string col_name)	Get Double value from column named col_name at the current row
get_date(string col_name)	Get Date value from column named col_name at the current row
get_datetime(string col_name)	Get Datetime value from column named col_name at the current row
get_varchar(string col_name)	Get Varchar value from column named col_name at the current row
get_nvarchar(string col_name)	Get Nvarchar value from column named col_name at the current row

Table 6. Set data by index following a bulk insert query

Function	Description
set_null(size_t col)	Set column at index col in the current row to null
set_bool(size_t col, bool val)	Set column at index col of type Boolean in the current row
set_ubyte(size_t col, uint8_t val)	Set column at index col of type UByte in the current row - unsignted bytes only
set_short(size_t col, uint16_t val)	Set column at index col of type Short in the current row
set_int(size_t col, uint32_t val)	Set column at index col of type Int in the current row
set_long(size_t col, uint64_t val)	Set column at index col of type Long in the current row
set_float(size_t col, float val)	Set column at index col of type Float in the current row
set_double(size_t col, double val)	Set column at index col of type Double in the current row
set_date(size_t col, uint32_t val)	Set column at index col of type Date in the current row
set_datetime(size_t col, uint64_t val)	Set column at index col of type Datetime in the current row
set_varchar(size_t col, string val)	Set column at index col of type Varchar in the current row
set_nvarchar(size_t 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

```
#include "SQream-cpp-connector.h"

// Connection parameters: IP, Port, Database, Username, Password
sqream::driver sqc;
sqc.connect("127.0.0.1", 5000, false, "sqream", "sqream", "master");
```

2.2. Run a query - Create a table

Example

```
string statement = "create or replace table table_name (int_column int)";
new_query(&sqc, statement);
sqc.execute_query();
sqc.finish_query();

OR

run_direct_query(&sqc, "create or replace table table_name (int_column int)");
```

2.3. Run a query - Insert values into table

Example

```
string statement = "insert into table_name(int_column) values (5), (6), (7), (8)";
new_query(&sqc, statement);
sqc.execute_query();
sqc.finish_query();

OR

run_direct_query(&sqc, "insert into table_name(int_column) values (5), (6), (7), (8)
");
```

2.4. Run a query - Get column values from table

Example

```
// Retrieve data
string statement = "select int_column from table_name";
new_query(&sqc, statement);
sqc.execute_query();

// Pull out the actual data
while (sqc.next_query_row())
    printf("Number recieved: %d" + sqc.get_int(0));
sqc.finish_query();

OR

new_query_execute(&sqc, "select int_column from table_name");
while (sqc.next_query_row())
    printf("Number recieved: %d" + sqc.get_int(0));
sqc.finish_query();
```

2.5. Run a query - Use bulk insert to insert large amount 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 represents a column to which the network insertion can
push
string statement = "insert into table_name(int_column, varchar_column) values(?, ?)";
new_query(&sqc, statement);
sqc.execute_query();
for (int idx = 0; idx < length_of_arrays; idx ++) {
    sqc.set_int(0, row1[idx])      // put a value at column 0 of the table
    sqc.set_varchar(1, row2[idx])  // put a value at column 1 of the table

    sqc.next_query_row();
}

sqc.finish_query();
```

2.6. Run a query - Starting and finishing

Example

```
/*  Initialization - Termination Example     */
#include "SQream-cpp-connector.h"

void Query() {
    // Connection parameters: IP, Port, Database, Username, Password
    sqream::driver sqc;
    sqc.connect("127.0.0.1", 5000, false, "sqream", "sqream", "master");
    string statement = "sql statement";
    new_query_execute(&sqc, statement);
    .
    .
    .
    // closes the statement (to do after execute + necessary fetch/put to close the
    // statement and be able to open another one through prepare())
    sqc.finish_query();

    // closes the connection completely, destroying the socket, a call to
"connect(..)"
        // needs to be done do continue using this "driver sqc" object
    sqc.disconnect();
}
```

Copyright