SQream Connector Native C++

SQream Technologies

Version 1.2.0

Table of Contents

SQream Connector Native C++

Version 1.2.0

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: 7

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.

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),std::string &service)`	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.

Function	Description
	username, password - connection credentials. default is 'sqream' for both. database - name of database to connect to. service - name of the service to connect to.
disconnect()	Close the connection handle and reset the driver

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. Needs to be called after new_query () (This advance the SQL query from Prepare -> Execute).
next_query_row (const size_t min_put_size=CONSTS::MIN_PUT_SIZE)	On an insert query - start setting the next row for insertion. SQream does not support partial inserts. If min_put_size is set, the connector will flush and put the rows if the buffer size is bigger than this value. If the function is called without argument a default value of 67108864 bytes (64 MiB) is set for comparison with the buffer size. On a select query - move to next row index to start selecting items from various columns using get() functions, the min_put_size argument has no influence on select.
finish_query()	Closes the statement

Helper functions

Table 3. Additional functions in order to accomplish a broader amount of operations or give access to some internal numbers. Their namespace is "sqream::" only

Function	Description
`new_query_execute (sqream::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 (sqream::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<sqream::column> get_metadata(driver *drv)`	Returns the metadata of the current statement if available (after a Prepare) or else an empty vector
`uint32_t retrieve_statement_id(sqream::driver *drv)`	Returns the current statement id, will throw an error if no statement is executed
`sqream::CONSTS::statement_type retrieve_statement_type(sqream::driver *drv)`	Returns the enum representing the type of statement currently executing. A statement can be a select, an insert or a DML (direct)
`uint32_t date(int32_t year,int32_t month,int32_t day)`	Convert date to sqream date (Julian day number, stored in an uint)
`uint64_t datetime(int32_t year,int32_t month,int32_t day,int32_t hour,int32_t minute,int32_t second,int32_t millisecond)`	Convert datetime to sqream datetime (Julian day number, stored in an uint in the high 32 bits and Julian day fraction stored in an uint in the low 32 bits)
`sqream::date_t make_date (uint32_t date)`	Convert sqream date to sqream::date_t structure
`sqream::datetime_t make_datetime(uint64_t datetime)`	Convert sqream datetime to sqream::datetime_t structure

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`	Get Long value from column named col_name

Function	Description
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

Code Samples

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");
```

Code Samples

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");
```

Run a query - Create a table

Example

```
string  statement  =  "create or replace table table_name (int_
column int)";
sqc.new_query(statement);
sqc.execute_query();
sqc.finish_query();

OR

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

Run a query - Insert values into table

Example

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

```
sqc.finish_query();

OR

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

Run a query - Get column values from table

Example

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

// Pull out the actual data row by row
while  (sqc.next_query_row())
       cout  <<  "Received: "  <<  sqc.get_int(0)  <<  endl;
sqc.finish_query();

OR

new_query_execute(&sqc,  "select int_column from table_name");
while  (sqc.next_query_row())
       cout  <<  "Received: "  <<  sqc.get_int(0)  <<  endl;
sqc.finish_query();
```

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)
int  len  =  3;
int  row1[len]  =  {1,2,3};
string  row2[len]  =  {"s1","s2","s3"};


string  statement  =  "create or replace table table_name (int_
column int, varchar_column varchar(10))";
run_direct_query(&sqc,  statement);

// each interrogation symbol represents a column to which the
network insertion can push
statement  =  "insert into table_name(int_column, varchar_column)
values(?, ?)";
sqc.new_query(statement);
sqc.execute_query();
```

```
for (int idx = 0; idx < len; 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();
```

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);

    //.
        //. Do something
        //.

    // 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();
}
```