

SQream Connector Native Python 2.1.1

SQream Technologies

2018-11-28

Table of Contents

The SQream Native Python Connector - Overview.....	1
1. API Reference	1
1.1. Initialization - Termination	1
1.2. Statement	1
1.3. High level protocol functions.....	1
2. Code Samples.....	3
2.1. Import and establish a connection, run a query.....	3
2.2. Example of classic Get data loop.....	5
2.3. Example of classic Set data loop, using network streaming (also called Network Insert)	5
Copyright	6

The SQream Native Python Connector - Overview

- This guide describes the implementation of the SQream Native Python connector and is designed for SQream DB administrators and developers.
- The SQream Native Python connector gives structures to initialize a connection, run SQL queries through the connection (statements), and enables network streaming (insert, select).
- SQream connector protocol version: 6

1. API Reference

All functions are accessed through the Connector class imported from `SQream_python_connector.py`

1.1. Initialization - Termination

```
import SQream_python_connector
con = SQream_python_connector.Connector()

# arg types are: string, integer, string, string, string, boolean, integer
con.connect(ip, port, database, username, password, clustered, timeout)

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

# closes the connection completely, destructing the socket, a call to "connect(..)"
needs to be done do continue
con.close_connection()
```

1.2. Statement

```
con.prepare(statement) #string of the query to run
con.execute()

# if the statement is an insert it produces a put and for select it produces a fetch,
rows are incremented through that function (see Usage example)
con.next_row()
```

1.3. High level protocol functions

Table 1. Retrieve results from a select query by column index

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

Table 2. 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 3. Set data by index following a bulk insert query

Function	Description
set_null(int col)	Set column at index col in the current row to null
set_bool(int col, boolean val)	Set column at index col of type Boolean in the current row
set_ubyte(int col, byte val)	Set column at index col of type UByte in the current row - unsigned bytes only

Function	Description
<code>set_short(int col, short val)</code>	Set column at index col of type Short in the current row
<code>set_int(int col, int val)</code>	Set column at index col of type Int in the current row
<code>set_long(int col, long val)</code>	Set column at index col of type Long in the current row
<code>set_float(int col, float val)</code>	Set column at index col of type Float in the current row
<code>set_double(int col, double val)</code>	Set column at index col of type Double in the current row
<code>set_date(int col, Date val)</code>	Set column at index col of type Date in the current row
<code>set_datetime(int col, Timestamp val)</code>	Set column at index col of type Datetime in the current row
<code>set_varchar(int col, String val)</code>	Set column at index col of type Varchar in the current row
<code>set_nvarchar(int col, String val)</code>	Set column at index col of type Nvarchar in the current row

2. Code Samples

2.1. Import and establish a connection, run a query

Example

```
## Import and establish a connection
# -----
import SQream_python_connector

# version information
print SQream_python_connector.version_info()

con = SQream_python_connector.Connector()
# Connection parameters: IP, Port, Database, Username, Password, Clustered,
# Timeout(sec)
sqream_connection_params = '127.0.0.1', 5000, 'master', 'sqream', 'sqream', False, 30
con.connect(*sqream_connection_params)

## Run queries using the API
# -----
# Create a table
statement = 'create or replace table table_name (int_column int)'
con.prepare_(statement)
con.execute()
con.close()

# Insert sample data
statement = 'insert into table_name(int_column) values (5), (6)'
con.prepare(statement)
con.execute()
con.close()

# Retrieve data
statement = 'select int_column from table_name'
con.prepare(statement)
con.execute()
con.next_row()

# Pull out the actual data
first_row_int = con.get_int(1)
con.next_row()
second_row_int = con.get_int(1)
con.next_row()
print (first_row_int, second_row_int)
con.close()

## After running all statements
# -----
con.close_connection()
```

2.2. Example of classic Get data loop

Example

```
# Here we create the according table by
# executing a "create or replace table table_name (int_column int, varchar_column
varchar(10))" statement

row1 = []
row2 = []

statement = 'select int_column, varchar_column from table_name'
con.prepare(statement)
con.execute()

while con.next_row():
    row1.append(con.get_int(1))
    row2.append(con.get_string(2))

con.close()
con.close_connection()
```

2.3. Example of classic Set data loop, using network streaming (also called Network Insert)

Example

```
# here we create the according table by executing a
# "create or replace table table_name (int_column int, varchar_column varchar(10))"
statement

row1 = [1,2,3]
row2 = ["s1","s2","s3"]
length_of_arrays = 3

# each interrogation symbol represent a column to which the network insertion can push
statement = 'insert into table_name(int_column, varchar_column) values(?, ?)'
con.prepare(statement)
con.execute()

for idx in range(length_of_arrays):
    con.set_int(1, row1[idx]) # we put a value at column 1 of the table
    con.set_varchar(2, row2[idx]) # we put a value at column 2 of the table

con.close()
con.close_connection()
```

Copyright

Copyright © 2010-2018. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchant- ability or fitness for a particular purpose.

We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document.

This document may not be reproduced in any form, for any purpose, without our prior written permission.