SQream DB Version 2019.1.2 - Administrator Guide

SQream Team

2019-03-27 | Version 2019.1.2

Table of Contents

Overview

This guide is intended for SQream DB administrators.

The guide will go through the DBA main tasks, as well as describing some of the best practices in SQream DB.

This guide is a complementary guide to the SQL Reference.

For further support please contact support@sqreamtech.com or your account manager.

1. Administration Guide

This document is mostly a conceptual overview and recommendations on best practices regarding SQream internal behavior. For SQL syntax and supported features, please refer to the SQL Reference Guide.

All keywords in are **case insensitive**.

2. Concepts

This section describes SQream's database concepts

2.1. SQream DB Daemon

In SQream DB, the `sqreamd` or SQream daemon is the server that deals with most of the operations against a certain GPU. Certain installations may have several `sqreamd` running, each with their own GPU ID.

A `sqreamd` may run with a specific storage cluster, or be teamed up with others, sharing a shared cluster.

2.2. Storage Cluster

A SQream DB Storage Cluster is a collection of all stored objects:

- Databases
- Schemas
- Tables
- Columns
- External Tables
- User Defined Functions
- Roles

A server instance can only run against a single storage cluster at one time. Clusters *can be changed*, but require a restart of the daemon.

A cluster will be created on installation, and shouldn't require any intervention during normal operation

2.3. Databases

A storage cluster may have many databases residing in it.

When you create an applicative connection (from a client or JDBC/ODBC) – you connect to a single database.

A database can have many Schemas and Tables.

Create different databases for different use-cases

To view existing databases, query the Catalog (information schema).

2.4. Schemas

Schemas are part of the effective table name. The default schema is called `public`.

Use schemas to split up the data tables logically

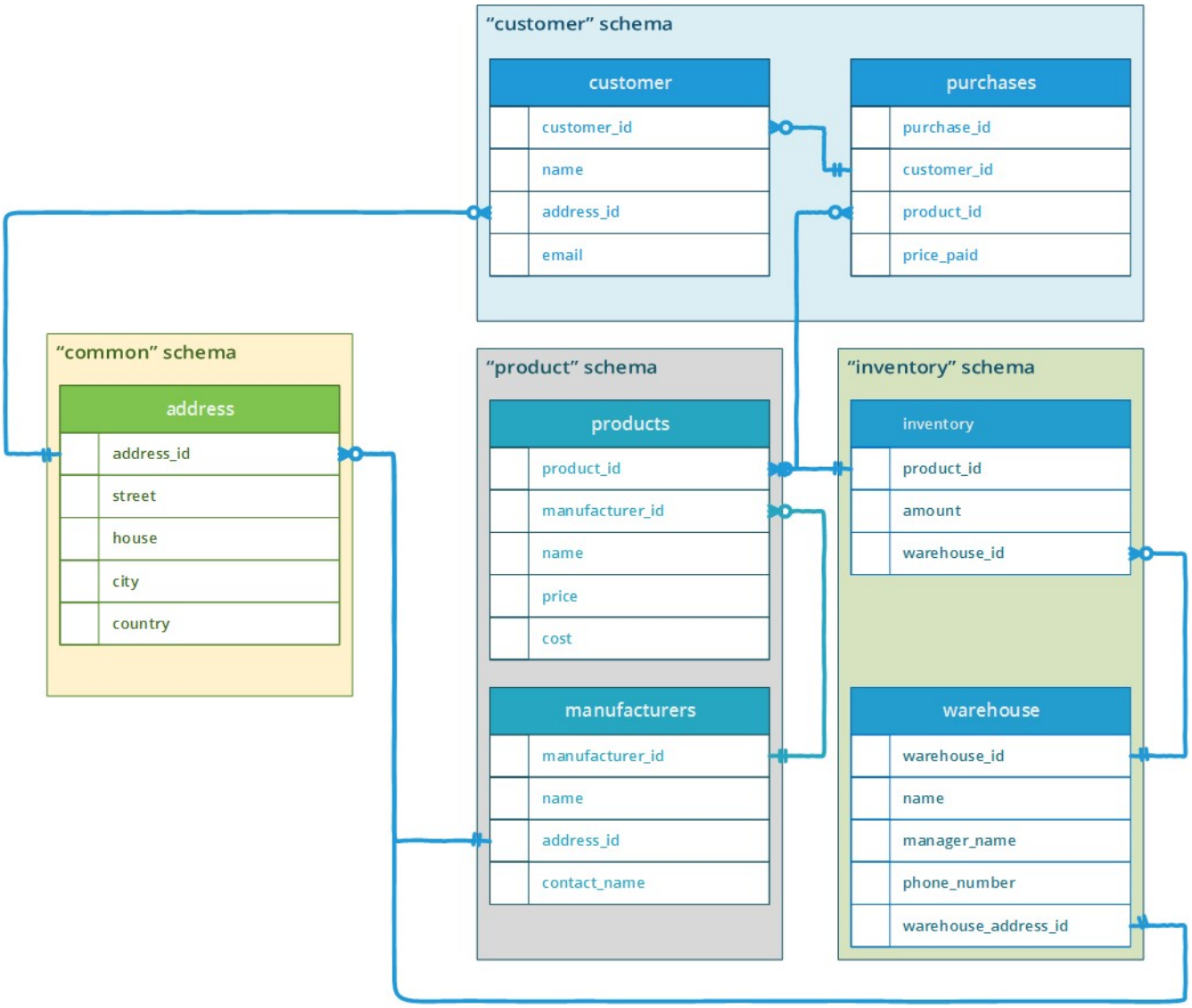


Figure 1. Example of schema usage

2.5. Tables

In SQream DB, a table is a set of data elements organized into horizontal rows and vertical columns. A table has a specified number of columns but can have any number of rows.

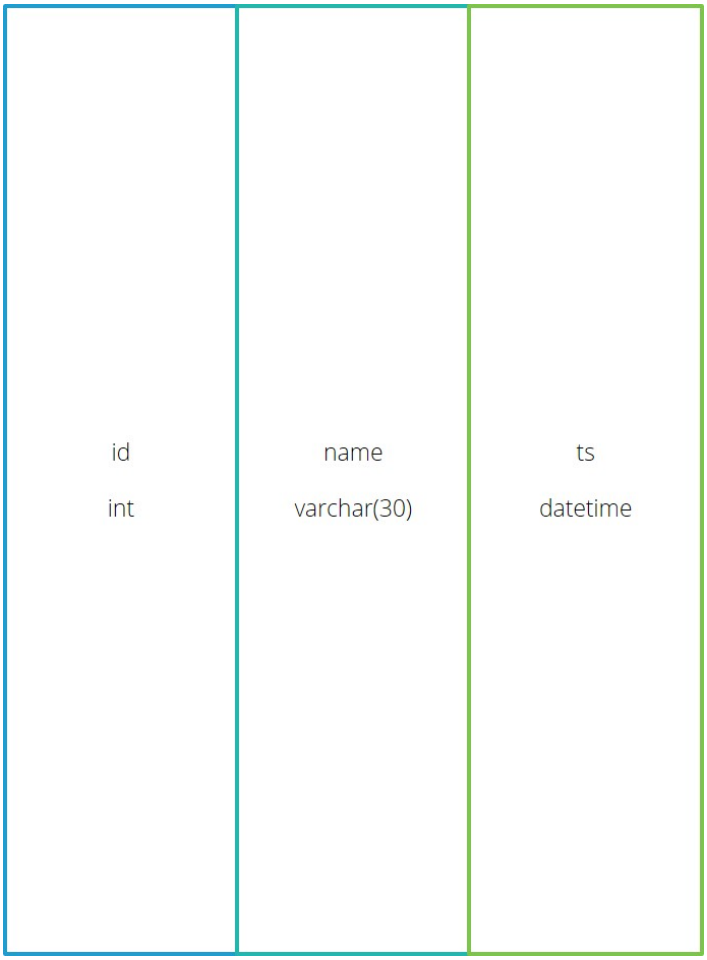


Figure 2. Logical table 'customers'

In a row-oriented database, would be stored as:
`1,John,1997-01-01 15:30:30` ; `2,Jane,1998-05-06 00:00:00;` ...

In a columnar database, would be stored as:
`1,2,3,4,5` ;
`John,Jane,Sam,David,Susan` ;
`1997-01-01 15:30:30,1998-05-06 00:00:00,1956-06-02 14:31:00,2006-12-25 14:40:00,1975-10-21 02:20:00`

Instead of pulling the entire row (all columns) every time, only select the columns you need.
SQream DB automatically removes columns not necessary for the calculations.

Example

```
SELECT name, MAX(ts) FROM customers GROUP BY name;
```

2.6. Columns

A column is the element that makes up a table.

Each column has a specific type and name, that do not change. By physically organizing data by columns, SQream DB is able to scan and aggregate data for individual columns faster, because less

data needs to be read. This makes columns highly suitable for analytical workloads.

2.7. External Tables

External tables are structured DDL that allow SQream to access data that which is stored outside the database in a none-SQream format, and to query it via SQL commands. Upon creation, the user should specify the external files format and location, and the needed table DDL. Once created, SQream will query the tables, as if they were regular tables.

2.8. User Defined Functions

User defined functions (UDF) are used by the DBA/BI/Data scientist to run custom made programs, written in Python, within SQream DB space. By using the UDF, the customers can extent the existing built-in operations in SQream, either as a row-by-row function, or as a utility function.

2.9. Chunks, Compression and Metadata

2.9.1. Chunks

SQream DB splits up columnar data into chunks. The chunk size will be the **minimal** bulk load into the GPU.

For better performance, SQream DB rearranges previously loaded data in new chunks based on the desired **chunk size**. While loading data into SQream DB, each bulk load (either `INSERT INTO` or `COPY`), will generate its own chunks (with sizes up to the chunk size).

The chunk size is a parameter at the cluster level that must be set before the first insert to the cluster. The default chunk size 1 million rows. The parameter can be set at the database level, before any tables are created. Ask your database administrator about setting the chunk size.

- SQream DB, in its own time, will **rechunk** those chunks into the desired chunk size. During the rechunk operation SQream will rearrange and recompress the data using the most appropriate compression type (according to the data type and data distribution).
- The chunk size has an influence on load/query time. Before tuning the parameter, consult your SQream account manager.

2.9.2. Compression specs

When `DEFAULT` compression spec (or no compression spec) is specified, each chunk may be compressed in a variety of different formats, based on the system's understanding. You may override the compression spec, but this is not recommended.

See Compression types in the SQL Manual for more information.

2.9.3. Metadata

SQream DB gathers and saves metadata information regarding the columns data at the chunk level during `COPY`. This information will serve the SQream optimizer while doing Data Skipping and other optimizations. This metadata is gathered automatically and transparently. It requires no user intervention.

2.10. Catalog (information schema)

The SQream DB catalog or information schema consists of views that contain information about all database objects. This provides access to database metadata, column types, tables and their row-counts, etc.

The catalog structure is specific to SQream DB.

2.10.1. Querying the SQream catalog

The catalog is available from any database, under the schema called `sqream_catalog`. You may query the schema as you would any other table in the system.

You can not perform any other operations on the catalog, like `INSERT`, `DELETE`, ...

Example

```
demo_db=> SELECT * from sqream_catalog.tables;
```

Table 1. Example result for a demo database

database_name	table_id	schema_name	table_name	row_count_valid	row_count	rechunker_ignore
demo_db	0	public	nation	1	25	0
demo_db	1	public	region	1	5	0
demo_db	2	public	part	1	20000000	0
demo_db	3	public	supplier	1	1000000	0
demo_db	4	public	partsupp	1	80000000	0
demo_db	5	public	customer	1	15000000	0
demo_db	6	public	orders	1	300000000	0
demo_db	7	public	lineitem	1	600037902	0

Example for identifying delete predicates on tables

```
demo_db=> select t.table_name,d.* from sqream_catalog.delete_predicates d
.> inner join sqream_catalog.tables t on
.> d.table_id=t.table_id;
```

2.10.2. Available catalog views

Database object catalog

Table 2. SQream catalog views

View name	Description
`sqream_catalog.databases`	All database objects in the current storage cluster

View name	Description
sqream_catalog.schemas	Schema objects in the database
sqream_catalog.tables	Table objects in the database
sqream_catalog.columns	Column objects in the current database
sqream_catalog.views	View objects in the database
sqream_catalog.external_tables	External table objects in the database
sqream_catalog.udf	User defined functions in the current database
sqream_catalog.catalog_tables	All catalog views available

Fine-grain storage catalog

View name	Description
sqream_catalog.extents	Extent storage objects in the database
sqream_catalog.chunks	Chunk storage objects in the database
sqream_catalog.delete_predicates	Logical delete predicates added to the compiler with a `DELETE` command

Role and permission catalog

View name	Description
sqream_catalog.roles	Roles (users) in the current databases
sqream_catalog.role_memberships	Roles membership
sqream_catalog.table_permissions	Tables and their assigned roles
sqream_catalog.database_permissions	Databases and their assigned roles
sqream_catalog.schema_permissions	Schemas and their assigned roles
sqream_catalog.permission_types	Permission types

Database objects

Databases view

Table 3. sqream_catalog.databases

Column name	Type	Description
database_id	varchar	Database ID
database_name	varchar	Database name
default_disk_chunk_size	bigint	Storage chunk size (in number of rows)
default_process_chunk_size	bigint	Process chunk size (in number of rows)
rechunk_size	bigint	Internal use
storage_subchunk_size	bigint	Internal use

Column name	Type	Description
compression_chunk_size_threshold	bigint	Internal use

Schemas view

Table 4. sqream_catalog.schemas

Column name	Type	Description
schema_id	varchar	Schema ID
schema_name	varchar	Schema name
schema_owner	varchar	Role who owns this schema
rechunker_ignore	bigint	Internal use

Tables view

Table 5. sqream_catalog.tables

Column name	Type	Description
database_name	varchar	Owning database name
table_id	varchar	Table ID
schema_name	varchar	Owning schema name
table_name	varchar	Table name
row_count_valid	bool	See warning below
row_count	bigint	Number of rows in the table
rechunker_ignore	int	Internal use

When `row_count_valid` is 0 (after a DELETE operation), the row count may be inaccurate. To get the accurate row-count, run

```
SELECT COUNT(column) FROM table;
```

Columns view

Table 6. sqream_catalog.columns

Column name	Type	Description
database_name	varchar	Owning database name
schema_name	varchar	Owning schema name
table_id	varchar	Owning table ID
table_name	varchar	Owning table name
column_id	int	Column ID
column_name	varchar	The column name
type_name	varchar	Column type

Column name	Type	Description
column_size	bigint	Column data size in bytes
has_default	int	Indicates whether or not the column has a default
default_value	varchar	Indicates the default column value
compression_strategy	varchar	User-overridden compression strategy
created	varchar	Creation date
altered	varchar	Last alter date

Views view

Table 7. sqream_catalog.views

Column name	Type	Description
view_id	varchar	View ID
view_schema	varchar	Owning schema name
view_name	varchar	The view name
view_data	varchar	Internal use
view_query_text	varchar	Full statement test that created this view

External tables view

Table 8. sqream_catalog.external_tables

Column name	Type	Description
database_name	varchar	Owning database name
table_id	varchar	External table ID
schema_name	varchar	Owning schema name
table_name	varchar	External table name
format	int	0=CSV, 1=Parquet
created	varchar	Creation data as a string

User defined functions

Table 9. sqream_catalog.udf

Column name	Type	Description
database_name	varchar	Owning database name
function_id	varchar	The function ID
function_name	varchar	The function name

Fine-grain storage

Extent view

Table 10. sqream_catalog.extents

Column name	Type	Description
`database name`	`varchar`	Owning database name
`table_id`	`varchar`	Owning table ID
`column_id`	`bigint`	Owning column ID
`extent_id`	`bigint`	The extent ID
`size`	`bigint`	Size of the extent in MB
`path`	`varchar`	Full path to the extent file on disk

Chunks view

Table 11. sqream_catalog.chunks

Column name	Type	Description
`database name`	`varchar`	Owning database name
`table_id`	`varchar`	Owning table ID
`chunk_id`	`bigint`	The chunk ID
`rows_num`	`bigint`	The amount of rows in this specific chunk
`deletion_status`	`bigint`	This chunk's deletion mark. 0 means keep, 1 means chunk needs partial deletion, 2 means delete entire chunk.

Delete predicates view

Table 12. sqream_catalog.delete_predicates

Column name	Type	Description
`database name`	`varchar`	Owning database name
`table_id`	`varchar`	Owning table ID
`max_chunk_id`	`bigint`	The highest chunk_id seen during the DELETE time
`delete_predicate`	`varchar`	The predicates added by the compiler (one predicate-statement per row in this view)

Role and permission catalog

Role view

Table 13. sqream_catalog.roles

Column name	Type	Description
role_id	bigint	The role ID
name	varchar	Role name
superuser	bool	1 for superuser, 0 otherwise
login	bool	1 if the role has login permission, 0 otherwise
has_password	bool	Does this role have a password?
can_create_function	bool	Does this role have the permissions to create/revoke user defined functions?

Role membership view

Table 14. sqream_catalog.role_memberships

Column name	Type	Description
role_id	int	The role ID
member_role_id	int	This role is member of role_id
inherit	bool	1 for inherit permission, 0 otherwise.

Table permission view

Table 15. sqream_catalog.table_permissions

Column name	Type	Description
database name	varchar	Owning database name
table_id	bigint	Owning table ID
role_id	bigint	The role ID
permission_type	int	The permission type

Database permission view

Table 16. sqream_catalog.database_permissions

Column name	Type	Description
database name	varchar	Owning database name
role_id	bigint	The role ID
permission_type	int	The permission type

Schema permission view

Table 17. sqream_catalog.schema_permissions

Column name	Type	Description
database name	varchar	Owning database name

Column name	Type	Description
schema_id	bigint	Owning schema ID
role_id	bigint	The role ID
permission_type	int	The permission type

Permission type view

Table 18. sqream_catalog.permission_types

Column name	Type	Description
permission_type_id	bigint	The permission type ID
name	varchar	Permission name

2.11. Locks

SQream DB operates in two modes: **exclusive**, which sends a single operation at a time, and **inclusive** which is a multi operations mode. DDL operations are always exclusive.

DML are separated to **DELETE/TRUNCATE** as exclusive; and **INSERT** as inclusive. This allows multiple inserts into the table, but prevents multiple **DELETE** operations.

Querying (**SELECT** operations) can coexists with both DDL and DML.

2.11.1. Locking

Table 19. Locks by SQream

Operation	Select	DML (Insert)	DML (Delete/Truncate)	DDL
Select	No lock	No lock	No lock	No lock
DML (insert)	No lock	No lock	No lock	Lock
DML (delete/truncate)	No lock	No lock	Lock	Lock
DDL	No lock	Lock	Lock	Lock

By default, when a session is requesting a lock on an object and the object is busy, SQream will wait 3 seconds before it return an error message. This wait time is defined in the configuration JSON. See the `statementLockTimeout` parameter in SQream Administrator Guide for more information.

DDL on an object will prevent other DDL/DML to wait on a lock on the same object.

For specific DDL operations, SQream uses global permissions that requires very short exclusive locks at the cluster level. Global permission will be used on operation such as **CREATE DATABASE/TABLE/ROLE**, **ALTER ROLE/TABLE**, **DROP ROLE/TABLE/DATABASE**, **GRANT/REVOKE**.

2.11.2. Viewing locks

To view all existing locks in the SQream database use the utility function show_locks() :

Example

```
SELECT show_locks();
```

2.11.3. Releasing locks

To release a specific lock in an active SQream instance, use the **stop_statement()** utility function with the relevant statement_id. Use the statement_id returned by the **show_locks()** utility function.

Example

```
SELECT stop_statement(12009);
```

To release all locks in a suspect/inactive SQream instance, use the utility function **release_defunct_locks()** which will remove the instance from the cluster and release all its resources. To see all SQream instances status, use **show_cluster_nodes()**.

The utility function **release_defunct_locks()** works only for a system running the metadata server. In a single instance (no metadata server) it will not work.

Example

```
SELECT show_cluster_nodes();
SELECT release_defunct_locks();
```

2.12. Workload Manager

SQream will distribute work throughout the hardware resources to maximize the hardware utilization. By default, this distribution will be done in an equal manner.
The DBA can change this setting and optimize the utilization to their needs by using SQream workload manager and defining each SQream instance to specific service/s.
The specific service to connect to is defined in the session connection string, with the property 'service'. Default service name is: sqream (for more details, see each driver connection string specification).

Each SQream instance can serve multiple services, and each service can work with multiple SQream instances.

2.12.1. Managing Services

Monitor services subscription

```
select show_subscribed_instances();
```

Instance ID is a unique identifier, defined by SQream at the installation, for each instance in SQream cluster.

Example

```
select  show_subscribed_instances('etl_service');
select  show_subscribed_instances();
```

Add services to an existing instance

```
select subscribe_service('instance_id', 'service_name'); ;
```

Example

```
select subscribe_service('node_11', 'etl_service');
```

Remove services to an existing instance

```
select unsubscribe_service('instance_id', 'service_name'); ;
```

Example

```
select unsubscribe_service('node_11', 'etl_service');
```

You can't unsubscribe the last instance from existing service with working/waiting statements in it's queue.

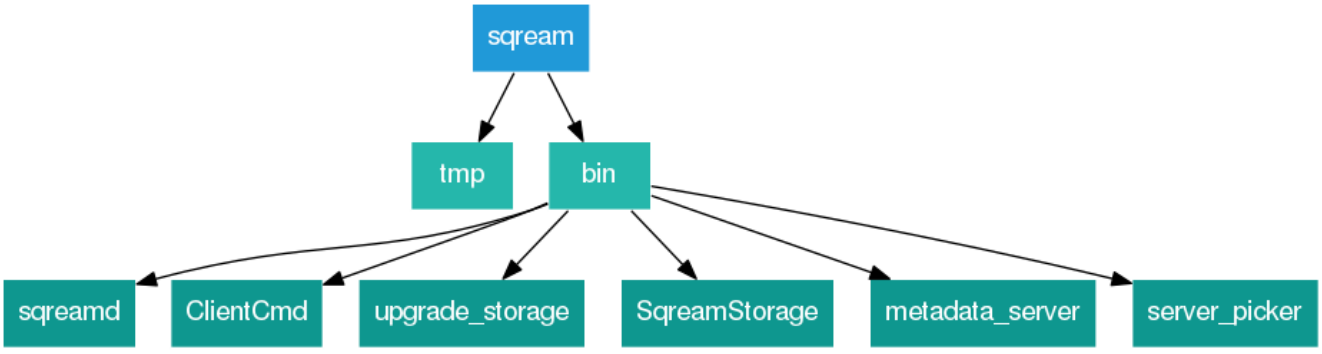
3. Getting Started

3.1. Understanding the SQream DB environment

The SQream environment is usually made up of two folders - the installation directory and the storage cluster.

The installation directory

This folder contains the SQream DB binary applications.

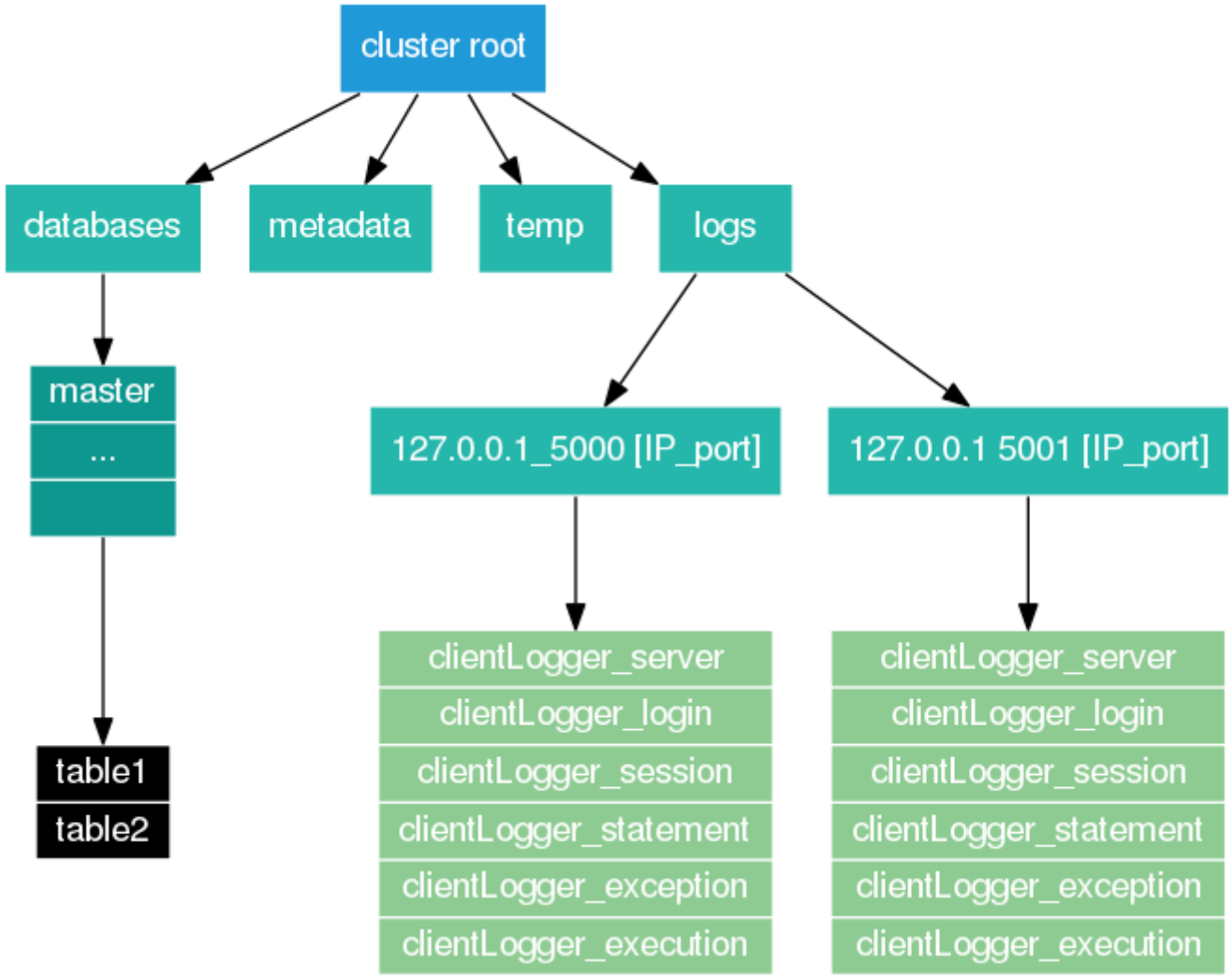


The main applications found in the binary directory are:

Executable Name	Description
`sqreamd`	The sqreamd server daemon
`ClientCmd`	Command line client
`upgrade_storage`	Metadata storage upgrader to be used between versions
`SqreamStorage`	Storage utility to create new clusters and restore access
`metadata_server`	Metadata server for clustered installations
`server_picker`	Load balancer for clustered installations

The storage cluster root

This directory contains the entire database storage.



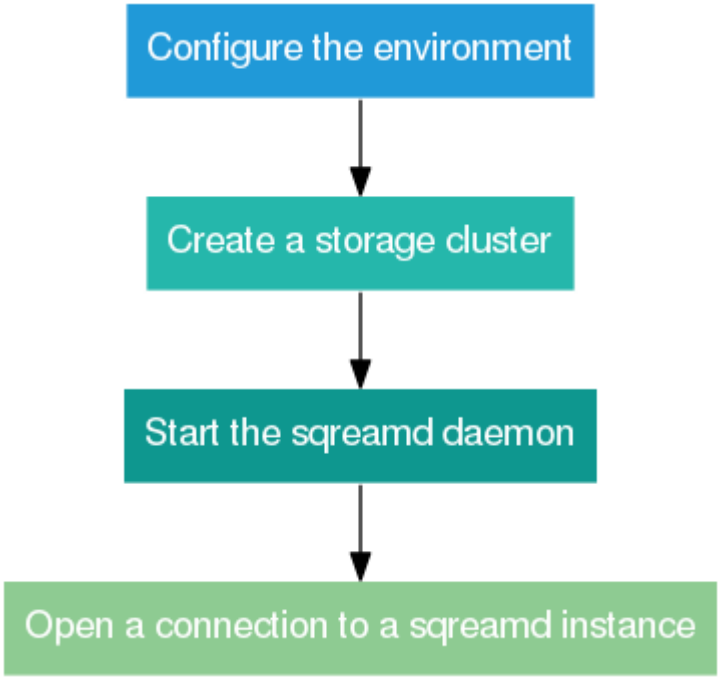
Directory	Description
`databases`	The files containing all databases, tables, columns, chunks, etc.
`metadata`	Internal metadata structures for accessing data from disk
`temp`	Temporary spools

Directory	Description
`logs`	The **logs** folder contains subfolders per SQream instance with folder name = [IP_port]. One folder per running SQream instance.
`127.0.0.1_5000` `[IP_port]`	Contains all logs files of a specific SQream instance

3.2. Setting up SQream DB

If you would like to deploy SQream DB on AWS or Azure please contact SQream directly at info@sqream.com.

Make sure that you are running on a 64-bit Linux operating system, and have an Nvidia GPU installed.
See the Hardware Requirements page to verify your system meets the minimum requirements for running SQream DB.



3.2.1. BIOS and RAID (Checklist)

Out-of-band management interface

- ☐ Ensure connection to management console, Linux host and verify credentials

RAID

- ☐ Ensure all drives are connected and appear in RAID interface
- ☐ Configure RAID drives as required

BIOS settings

Ignore settings where not applicable - some are Dell or HP specific

- ☐ (For Dell PowerEdge servers) Enable Memory Map I/O Over 4GB
- ☐ Set power profile to maximum performance
- ☐ Set power regulator to high performance mode
- ☐ Enable Intel Turbo Boost and Hyperthreading

- ▢ Disable Intel Virtualization Technology
- ▢ Disable Intel VT-d
- ▢ Disable processor C-States (Minimum processor idle power core state)
- ▢ Set Energy/Performance bias to maximum performance
- ▢ Disable dynamic power capping
- ▢ Set DIMM voltage to Optimized for Performance
- ▢ Set memory power savings mode to Maximum performance
- ▢ Enable ACPI SLIT
- ▢ Set QPI Snoop configuration to Home-Snoop or Early-Snoop

3.2.2. OS

We recommend CentOS 7.3, Amazon Linux 2017.03 or Ubuntu 16.04.

- ☐ For CentOS 7.3, install DEVEL with basic video driver
- ☐ Edit your `fstab` to mount all RAID LUNs as required
- ☐ Configure networking as desired
- ☐ Add `sqream` user and set password:

```
sudo useradd -m -U sqream
sudo passwd sqream
```

- ☐ Update user sqream .bashrc with the following

```
echo "alias sqream_status=\"ps axuwww | grep -v 'grep\|tail\|monitor' | grep -i
'sqreamd\|metadata_server\|server_pick\|monit'\"" >> ~sqream/.bashrc
echo "export PATH=$SQREAM_BIN_DIR:$PATH" >> ~sqream/.bashrc
echo "export LD_LIBRARY_PATH=$SQREAM_LIB_DIR:$LD_LIBRARY_PATH" >> ~sqream/.bashrc
```

- ☐ Edit sudoers with 'sudo visudo' and add the following last line:

```
sqream ALL=(ALL)    ALL
```

- ☐ Set UTF8 as system language

```
export LANG=en_US.UTF-8
```

- ☐ Disable X/GUI

```
systemctl set-default multi-user.target
```

- ☐ Add some vm settings to sysctl to prevent kernel panic

```
echo -e " fs.file-max=200000\n vm.dirty_background_ratio = 5 \n vm.dirty_ratio = 10 \n
vm.swappiness = 10 \n vm.zone_reclaim_mode = 7 \n vm.vfs_cache_pressure = 200 \n"  >>
/etc/sysctl.conf
```

- ☐ Open network ports:
  For each installation of SQream DB, make sure ports 3105, 3108 and 5000 are open.
  Port 3105 is used by `metadata_server`, port 3108 is used by `server_picker`, and a single port is used for every `sqreamd` instance (5000 is the default port).
  You may also choose to disable `firewalld` completely.

- [ ] Disable SELINUX:
  Configure `SELINUX=disabled` in the /etc/selinux/config file.

3.2.3. Obtain the tarball from SQream

- [ ] Obtain the latest compressed SQream DB tarball from SQream.

3.2.4. Install prerequisite libraries and kernel headers

On CentOS/RHEL

```
sudo yum install -y zlib-devel openssl openssl-devel nano bzip2 bzip2-devel \
python python-devel libffi libffi-devel apr apr-util apr-devel apr-util-devel \
readline readline-devel gmp-devel libmp3-devel wget gcc kernel-headers kernel-devel
```

Reboot your machine to enable the new kernel.

On Ubuntu

```
sudo apt-get install zlib1g-dev libssl libssl-dev libbz2 libbz2-dev python \
python-dev libffi libffi-dev libapr1 libapr1-dev libreadline6 libreadline6-dev \
libgmp libgmp-dev wget gcc linux-headers-`uname -r`
```

Reboot your machine to enable the new kernel.

3.2.5. Install the latest NVIDIA driver for your GPU

1. Stop any running UI by stopping X
   (`sudo init 3` or `sudo systemctl set-default multi-user.target`)
2. Get the latest NVIDIA driver
   `wget https://developer.nvidia.com/compute/cuda/8.0/prod/local_installers/cuda_8.0.44_linux-run`
3. Run the installer
   `sudo sh ./cuda_8.0.44_linux-run`

3.2.6. Install the SQream DB daemon on the machine

1. Change to the newly created sqream user:
   `su sqream`
2. Unpack the tarball. This expands into the `sqream` directory.
3. Unpacking `tar -xf sqream-<version>.tar.gz`

3.2.7. Create the cluster

1. Enter the `sqream` directory and create a storage cluster
2. `./bin/SqreamStorage -C -r <full path to new cluster>`

Example: Create storage cluster

If your main storage is on `/mnt/storage`, you can run
`./bin/SqreamStorage -C -r /mnt/storage/sqream_storage`

3.2.8. Configure the instances

Each SQream DB daemon must run against a configuration file.

Below is a sample **minimal** configuration file. It is recommended that it be placed in `/etc/sqream/sqream_config.json`. This file will start the server on port **5000**, against GPU #**0**.

Example minimal configuration JSON

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
    "server": {
        "licensePath" : "/etc/sqream/license.enc",
        "port": 5000,
        "cluster": "/mnt/storage/sqream_storage",
        "gpu": 0,
        "ssl_port": 5100
    }
}
```

- JSON files can not contain any comments
- When altering the JSON file, pay close attention to the field-separating commas at the end of the lines.

Permanent instance options may be placed in this file, based on consultation with SQream support or through the configuration utility.

3.2.9. Configuring instances parameters

Table 20. Common parameters setting

Flag Name	Description	Default Value	Range of Values	Remark
`spoolmemorygb`	Select what size spool SQream DB can use for writing intermediate results to RAM, in GB	`128gb`	`1-machine ram size`	Should consider total ram size and number of SQream instances
`insertParsers`	Set the number of parsing threads to be launched for each file, during the bulk load process	`4`	`1-32`	Should consider total ram size and number of SQream instances

Flag Name	Description	Default Value	Range of Values	Remark
insertCompressors	Set the number of compression threads to be launched for each file, during the bulk load process	4	1-32	Should consider total ram size and number of SQream instances
statementLockTimeout	Set the number of seconds SQream will wait for a lock before returning an error	3	1-no limit	
showFullExceptionInfo	Show complete error message	FALSE	TRUE/FALSE	Enabling this setting will often show more detailed error message
initialSubscribedServices	List of services the instance will be subscribed to	sqream	list of strings	Example: "sqream,etl_service,query_service"
useLogMaxFileSize	Defines whether SQream logs should be cycle every logMaxFileSizeMB size	TRUE	TRUE/FALSE	
logMaxFileSizeMB	Set the size of SQream logs cycle	20	>0	Set in MB

3.2.10. Starting and Stopping the SQream DB daemon

Service

The recommended way to run the daemon is via the service, which runs it through `monit`:

Service permissions

Altering the service state requires superuser access (`sudo`), but the SQream DB daemon will start as an unprivileged user.

Starting the service

Starting the service

```
sudo service sqream start
```

Stopping the service

Stopping the service

```
sudo service sqream stop
```

Service permissions

Altering the service state requires superuser access (`sudo`), but the SQream DB daemon will start as an unprivileged user.

Identifying which SQream daemons are running

There are several methods for identifying running SQream daemons. In these examples, two daemons with metadata server and server-picker running.

Using the service script

```
~ $ sudo service sqream status

Metadata Server is running, with pid 41203                 [  OK  ]
Server picker is running at pid 41695                      [  OK  ]
SQream DB is running, with pid 41309.                      [  OK  ]
Running on GPU 0, listening on port 5000
SQream DB is running, with pid 41444.                      [  OK  ]
Running on GPU 1, listening on port 5001
```

Using ps

```
~ $ ps aux | grep sqreamd
sqream    9841  0.0  0.0 112656   972 pts/0    S+   12:42   0:00 grep --color=auto
sqreamd
sqream   41309  1.5  0.7 279354084 1037804 ?   Sl   Jun01 110:20
/home/sqream/sqream/bin/sqreamd -config /etc/sqream/sqream1_config.json
sqream   41444  0.0  0.4 248082604 547636 ?    Sl   Jun01   4:07
/home/sqream/sqream/bin/sqreamd -config /etc/sqream/sqream2_config.json
```

Using nvidia-smi

```
~ $ nvidia-smi
Tue Jun  6 12:43:17 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.48                 Driver Version: 367.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K40m          On   | 0000:04:00.0     Off |                    0* |
| N/A   48C    P0    67W / 235W |  11309MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K40m          On   | 0000:42:00.0     Off |                    0* |
| N/A   54C    P0    68W / 235W |  11309MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     41309    C   /home/sqream/sqream/bin/sqreamd              11305MiB |
|    1     41444    C   /home/sqream/sqream/bin/sqreamd              11305MiB |
+-----------------------------------------------------------------------------+
```

Restarting a specific server daemon

Once you have identified a server that needs restarting, you may control it directly while keeping the rest of the cluster up.

Restarting SQream daemon #2

```
sudo service sqream2 restart
```

Temporarily running the daemon

You may start the daemon temporarily from any unprivileged user.

Running as superuser

Running the server daemon as superuser is discouraged, as it may expose the entire filesystem to SQL users. SQream DB is designed to run as an unprivileged user.

Starting the daemon temporarily

To start the daemon temporarily, run it with `nohup`:

Starting the daemon temporarily

```
nohup ./bin/sqreamd -config /etc/sqream/sqream_config.json &
```

Stopping the daemon temporarily

It is safe to shut down the server by killing it from the operating system; the system is designed to handle this. Any data committed will remain consistent, while uncommitted data will be removed upon the daemon's next start-up.

Stopping a temporarily running server from the command line

```
kill -9 `pid of sqreamd`
```

Stopping a temporarily running server from the SQL interface

```
SELECT shutdown_server();
```

Overriding parameters

You may override the port, GPU and cluster parameters while retaining a configuration file settings by running

```
./bin/sqreamd <path to different cluster> <GPU #> <Port #> -config
<path to configuration json>
```

3.2.11. Connect to the SQream DB server with ClientCmd

To connect to the database server, you may use the ClientCmd command line interface by running

ClientCmd command line arguments

```
./bin/ClientCmd --user=<username> --password=<password> --database=<database name>
--host=<host> --port=<port> --service=<service_name>
```

Connect to the `master` database:

Connecting to `master` on a local host running SQream DB on port 5000 via service 'etl_service'

```
./bin/ClientCmd --user=sqream --password=sqream --database=master --host=127.0.0.1
--port=5000 --service=etl_service
```

3.2.12. Using SSL Server Authentication with SQream

From version 2.1, SQream DB supports secure sockets layer (SSL) encryption and authentication for connections to its cluster via JDBC and ODBC drivers.

To use this option, the server must first be set-up to accept SSL connections.

Enabling SSL server authentication within SQream Instance:

Configure the SERVER flag ssl_port in the SQream instance configuration file to the needed port number.

For example:

```
{
"server":{
    "port": 5001,
    "ssl_port": 5100,
    "cluster": "/path/sqream_cluster",
    "gpu": 0,
    "licensePath": "/path/license.enc"
}
}
```

Restart the SQream DB daemons after making this configuration change

Configure SSL Authentication for JDBC/ODBC drivers

Adding SSL to JDBC

1. Add ssl=true in the connection string
2. Change the port to the SSL port

When connecting via load balancer, the default endpoint for SSH is port 3109

Example for direct connection

```
jdbc:Sqream://hostname:5100/master;user=sqream;password=mypassword;ssl=true;
```

Example for connection to load balancer

```
jdbc:Sqream://hostname:3109/master;user=sqream;password=mypassword;service=sqream;clus
ter=true;ssl=true;
```

Adding SSL to ODBC

In Windows, make sure the SSL checkbox is selected in the DSN settings.

In Linux, add Ssl=true to the connection string

Linux ODBC connection string sample

```
Driver={libODBCDrv.so}:Server=hostname:Port=5100:Database=master:User=sqream:Password=
mypassword:Ssl=true:Service=sqream
```

3.3. Highly available installations

Contact your SQream representative for further information about installing our highly available solutions.

4. Operations

4.1. Upgrading a version

Here are the necessary steps that ensure a smooth upgrade of your SQream DB version

1. Stop SQream instances on all servers
2. On each node that SQream is installed, unpack the new tarball
   alongside the old SQream DB directory.
   For example:

```
$ cd /home/sqream
$ mv sqream sqream-old
$ tar xf sqream-<version>.tar.gz
```

3. Repeat the above step for each node that the SQream DB executables exist
4. It may be necessary to run the metadata upgrade utility.
   (This may take a few moments)

```
$ cd sqream/bin
$ ./upgrade_storage <path to sqream storage cluster>
```

5. Restart the services

4.2. Key administration concepts

See Concepts above

4.3. Monitoring the system

Because SQream DB can be run in a distributed setting, all nodes should be monitored to ensure

smooth operation. It is possible to monitor SQream DB with third party tools like Zabbix, Nagios and others, but also through the OS and SQream DB directly.

4.3.1. From the OS

See Identifying which SQream daemons are running

4.3.2. From each node

See connections to the server

You can monitor existing connections to the database by using the `show_connections()` utility function:

```
SELECT show_connections();
```

Table 21. Sample result from `show_connections()`

ip	conn_id	conn_start_time	stmt_id	stmt_start_time	stmt
192.168.0.93	19	2017-06-22 18:56:54	14	2017-06-22 18:56:54	select show_connections()
192.168.0.93	17	2017-06-22 18:56:48	-1	2017-06-22 18:56:48	

Show server / cluster status

The `show_server_status()` utility function can be used to see which statements are running across the cluster, across all databases.

If no queries are running, this query will return 0 rows in the result set.

```
SELECT show_server_status();
```

Table 22. Sample result from `show_server_status()`

service_id	connection_id	server ip	server port	database_name	user_name	client ip	statement id	statement	statement start time	statement status	statement status start
sqream	32	192.168.0.93	5000	faa	sqream	192.168.0.1	25	SELECT Year	Carrier	destCityName	COUNT(DISTINCT originCityName) from ontime JOIN l_airport_i

Possible statement status values

Status	Description
Executing	The statement is in execution, awaiting results
Preparing	The statement is compiling, and is awaiting execution
Waiting	The statement is waiting in the queue for execution

The DBA can use the show server status output as a baseline for identifying locks and if needed to stop running statements (based on the **server ip : server port** and **statement_id** columns).

By running a query

Running a query, even the most basic one, should give you an indication if a server is up. If you immediately get "Connection refused" or similar, the server is down.

```
SELECT 1;
```

4.4. Stopping existing statements

The **stop_statement()** utility function can be used to cancel or stop a running statement before it finishes.

Usage

- Identify the running statement ID and server IP and port (see **show server status** or **show_connections** above)
- From the same server/port combination - run the stop_statement command:

```
SELECT stop_statement(42);
```

4.5. Logs

SQream DB generates the following log files for the DBA everyday work

Log files:

- **clientLogger_statement** - Keeps track of statements being run in the server, including success/failure, number of rows returned from the query and number of rows processed by the query.
- **clientLogger_server** - Keeps track of server start/stop times
- **clientLogger_session** - Keeps track of different sessions
- **clientLogger_login** - Tracks user login times and IP addresses
- **clientLogger_exception** - Tracks system exceptions
- **clientLogger_execution** - Tracks heavy statements execution statistics (over nodeInfoLoggingSec flag)
- **clientLogger_debug** - Logs debug information. Default debug log level = 4 (INFO). The debug log level can be changed running a statement (e.g., SET logDebugLevel = 6;). No need to restart the SQream instance.
- **metadata_server.log** - Tracks metadata server status and executions queue

File names: Log file names contain a day and time stamp. For example: **clientLogger_statement_20181108_000000.log**

Statement log messages are cut at a length of 10,000 characters. This is to allow loading statements logs files (CSV) into External Tables of SQream. Remember: Row length is limited to 10,240 characters.

Log file configuration:

From V2.18.1 the following default behavior is configured:

- Each SQream instance generates a separate set of log files into a dedicated folder (folder name contains IP and port). For example: /home/sqream/sqream_storage/sqreamdb/logs/**127.0.0.1_5000**.
- Log file rotation: One separate log file per calendar day. New file is generated at midnight.

Related runtimeGlobalflags:

- **useClientLog**: This flag is set the TRUE (by default) to activate the new logging framework.
- **useLogFileRotateHourOfDay**: Enable/disable the rotation of log files. By default this flag is set to TRUE to allow seperate files for separate days.
- **logFileRotateHourOfDay**: Rollover time parameter. By default this flag is set to 0 (=rotation at midnight). Set in the hour of the day. Possible range of values: 0-23.

- **enableLogDebug**: Activate / deactivate the debug log. Default value = true;
- **logDebugLevel**: Supported Log Levels are: 0 = [SYSTEM], 1 - [FATAL], 2 - [ERROR], 3 - [WARN], 4 - [INFO], 5 - [DEBUG], 6 - [TRACE]. Default = 4 (INFO);

4.6. Support Utilities

4.6.1. Report Collection

This support utility is used to collect logs and/or the leveldb at a client's site. The generated tar file can then be sent to the SQream support team for further investigation.

- Output file format: report_[date]_[time].tar

Can run as a utility function or executable, allowing to collect information also if the SQream server is not running. Supported collection modes:

- log = only log files are collected
- db = only leveldb
- db_and_log = both log files and leveldb are collected

Syntax and example when running as a utility function:

```
SELECT report_collection('</pathToOutputFolder>', '<mode>');

SELECT report_collection('/home/sqream/log_collection','log'); )
```

Syntax and example for executable. The executable works only when no SQream instance is running:

```
./bin/report_collection <pathToSqreamDb> </pathToOutputFolder> <mode>

./bin/report_collection /home/sqream/sqream_storage/ /home/sqream/log_collection log
```

4.6.2. Export Reproducible Sample Data

This support utility is used to collect data in order to reproduce a problematic query in a support lab (not only onsite at customer's site). Typically used for query issues that are data related. It runs a query, collects the data and stores the data in a small SQream DB (compressed into a single tar file). This file together with the query can be used in a remote system (support lab) to recreate and investigate the issue.

The output folder contains both the final tar file and data before its compression. It is sufficient to send the tar file. Works as utility function. No executable.

Syntax and example:

```
select export_reproducible_sample('</pathToOutputFolder>', 'sql query1', 'sql query
2', ..);

select export_reproducible_sample('home/sqream/data_collection', 'select * from t');
```

Copyright