



Concepts

SQream Technologies

Version 3.3

Copyright © 2010-2019. 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.

Table of Contents

Table of Contents	3
Concepts	5
SQream DB Daemon	6
Storage Cluster	7
Databases	8
Schemas	9
Tables	10
Columns	12
External Tables	13
User Defined Functions	14
Chunks, Compression and Metadata	15
Chunks	15
Compression specs	15
Metadata	15
Catalog (information schema)	16
Querying the SQream catalog	16
Available catalog views	17
Database object catalog	17
Fine-grain storage catalog	17
Role and permission catalog	17
Database objects	18
Databases view	18
Schemas view	18
Tables view	18
Columns view	19
Views view	19
External tables view	19
User defined functions	20
Fine-grain storage	20

Extent view	20
Chunks view	20
Delete predicates view	21
Role and permission catalog	21
Role view	21
Role membership view	21
Table permission view	22
Database permission view	22
Schema permission view	22
Permission type view	22
Locks	23
Locking	23
Monitoring locks	23
Releasing locks	24
Workload Manager	25
Managing Services	25
Monitor services subscription	25
Add services to an existing instance	25
Remove services to an existing instance	25

Concepts

[Download as PDF](#)

This section describes SQream's database concepts.

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.

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

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 \(information schema\)](#).
- To create a new database, query the [CREATE DATABASE](#) command.

Schemas

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

- To view existing schemas, query the [Catalog \(information schema\)](#).
- To create a new schema, query the [CREATE SCHEMA](#) command.
- To manage schemas permissions, query [Database Roles and Permissions](#).



Use schemas to split up the data tables logically

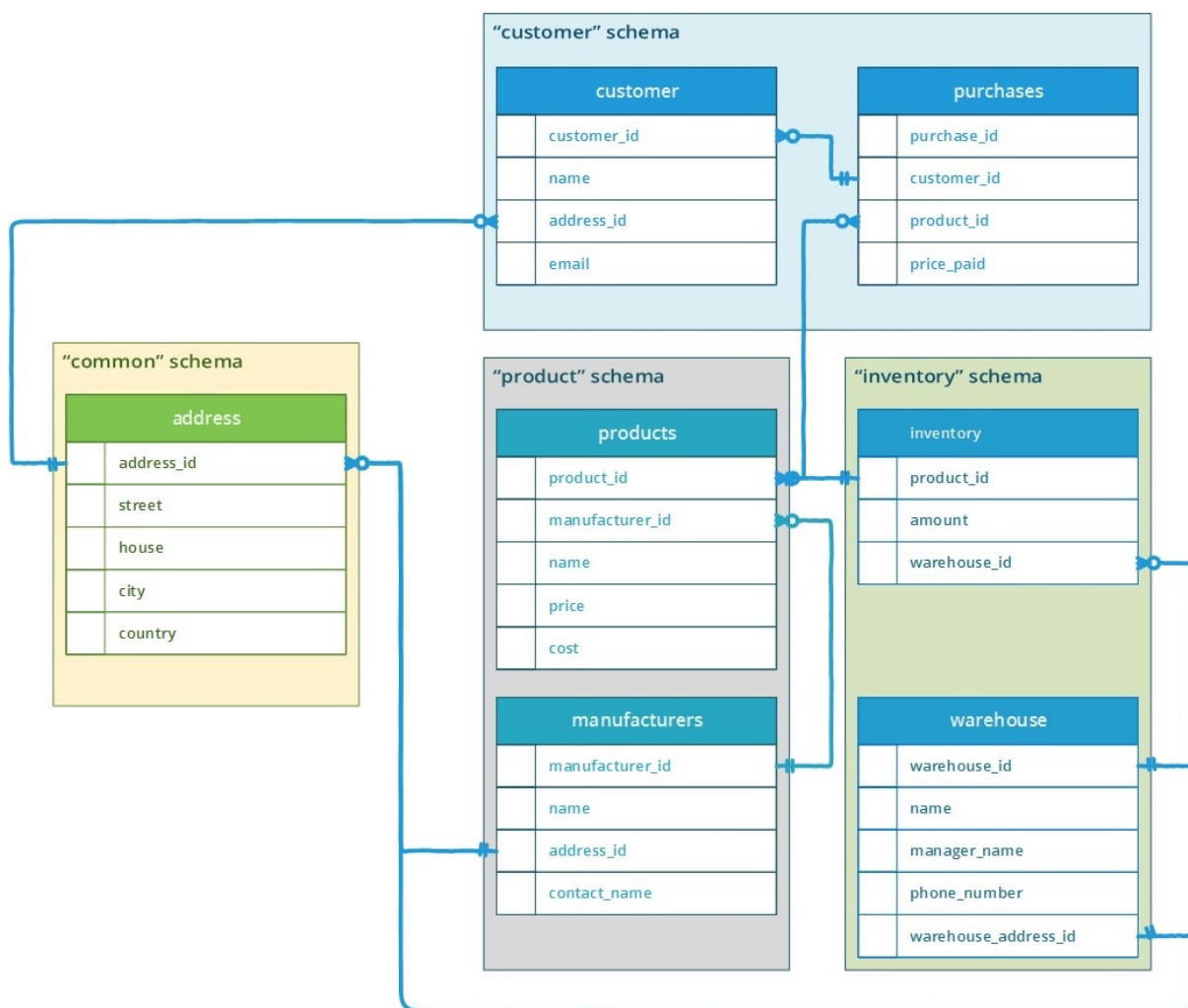


Figure 1. Example of schema usage

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.

- To view tables, query the [Catalog \(information schema\)](#).
- To create a new table, query the [CREATE TABLE](#) command.
- To manage tables permissions, query [Database Roles and Permissions](#).

id	name	ts
int	varchar(30)	datetime

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;
```

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.

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.

- To view existing external tables, query the [Catalog \(information schema\)](#).
- To create a new external table, query the [CREATE EXTERNAL TABLE](#).
- To manage external tables permissions, query [Database Roles and Permissions](#).

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.

To read more about UDF, please see [User Defined Functions](#) in the SQL Reference Guide.

Chunks, Compression and Metadata

Chunks

SQream DB splits columnar data into chunks. The chunk size will determine the minimum bulk load into the GPU. For better performance, SQream DB rearranges previously loaded data into new chunks based on the configured **chunk size**. When 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). Chunk size is a parameter at the cluster level. It must be set before the first insert to the cluster. It can also be set at the database level, before any tables are created. Default chunk size is 1 million rows. Ask your database administrator about setting the chunk size.



- The chunk size influences load/query time. Before tuning the parameter, consult your [SQream account manager](#).

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.

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. Gathering metadata is automatic and transparent and requires no user intervention.

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.

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;
```


Available catalog views

Database object catalog

Table 2. SQream catalog views

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

Fine-grain storage catalog

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

Role and permission catalog

View name	Description
<code>sqream_catalog.roles</code>	Roles (users) in the current databases
<code>sqream_catalog.role_memberships</code>	Roles membership
<code>sqream_catalog.table_permissions</code>	Tables and their assigned roles
<code>sqream_catalog.database_permissions</code>	Databases and their assigned roles
<code>sqream_catalog.schema_permissions</code>	Schemas and their assigned roles
<code>sqream_catalog.permission_types</code>	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
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. *scream_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_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. *scream_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 text that created this view

External tables view

Table 8. *scream_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

Column name	Type	Description
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. *scream_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. *scream_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. *scream_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
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

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.

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**.

Monitoring locks

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

Example

```
SELECT show_locks();
```

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 ();
```


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.

For an easy start, see [Quick guide to using SQream Dynamic Workload Manager \(DWLM\)](#).

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.