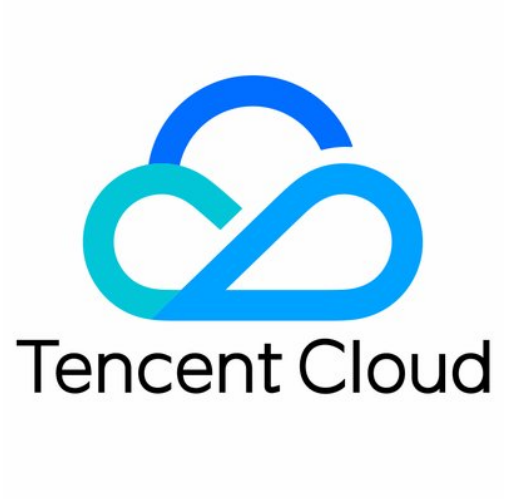
Stream Compute Service

SQL Manual

Product Documentation



©2013-2019 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

©2013-2019 Tencent Cloud. All rights reserved.

SQL Manual
Terminology & Data Type
Terminology

Last updated：2019-11-12 18:32:27

Common terminology for SCS:

Terminology	Description
Stream Compute Service (SCS)	SCS can persistently read data from streaming sources, and write the data into a sink after it is processed by an SQL program written by the user. It can also perform multiple tasks in series.
Source	Provides input data for SCS, such as Tencent Cloud CKafka.
Sink	Indicates a place where the computing results of SCS are output, such as Tencent Cloud CKafka.
Schema	Indicates the structural information of a table, such as the column names, column types, etc.
Time mode	Instructs the system on how to get timestamps when processing data.
Event Time	In the Event Time mode, timestamps are provided by a field in the input data. You can use the WATERMARK FOR statement to specify the field and enable the Event Time mode. This mode is suitable for the scenarios where sources contain exact timestamps.
Watermark	Indicates a specified point in time before which all data has been processed properly. Watermark is automatically generated by the system, and you can specify the maximum tolerance of timestamps through the WATERMARK FOR BOUNDED statement.
Processing Time	In the Processing Time mode, timestamps are automatically generated by the system and added to the source (named after `PROCTIME`, not visible in SELECT *, and must be explicitly specified when used). A timestamp is the time when each piece of data is processed by the system, which is uncontrollable and suitable for scenarios that do not require high time accuracy.
Time window	Defines multiple time periods and the relationship between them (for example, whether they can overlap, or whether they are fixed in size). Supported values include TUMBLE, HOP, and SESSION.

©2013-2019 Tencent Cloud. All rights reserved.

Terminology	Description
Stream Connector	It is a high-performance and high-availability messaging system provided by Tencent Cloud, which supports the definition of Schema and inputs and outputs data in this format. It is the most perfect source and sink supported by SCS. The topic types of Stream Connector include Tuple, Upsert, and Blob (not available). You can create a project and a topic under it in the "Stream Connector" page.
Integrator	Stream Connector's integrator is responsible for outputting the data in Stream Connector to other sinks, such as CDB (MySQL PostgreSQL), and COS (Cloud Object Storage), so as to realize the final output of computing results.
Tuple and Append stream	Tuple is a type of Stream Connector that can store incremental Append streaming data. An Append stream is a data stream that is consistently appended with new data. It does not update previously issued data. A variety of sources and sinks support input and output of Append stream.
Upsert and Upsert stream	Upsert is a type of Stream Connector data table that can store Upsert streaming data. Upsert (abbreviation of update or insert) is generated by queries such as DISTINCT, non-window-based GROUP BY, non-window-based JOIN and other statements, which has a primary key. If the data issued at a later point in time has the same primary key as a piece of previous data, the record will be updated to a new value. Otherwise, a new row of data is added. It ensures that previously issued data is updated to reflect the latest value. Upsert streams can only be written into Stream Connector.
CKakfa	CKafka is a distributed, high-throughput and highly scalable messaging system provided by Tencent Cloud, and is fully compatible with Kafka 0.9 API. CSV and JSON are supported as input and output formats.
CDB (for MySQL)	CDB is a high-performance, high-reliability and scalable database hosting service provided by Tencent Cloud. It allows users to easily deploy and use MySQL databases on the cloud.
DDL	DDL, short for Data Definition Language, is a subset of the SQL language and consists of CREATE statements. It can be used to define tables, views, and user-defined functions (UDF), etc.
DML	DML, short for Data Manipulation Language, is a subnet of the SQL language and consists of INSERT and SELECT statements. It can be used to select, convert, filter and insert data tables and views.

©2013-2019 Tencent Cloud. All rights reserved.

Data Type

Last updated：2019-08-08 13:53:12

SCS employs the type definition conforming to ANSI SQL standards. The types allowed when you define a Stream Connector or a CKafka source or sink are limited, while those allowed when you define a CDB or a view can be all supported types.

Types Allowed When Defining a Stream Connector or a CKafka Source or Sink Table

If Stream Connector or CKafka is used as source and sink tables, SCS supports the following data types:

Type Name	Description for Java Users
BIGINT	It is equivalent to Java's Long type, which takes up 8 bytes.
DOUBLE	It is equivalent to Java's Double type, which takes up 8 bytes.
BOOLEAN	It is equivalent to Java's Boolean type, which can be True or False.
TIMESTAMP	Supports standard SQL timestamps in the format of YYYY-MM-DD HH:MM:SS Also supports Unix timestamps (in ms), such as 1527501994642.
VARCHAR	Indicates a string. It is equivalent to Java's String ring type with unlimited length. Size is not required. It is compatible with VARCHAR (size), but the size can be set to any positive number, which has no actual meaning.

Types Allowed When Defining a CBD Source or Creating a VIEW

When you define a source with CDB (a relational database) or create a view (CREATE VIEW), and use CAST() for type conversion, this system supports the following data types:

Type Name	Description for Java Users
VARCHAR	Indicates a string. It is equivalent to Java's String ring type with unlimited length. Size is not required.

©2013-2019 Tencent Cloud. All rights reserved.

Type Name	Description for Java Users
BOOLEAN	It is equivalent to Java's Boolean type, which can be True or False.
TINYINT	It is equivalent to Java's Byte type, which takes up 1 bytes.
SMALLINT	It is equivalent to Java's Short type, which takes up 2 bytes.
INTEGER or INT	It is equivalent to Java's Integer type, which takes up 4 bytes.
BIGINT	It is equivalent to Java's Long type, which takes up 8 bytes.
REAL or FLOAT	It is equivalent to Java's Float type, which takes up 4 bytes.
DOUBLE	It is equivalent to Java's Double type, which takes up 8 bytes.
DECIMAL	It is equivalent to Java's BigDecimal, which represents large numbers and decimals with any precision.
DATE	It is equivalent to java.sql.Date, which indicates the specified year, month and day (YYYY-MM-DD).
TIME	It is equivalent to java.sql.Time, which indicates the specified hour, minute and second (YYYY-MM-DD).
TIMESTAMP	Supports standard SQL timestamps in the format of YYYY-MM-DD HH:MM:SS, such as 2018-06-13 16:58:10. Also supports Unix timestamps (in ms), such as 1527501994642.
INTERVAL YEAR TO MONTH	Indicates the time period calculated by month. Data is stored internally in Integer type, which takes up 4 bytes.
INTERVAL DAY TO SECOND	Indicates the time period calculated in ms. Data is stored internally in Long type, which takes up 8 bytes.
ARRAY	Indicates an array, which corresponds to Java's array.
MAP	Indicates map, which corresponds to Java's HashMap.
MULTISET	Indicates a collection of duplicate values allowed to be saved.

©2013-2019 Tencent Cloud. All rights reserved.

DDL Statements

CREATE TABLE

Last updated : 2019-08-07 11:35:31

The CREATE TABLE statement is used to describe a source or a sink table.

Syntax:

```
CREATE TABLE `Table name` (
`Field Name` Field Type
[, `Field Name` Field Type ]*
[, WATERMARK FOR BOUNDED (Name of the timestamp field, the maximum time allowed for out-of-order arrival) ]
[, WATERMARK FOR ROWS (how many rows that can generate a Watermark) ]
[, PRIMARY KEY (Primary key 1, ...) ]
) WITH (
`Parameter Name` = 'Parameter Value'
[, `Parameter Name` = 'Parameter Value' ]*
)
```

BOUNDED and ROWS belongs to Event Time mode, i.e., the source comes with a timestamp field. Both types of WATERMARK of ROWS are mutually exclusive, so only either of which can be selected.
The maximum time allowed for out-of-order arrival is meaningful in the Event Time mode, while the processing order cannot be guaranteed in the Processing Time mode because the source data comes with no timestamp.

Example:

```
CREATE TABLE KafkaSource1 (
`time_` VARCHAR,
`client_ip` VARCHAR,
`method` VARCHAR(20),
) WITH (
`type`='ckafka',
`instanceId` = 'ckafka-cky18642',
`encoding` = 'json',
`topic` = 'test-input'
);
```

Source and Sink

©2013-2019 Tencent Cloud. All rights reserved.

SCS can automatically determine the source and sink tables according to the subsequent INSERT INTO and SELECT FROM statements, so you don't need to explicitly specify the types of both tables, but should note that CDB (MySQL) can only be used as the source and applied to the right table in JOIN operations.

You can specify the type of source or sink in the WITH parameter of CREATE TABLE. For example, if type = 'cdp', Stream Connector is used, and if type = 'ckakfa', CKafka is used as the source.

> **Note:**
> The parameter after the equal sign must be enclosed in half-angle single quotation marks. Double quotation marks or full-angle quotation marks are not allowed. Generally, field names are not case-sensitive (e.g., `type` and `TYPE` are equivalent), but the string inside the single quotes is case-sensitive when referenced as an external value (e.g., `root` and `ROOT` are different as user names).

The following parameters are required for various types of source and sink:

Stream Connector

Parameter	Description
type	"cdp" should be specified if source or sink is Stream Connector.
project	The name of a Stream Connector project.
topic	The topic of the project specified by Stream Connector.
startMode	(Optional) Available values: EARLIEST (read from the earliest offset) and LATEST (read from the latest offset).
timestampMode	(Optional) It is used to specify the timestamp format for a source or sink table. It is "AUTO" by default. For a source table, the timestamp is determined depending on the format of input data (a value larger than 99999999999 is regarded as MILLISECOND, and that less than 99999999999 is regarded as SECOND). For a sink table, the timestamp is output as MILLISECOND format. If it is explicitly specified as "MILLISECOND", a Unix timestamp in milliseconds is used. "SECOND" indicates a Unix timestamp in seconds.** Note: **Since "AUTO" mode will judge each piece of data, which may slow the performance. In a low-latency and high-throughput environment, explicitly specify the timestampMode parameter for better performance.

©2013-2019 Tencent Cloud. All rights reserved.

> **Note:**
> Stream Connector tables include Tuple and Upsert tables. A Tuple table does not have a primary key (i.e., it comes with no PRIMARY KEY statement), only supports the Append (data are only appended and those previously written are not updated) operation, and can receive the results of most queries (i.e. Append streams).

An Upsert table has a primary key (i.e., PRIMARY KEY is used to define the primary key), supports INSERT INTO and Upsert operations, and can receive Upsert streams (Upsert is short for Update OR Insert, that is, if a record with the same primary key as a piece of data has been previously output, the record is updated, otherwise new data is inserted) generated from DISTINCT, non-window-based JOIN, non-window-based GROUP BY and other operations. These Upsert streams can only be written into Stream Connector sink tables of Upsert type that cannot be used as source tables, and they should not be mixed.

Example: Stream Connector source and sink tables of Tuple type in the Processing Time mode

For more information on time modes and WATERMARK, see the WATERMARK section below.

```
CREATE TABLE `traffic_output` (
`f1` VARCHAR,
`f2` BIGINT
) WITH (
`type` = 'cdp',
`project` = 'test',
`topic` = 'Output',
`startMode` = 'EARLIEST'
);
```

If the Processing Time mode is used, a Stream Connector Tuple table that contains f1, f2 and PROCTIME (automatically generated to indicate the timestamp of each record when it is processed, which can be used to describe the time window) columns is defined, which can be source or sink.

Example: Stream Connector source and sink tables of Tuple type in the Event Time mode

```
CREATE TABLE `public_traffic_output` (
`rowtime` TIMESTAMP,
`f1` VARCHAR,
`f2` BIGINT,
WATERMARK FOR BOUNDED(`rowtime`, 5000) -- The timestamp field and the maximum allowable time rang
e for out-of-order arrival of data used to define the Event Time mode.
) WITH (
`type` = 'cdp',
`project` = 'test',
```

©2013-2019 Tencent Cloud. All rights reserved.

```
`topic` = 'Output'
);
```

If the Event Time mode is used for the table defined above, a Stream Connector Tuple table that contains f1 and f2 columns is defined, which can be a source or sink table.

Example: Stream Connector sink table of Upsert type

```
CREATE TABLE `public_traffic_output` (
`f1` VARCHAR,
`f2` BIGINT,
PRIMARY KEY(`f1`) -- The type of Stream Connector table used to define the primary key is Upsert
) WITH (
`type` = 'cdp',
`project` = 'test',
`topic` = 'Output'
);
```

If the Processing Time mode is used for the table defined above, a Stream Connector Upsert table that contains f1 and f2 columns is defined, which can only be a sink table.

CKakfa

Parameter	Description
type	"ckafka" should be specified if source or sink is CKafka.
instanceId	The instanceId of CKafka.
encoding	It can be json or csv. In case of csv, fieldDelimiter must also be specified.
topic	The topic under the instanceId specified by Ckafka.
timestampMode	(Optional) It is used to specify the timestamp format for a source or sink table. It is "AUTO" by default. For a source table, the timestamp is determined depending on the format of input data (a value larger than 99999999999 is regarded as MILLISECOND, that less than 99999999999 is regarded as SECOND, and a string is regarded as SQL). For a sink table, the timestamp is output as MILLISECOND format. If it is explicitly specified as "MILLISECOND", a Unix timestamp in milliseconds is used. "SECOND" indicates a Unix timestamp in seconds.** "SQL" means a string timestamp in the format of yyyy-MM-dd HH:mm:SS. ** Note: **Since "AUTO" mode will judge each piece of data, which may slow the performance. In a low-latency and high-throughput environment, explicitly specify the timestampMode parameter for better performance.

©2013-2019 Tencent Cloud. All rights reserved.

Parameter	Description
fieldDelimiter	This is optional when encoding is CSV, which specifies the delimiter for each field of CSV. It is a comma (',') by default.
startMode	(Optional) Available values: EARLIEST (read from the earliest offset), LATEST (read from the latest offset), and GROUP (read from the specified groupId that must be used).
groupId	Specifies the groupId to be read (only for startMode = 'GROUP' mode).
ignoreErrors	(Optional) It is true by default, which means to skip wrong rows. If it is set to false, the program will be terminated directly in case of data error.

Notes:

- If data contains the same character as the delimiter, it will be enclosed in double quotation marks to avoid ambiguity. If data itself contains double quotation marks, each double quote will be replaced with two double quotes ("").
- CKafka only supports writing Append streams instead of Upsert streams. Use Stream Connector to write Upsert streams.

CDB (only used as the right table in a JOIN condition)

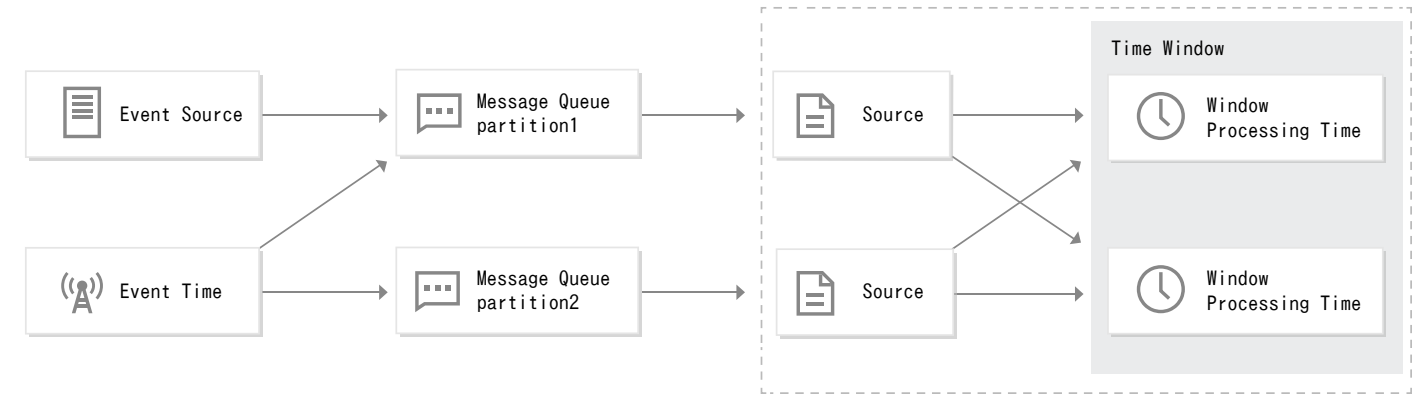
Field	Meaning
instanceId	ID of the CDB instance (case-sensitive).
database	Database name (case-sensitive).
table	Table name (case-sensitive).
user	Username (case-sensitive).
password	Password (case-sensitive).

WATERMARK

Event Time / Processing Time

For window-based operations (such as the assignment of time periods in GROUP BY, OVER, and JOIN conditions), SCS supports two time processing modes: Event Time and Processing Time.

©2013-2019 Tencent Cloud. All rights reserved.



The Event Time mode uses the timestamp of the input data to tolerate a certain degree of out-of-order data input (for example, the earlier data arrived later due to unpredictable reasons such as the processing capacity of each node and network fluctuations), and this parameter can be specified by BOUNDED's second parameter (in milliseconds). It is the most accurate processing mode, but requires the input data to have a timestamp. Only fields defined by the timestamp type in the source are supported. Virtual columns will be supported in the future, and other types of columns can be converted into timestamps accepted by the system by applying processing functions.
The Processing Time mode does not require the input data to have a timestamp, but automatically adds the timestamp of the data when it is processed to the data and names it with the PROCTIME (uppercase) field. This column is hidden and will not appear when you perform SELECT *. It is read only when you use it manually.

Note:
Only one time mode is allowed for all sources of the same task. If the Event Time mode is used in a task, a timestamp must be defined for all defined Table Sources, and WATERMARK timestamp field must be declared.

ROWS / BOUNDED

To process window-based data that contains timestamp information (columns are represented by timestamps in SQL format or UNIX timestamps), it is recommended to use Event Time mode. For example, if the data contains a generation_time field, and the maximum error allowed for out-of-order arrival is 1,000 milliseconds, then you can declare `WATERMARK FOR BOUNDED( generation_time , 1000)` to enable the Event Time mode.

Example:
If the data contains a generation_time field, and the maximum error allowed for out-of-order arrival

©2013-2019 Tencent Cloud. All rights reserved.

is 1,000 milliseconds, you can declare:

WATERMARK FOR BOUNDED(`generation_time` , 1000)

If you want Watermark to be generated once every 100 pieces of data, then you can declare:

WATERMARK FOR ROWS(`generation_time` , 100)

The Event Time mode can be enabled for both declarations.

If you do not declare Watermark to specify a timestamp, the Processing Time mode is used. In this mode, Watermark is generated with the timestamp of the data when it is processed and will be used later, which cannot guarantee the order and accuracy. This mode is applicable to scenarios that do not require high time accuracy.

©2013-2019 Tencent Cloud. All rights reserved.

CREATE VIEW

Last updated : 2019-08-08 13:58:10

You can use the CREATE VIEW statement to create a view. A view is a virtual table based on the SELECT statement. Views can be used to define new virtual sources (type conversion, column conversion, virtual column, etc.), split long code, and so on.

Syntax

```
CREATE VIEW View name AS
SELECT Clause
```

Example 1

Create a view named MyView:

```
CREATE VIEW MyView AS
SELECT s1.time_, s1.client_ip, s1.uri, s1.protocol_version, s2.status_code, s2.date_
FROM KafkaSource1 AS s1, KafkaSource2 AS s2
WHERE s1.time_ = s2.time_ AND s1.client_ip = s2.client_ip;
```

Example 2

In calculations, due to a large amount of data and the matching requirements of function method types, TINYINT, SMALLINT, REAL and other types must be used. However, when a Stream Connector input type does not meet the requirements, a virtual view can be defined as a new source using the CREATE VIEW statement in conjunction with CAST() type conversion function .

In the following example, a view named KafkaSource2 is defined to convert a status_code column in the KafkaSource1 source from BIGINT type to VARCHAR type:

```
CREATE VIEW KafkaSource2 AS
SELECT
`time_`,
`client_ip`,
`method`,
CAST(`status_code` AS VARCHAR) AS status_code,
FROM KafkaSource1;
```

Notes:

Improper use of the CAST() data conversion function may result in accuracy loss, for example,

©2013-2019 Tencent Cloud. All rights reserved.

```
from BIGINT to INTEGER or TINYINT. Please use it with caution.
```

<!-->For the type conversion between a string (VARCHAR) and a timestamp (TIMESTAMP), see functions such as TO_TIMESTAMP, DATE_FORMAT_SIMPLE and DATE_FORMAT in Time-related Functions.-->

©2013-2019 Tencent Cloud. All rights reserved.

DML Statements

Last updated：2019-08-08 14:02:56

INSERT INTO

The INSERT INTO statement must be used in conjunction with SELECT subqueries.

Syntax

```
INSERT INTO Sink
SELECT Clause
```

Example

Insert the result of a SELECT query into the sink named KafkaSink1:

```
INSERT INTO KafkaSink1
SELECT s1.time_, s1.client_ip, s1.uri, s1.protocol_version, s2.status_code, s2.date_
FROM KafkaSource1 AS s1, KafkaSource2 AS s2
WHERE s1.time_ = s2.time_ AND s1.client_ip = s2.client_ip;
```

SELECT FROM

Syntax

```
SELECT Comma-separated fields to be selected
FROM Source or view
WHERE Filter condition
Other subqueries
```

Example

```
SELECT s1.time_, s1.client_ip, s1.uri, s1.protocol_version, s2.status_code, s2.date_
FROM KafkaSource1 AS s1, KafkaSource2 AS s2
WHERE s1.time_ = s2.time_ AND s1.client_ip = s2.client_ip;
```

©2013-2019 Tencent Cloud. All rights reserved.

> **Note:**
> SELECT cannot be used alone. It must be used with CREATE VIEW … AS or INSERT INTO, otherwise a prompt saying that no suitable operator exists will appear.

WHERE

WHERE is used to filter query conditions (predicates). Multiple parallel conditions can be joined by AND/OR.

> **Note:**
> To JOIN with a table of an external CDB, only AND is used to join conditions. To use OR, see UNION ALL to achieve the same purpose.

HAVING

HAVING is used to filter the results after GROUP BY. Note that WHERE is used before GROUP BY, and HAVING is used after GROUP BY.

Example

```
SELECT SUM(amount)
FROM Orders
WHERE price > 10
GROUP BY users
HAVING SUM(amount) > 50
```

GROUP BY

In SCS, GROUP BY is used to group and aggregate results, including time window-based GROUP BY, and non-window-based GROUP BY (also known as persistent query). Since the former will not update the previous results, a data stream of Append type is generated, which can only be written into the Stream Connector sink of Tuple type or CKafka. However, the latter will update the previous records,

©2013-2019 Tencent Cloud. All rights reserved.

so a data stream of Upsert type is generated, which can only be written into the Stream Connector sink of Upsert type.

Time window-based GROUP BY

This example defines the GROUP BY query statement containing a time window.

```
SELECT user, SUM(amount)
FROM Orders
GROUP BY TUMBLE(rowtime, INTERVAL '1' DAY), user
```

> **Note:**
> In the Event Time mode (where WATERMARK FOR BOUNDED is used to define the timestamp field), the first parameter of the TUMBLE window function must be this field. In the Processing Time mode, the first parameter of the TUMBLE window function must be PROCTIME (uppercase). This applies to both HOP and SESSION.

Non-window-based GROUP BY (persistent query)

This example defines the GROUP BY query statement that does not contain a time window, which is called persistent query. Because it calculates and determines whether to update the results issued previously based on each piece of new data, an Upsert stream is generated.

```
SELECT a, SUM(b) as d
FROM Orders
GROUP BY a
```

> **Note:**
> This method may cause memory overflow due to too many keys or too much data. So, be careful when setting the object timeout. Do not set this value too large.

JOIN

SCS only supports Equi-JOIN (where the JOIN condition contains at least a filter condition that makes a field in the left table equivalent to that in the right table) and Inner JOIN. Outer JOIN will be available in future versions.

©2013-2019 Tencent Cloud. All rights reserved.

Inner Equi-JOIN between streams

There are two types of stream-stream JOIN: those with and without a time range. The former generates streams of Append type, while the latter generates streams of UP SERT type.

Stream-stream JOIN with a time range

The WEHERE condition of a JOIN with a time range contains at least an Equi-JOIN condition and a specified time range. The time range can be represented by <, <=, >=, >, or BETWEEN ... AND.

Example of a time range:

```
ltime = rtime
ltime >= rtime AND ltime < rtime + INTERVAL '10' MINUTE
ltime BETWEEN rtime - INTERVAL '10' SECOND AND rtime + INTERVAL '5' SECOND
```

Exampe:

```
SELECT *
FROM Orders o, Shipments s
WHERE o.id = s.orderId AND
o.ordertime BETWEEN s.shiptime - INTERVAL '4' HOUR AND s.shiptime
```

Stream-stream JOIN without a time range

It only needs at least one Equi-JOIN, and the time range is optional. That is, it calculates all active data in the history (inactive elements can be removed by specifying a timeout).

> **Notes:**
> It may take up too much memory and should be used with caution. Generally, an appropriate object timeout should be set to remove inactive objects in a timely manner.
> This query will generate an Upsert stream, and only a Stream Connector sink of Upsert type can be used to receive data.

Example:

```
SELECT *
FROM Orders INNER JOIN Product ON Orders.productId = Product.id
```

JOIN between a stream and a CDB table

SCS also supports JOIN between a stream and a data table of CDB for MySQL. The syntax is the same as described above, but the CDB table must be the right table in a JOIN condition.

©2013-2019 Tencent Cloud. All rights reserved.

Note that a JOIN query condition should include all the defined keys of the table, otherwise the task will fail due to excessive query results and memory usage.

Example:

```
SELECT d.client_agent AS time_, d.client_ip, d.numbers AS request_body_length
FROM StreamSource AS s, DimSource AS d
WHERE s.client_ip = d.client_ip AND d.`month` LIKE '20180%' AND ABS(d.numbers) BETWEEN 0 AND 2000
```

JOIN with an array

Joining a defined array object (the value constructor in the section 4.10.4 can be used to construct an array object ARRAY) is also allowed by SCS.

Example: (if tags is a defined array)

```
SELECT users, tag
FROM Orders CROSS JOIN UNNEST(tags) AS t (tag)
```

UNION ALL

UNION ALL is used to merge the results of two queries. Besides, since joining query conditions with OR is not supported in case of a JOIN between a stream and a CDB table, UNION ALL can also be used to achieve the same query result.

Example

```
SELECT *
FROM (
(SELECT user FROM Orders WHERE a % 2 = 0)
UNION ALL
(SELECT user FROM Orders WHERE b = 0)
)
```

Note:
SCS only supports UNION ALL instead of UNION, which means de-duplication is not implemented in a row. To perform de-duplication to implement UNION, use it with DISTINCT. Note that DISTINCT will change the result from Append stream to Upsert stream, so only a Stream Connector sink of Upsert type can be used.

©2013-2019 Tencent Cloud. All rights reserved.

OVER Window Aggregation

Use OVER to perform sliding-window aggregation (instead of GROUP BY aggregation) on data streams. You can specify PARTITION, ORDER, window range and other parameters in OVER.

Example

The following example defines a sliding-window aggregation query to calculate the amount of a sliding window with a size of 3. Perform PRECEDING on the previous rows, but FOLLOWING is not supported.
In addition, only one timestamp filed can be placed after ORDER BY. In this example, the PROCTIME column is automatically added by the system in the Processing Time mode.

```
SELECT SUM(amount) OVER (
PARTITION BY user
ORDER BY PROCTIME
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM Orders
```

ORDER BY

ORDER BY is used to sort the results of a query. The default value is ASC (ascending order), or you can also explicitly specify DESC (descending order).

> **Note:**
> The first sorting item must be the time-based column (Event Time timestamp, or Processing Time timestamp, i.e. PROCTIME) in ascending order, and other sorting items can be freely specified.

Example

```
SELECT *
FROM Orders
ORDER BY `orderTime`, `username` DESC, `userId` ASC
```

DISTINCT

DISTINCT is used to de-duplicate query results, which must be placed after SELECT.

©2013-2019 Tencent Cloud. All rights reserved.

Example

```
SELECT DISTINCT users FROM Orders
```

Notes:
DISTINCT will generate an Upsert stream, so only a sink of Upsert type can receive its results. Moreover, queries that take a long time may lead to excessive memory usage. Please use it with caution.
Set an appropriate object timeout to remove inactive objects in time to save memory.

Syntax Structures Not Yet Supported

The following SQL syntax structures are not supported:
GROUPING SETS, ROLLUP, CUBE, IN, UNION (which can be implement using UNION ALL and DISTINCK), LIMIT, etc. More syntaxes will be available in future versions, such as real-time query of Top data.

©2013-2019 Tencent Cloud. All rights reserved.

Identifiers & Reserved Words

Naming Rules

Last updated：2019-08-07 11:39:56

Identifiers uniquely indicate table names or column names. The naming rules of identifiers in SCS are as follows:

- The reserved words specified in Data Type cannot be used. If you must use a reserved word as table name or column name, enclose it in back quotes (` `), for example `time`.
- Do not use PROCTIME as column name to avoid conflicts with system-generated timestamps.
- Do not start with _DataStreamTable_.
- If a table name or column name contains spaces or special characters, enclose it in back quotes, for example `HELLO WORLD`.
- The length must be less than or equal to 128 characters (half-angle).

©2013-2019 Tencent Cloud. All rights reserved.

Reserved Words

Last updated：2018-11-27 11:06:30

The reserved words in SCS are as follows. To use any of these reserved words as table name or column name, enclose it in back quotes (` `). Otherwise, an error will be returned during syntax check.

A

A, ABS, ABSOLUTE, ACTION, ADA, ADD, ADMIN, AFTER, ALL, ALLOCATE, ALLOW,
ALTER, **ALWAYS**, **AND**, **ANY**, **ARE**, ARRAY, **AS**, **ASC**, ASENSITIVE, **ASSERTION**,
ASSIGNMENT, ASYMMETRIC, **AT**, ATOMIC, **ATTRIBUTE**, **ATTRIBUTES**, AUTHORIZATION, **AVG**

B

BEFORE, **BEGIN**, BERNOULLI, **BETWEEN**, BIGINT, BINARY, BIT, BLOB, BOOLEAN, **BOTH** BREADTH, **BY**

C

C, **CALL**, CALLED, CARDINALITY, **CASCADE**, **CASCADED**, **CASE**, **CAST**, **CATALOG**, CATALOG_NAME,
CEIL, **CEILING**, CENTURY, **CHAIN**, CHAR, CHARACTER, CHARACTERISTICTS, **CHARACTERS**,
CHARACTER_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_NAME, CHARACTER_SET_SCHEMA,
CHAR_LENGTH, **CHECK**, CLASS_ORIGIN, **CLOB**, **CLOSE**, **COALESCE**, COBOL, **COLLATE**, **COLLATION**,
COLLATION_CATALOG, COLLATION_NAME, COLLATION_SCHEMA, **COLLECT**, **COLUMN**, COLUMN_NAME,
COMMAND_FUNCTION, COMMAND_FUNCTION_CODE, **COMMIT**, COMMITTED,CONDITION, CONDITION_NUMBER,
CONNECT, **CONNECTION**, CONNECTION_NAME, **CONSTRAINT**, **CONSTRAINTS**, CONSTRAINT_CATALOG,
CONSTRAINT_NAME, CONSTRAINT_SCHEMA, **CONSTRUCTOR**, CONTAINS, CONTINUE, **CONVERT**, **CORR**,
CORRESPONDING, **COUNT**, **COVAR_POP**, **COVAR_SAMP**, **CREATE**, **CROSS**, **CUBE**, **CUME_DIST**, **CURRENT**,
CURRENT_CATALOG, **CURRENT_DATE**, CURRENT_DEFAULT_TRANSFORM_GROUP, CURRENT_PATH, CURRENT_ROLE,
CURRENT_SCHEMA, **CURRENT_TIME**, **CURRENT_TIMESTAMP**, CURRENT_TRANSFORM_GROUP_FOR_TYPE,
CURRENT_USER, **CURSOR**, CURSOR_NAME, **CYCLE**, **DATA**, **DATABASE**, DATE, DATETIME_INTERVAL_CODE

D

DATETIME_INTERVAL_PRECISION, DAY, **DEALLOCATE**, DEC, DECADE, DECIMAL, **DECLARE**, **DEFAULT**,
DEFAULTS, DEFERRABLE, **DEFERRED**, DEFINED, DEFINER, DEGREE, **DELETE**, **DENSE_RANK**, **DEPTH**, DEREF,
DERIVED, **DESC**, **DESCRIBE**, DESCRIPTION, **DESCRIPTOR**, **DETERMINISTIC**, **DIAGNOSTICS**, **DISALLOW**, **DISCONNEC**
T,
DISPATCH, **DISTINCT**, **DOMAIN**, **DOUBLE**, DOW, DOY, **DROP**, DYNAMIC, DYNAMIC_FUNCTION, DYNAMIC_FUNCTION_C
ODE

E

©2013-2019 Tencent Cloud. All rights reserved.

```
EACH, ELEMENT, ELSE, END, END-EXEC, EPOCH, EQUALS, ESCAPE, EVERY, EXCEPT, EXCEPTION,
EXCLUDE, EXCLUDING, EXEC, EXECUTE, EXISTS, EXP, EXPLAIN, EXTEND, EXTERNAL, EXTRACT
```

F

```
FALSE, FETCH, FILTER, FINAL, FIRST, FIRST_VALUE, FLOAT, FLOOR, FOLLOWING, FOR, FOREIGN,
FORTRAN, FOUND, FRAC_SECOND, FREE, FROM, FULL, FUNCTION, FUSION
```

G

```
G, GENERAL, GENERATED, GET, GLOBAL, GO, GOTO, GRANT, GRANTED, GROUP, GROUPING
```

H

```
HAVING, HIERARCHY, HOLD, HOUR
```

I

```
IDENTITY, IMMEDIATE, IMPLEMENTATION, IMPORT, IN, INCLUDING, INCREMENT, INDICATOR,
INITIALLY, INNER, INOUT, INPUT, INSENSITIVE, INSERT, INSTANCE, INSTANTIABLE, INT,
INTEGER, INTERSECT, INTERSECTION, INTERVAL, INTO, INVOKER, IS, ISOLATION
```

J

```
JAVA, JOIN
```

K

```
K, KEY, KEY_MEMBER, KEY_TYPE
```

L

```
LABEL, LANGUAGE, LARGE, LAST, LAST_VALUE, LATERAL, LEADING, LEFT, LENGTH, LEVEL,
LIBRARY, LIKE, LIMIT, LN, LOCAL, LOCALTIME, LOCALTIMESTAMP, LOCATOR, LOWER
```

M

©2013-2019 Tencent Cloud. All rights reserved.

M, MAP, MATCH, MATCHED, MAX, MAXVALUE, MEMBER, **MERGE**, MESSAGE_LENGTH, MESSAGE_OCTET_LENGTH, MESSAGE_TEXT, METHOD, **MICROSECOND**, MILLENNIUM, **MIN**, **MINUTE**, **MINVALUE**, **MOD**, MODIFIES, **MODULE**, **MONTH**, MORE, **MULTISET**, MUMPS

N

NAME, NAMES, NATIONAL, NATURAL, NCHAR, NCLOB, NESTING, NEW, NEXT, NO, NONE, NORMALIZE, NORMALIZED, NOT, NULL, NULLABLE, NULLIF, NULLS, NUMBER, NUMERIC

O

OBJECT, OCTETS, OCTET_LENGTH, OF, OFFSET, OLD, ON, ONLY, OPEN, OPTION, OPTIONS, OR, ORDER, ORDERING, ORDINALITY, OTHERS, OUT, OUTER, OUTPUT, OVER, OVERLAPS, OVERLAY, OVERRIDING

P

PAD, PARAMETER, PARAMETER_MODE, PARAMETER_NAME, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_NAME, PARAMETER_SPECIFIC_SCHEMA, PARTIAL, PARTITION, PASCAL, PASSTHROUGH, PATH, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK, PLACING, PLAN, PLI, POSITION, POWER, PRECEDING, PRECISION, **PREPARE**, **PRESERVE**, PRIMARY, **PRIOR**, **PRIVILEGES**, **PROCEDURE**, **PUBLIC**

Q

QUARTER

R

RANGE, RANK, READ, READS, REAL, RECURSIVE, REF, REFERENCES, REFERENCING, REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE, REGR_SXX, REGR_SXY, REGR_SYY, RELATIVE, **RELEASE**, REPEATABLE, **RESET**, RESTART, RESTRICT, **RESULT**, **RETURN**, RETURNED_CARDINALITY, RETURNED_LENGTH, RETURNED_OCTET_LENGTH, RETURNED_SQLSTATE, **RETURNS**, **REVOKE**, **RIGHT**, **ROLE**, **ROLLBACK**
,
ROLLUP, ROUTINE, ROUTINE_CATALOG, ROUTINE_NAME, ROUTINE_SCHEMA, **ROW**, **ROWS**, **ROW_COUNT**, ROW_NUMBER

S

SAVEPOINT, SCALE, **SCHEMA**, SCHEMA_NAME, **SCOPE**, SCOPE_CATALOGS, SCOPE_NAME, SCOPE_SCHEMA, **SCROLL**, **SEARCH**, **SECOND**, **SECTION**, **SECURITY**, **SELECT**, **SELF**, SENSITIVE, **SEQUENCE**, **SERIALIZABLE**, **SERVER**, SERVER_NAME, **SESSION**, **SESSION_USER**, **SET**, **SETS**, SIMILAR, SIMPLE, **SIZE**, SMALLINT, **SOME**, **SOURCE**, **SPACE**, SPECIFIC, SPECIFICTYPE, SPECIFIC_NAME, **SQL**, SQLEXCEPTION, **SQLSTATE**,

©2013-2019 Tencent Cloud. All rights reserved.

```
SQLWARNING, SQL_TSI_DAY, SQL_TSI_FRAC_SECOND, SQL_TSI_HOUR, SQL_TSI_MICROSECOND, SQL_TSI_MINUTE,
SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_SECOND, SQL_TSI_WEEK, SQL_TSI_YEAR, SQRT, START,
STATE, STATEMENT, STATIC, STDDEV_POP, STDDEV_SAMP, STREAM, STRUCTURE, STYLE, SUBCLASS_ORIGIN,
SUBMULTISET, SUBSTITUTE, SUBSTRING, SUM, SYMMETRIC, SYSTEM, SYSTEM_USER
```

T

```
TABLE, TABLESAMPLE, TABLE_NAME, TEMPORARY, THEN, TIES, TIME, TIMESTAMP, TIMESTAMPADD,
TIMESTAMPDIFF, TIMEZONE_HOUR, TIMEZONE_MINUTE, TINYINT, TO, TOP_LEVEL_COUNT, TRAILING,
TRANSACTION, TRANSACTIONS_ACTIVE, TRANSACTIONS_COMMITTED, TRANSACTIONS_ROLLED_BACK,
TRANSFORM, TRANSFORMS, TRANSLATE, TRANSLATION, TREAT, TRIGGER, TRIGGER_CATALOG,
TRIGGER_NAME, TRIGGER_SCHEMA, TRIM, TRUE, TYPE
```

U

```
UESCAPE, UNBOUNDED, UNCOMMITTED, UNDER, UNION, UNIQUE, UNKNOWN, UNNAMED, UNNEST,
UPDATE, UPPER, UPSERT, USAGE, USER, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_CODE,
USER_DEFINED_TYPE_NAME, USER_DEFINED_TYPE_SCHEMA, USING
```

V

```
VALUE, VALUES, VARBINARY, VARCHAR, VARYING, VAR_POP, VAR_SAMP, VERSION, VIEW
```

W

```
WATERMARK, WEEK, WHEN, WHENEVER, WHERE, WIDTH_BUCKET, WINDOW, WITH, WITHIN, WITHOUT,
WORK, WRAPPER, WRITE
```

X

```
XML
```

Y

```
YEAR
```

Z

```
ZONE
```

©2013-2019 Tencent Cloud. All rights reserved.

Note:

In the Processing Time mode (where WATERMARK FOR is not used to define the timestamp field of the source), do not use PROCTIME as column name to avoid naming conflicts with system-generated timestamps.

©2013-2019 Tencent Cloud. All rights reserved.