Stream Compute Service

ETL Developer Guide

Product Documentation

Contents

ETL Developer Guide

Overview

Last updated：2023-11-08 14:22:35

ETL is the process of extracting data from a business system, cleaning and transforming the data, and then loading the data to a data warehouse. It aims to consolidate and standardize raw data to facilitate the decision-making of enterprises. An ETL job collects data from a data source, transforms the data or adds information to the data, and loads the results to a data sink. You don't even need to know programming languages to start an ETL job. Just select a data source and a sink and configure field mappings based on your business logic.

This section shows you how to develop an ETL job with a private cluster. It includes the following documents:

Glossary

MySQL Sources

MySQL Sinks

PostgreSQL Sinks

ClickHouse Sinks

Elasticsearch Sinks

Glossary

Last updated：2023-11-08 16:21:05

The table below lists some common terms for ETL jobs.

Term	Description
Stream computing	Stream computing is the computing of stream data. It reads data in stream form from one or more data sources, efficiently computes the continuous data streams using multiple operators of the engine, and outputs the results to different sinks such as message queues, databases, data warehouses, and storage services.
Data source	The source that continuously generates data for stream computing.
Data sink	The destination of the results of stream computing.
Schema	The structure information of a table, such as column headings and types. In the context of PostgreSQL, a schema is smaller than a database and larger than a table. It can be seen as a namespace inside a database.
MySQL	A common type of database, which can be used as the data source or sink of an ETL job.
PostgreSQL	A type of relational database similar to MySQL.
ClickHouse	A columnar database management system (DBMS) for online analytical processing (OLAP). It can be used as the data sink of an ETL job.
Elasticsearch	A real-time search and data analytics engine.
Field mapping	Field mapping is the process of extracting data from a data source, computing and cleansing the data, and then loading the data to the sink.
Constant field	You can input a custom constant field to the data source and data sink.
Calculated field	You can perform value conversion or calculation on fields extracted from a data sink using the built-in functions of Stream Compute Service.

Connectors
MySQL Sources

Last updated：2023-11-08 16:21:31

Overview

A MySQL data source can read all or new data from the MySQL database and supports exactly-once semantics. It uses Debezium for change data capture (CDC). It works as follows:
1. Acquires a global read lock to prohibit write operations by other database clients.
2. Starts a repeatable read transaction to ensure that data is always read from the same checkpoint.
3. Reads the current binlog location.
4. Reads the schema of the database and table.
5. Releases the global read lock to allow write operations by other database clients.
6. Scans the table and, after all data is read, obtains the changes made after the binlog location in step 3.
Records the binlog location as checkpoints are created regularly during the execution of the job, so that in case of a crash, processing can resume from the last recorded binlog location. This ensures exactly-once semantics.

Type mapping

MySQL Field Type	**Flink Field Type**
TINYINT	TINYINT
SMALLINT	SMALLINT
TINYINT UNSIGNED	
INT	INT
MEDIUMINT	
SMALLINT UNSIGNED	
BIGINT	BIGINT
INT UNSIGNED	
BIGINT UNSIGNED	DECIMAL(20, 0)

FLOAT	FLOAT
DOUBLE	DOUBLE
DOUBLE PRECISION	
NUMERIC(p, s)	DECIMAL(p, s)
DECIMAL(p, s)	
BOOLEAN	BOOLEAN
TINYINT(1)	
DATE	DATE
TIME [(p)]	TIME [(p)] [WITHOUT TIMEZONE]
DATETIME [(p)]	TIMESTAMP [(p)] [WITHOUT TIMEZONE]
TIMESTAMP [(p)]	TIMESTAMP [(p)]
TIMESTAMP [(p)] WITH LOCAL TIME ZONE	
CHAR(n)	STRING
VARCHAR(n)	
TEXT	
BINARY	BYTES
VARBINARY	
BLOB	

Notes

User permissions

The user of the source database must have the following permissions: SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT, SELECT, and RELOAD.

Database parameters

The running value of `binlog_row_image` should be `FULL` .

WITH parameters

MySQL data sources are developed based on MySQL CDC. The two have identical WITH parameters. For details on how to configure them, see MySQL CDC.

MySQL Sinks

Last updated：2023-11-08 16:19:33

Overview

A MySQL data sink can write data to the MySQL database.

Notes

Primary key

All data generated by an ETL data source table is UPSERT data, so it's **important** that you correctly define the primary key for the MySQL database table.

WITH parameters

MySQL data sinks are developed based on JDBC and support the following sink-related configurations:

Parameter	Required	Default Value	Description
sink.buffer-flush.max-rows	No	100	The maximum size of buffered records before flush. If this is set to `0`, data will not be buffered.
sink.buffer-flush.interval	No	1s	The maximum interval (ms) between flushes. **If `sink.buffer-flush.max-rows` is `0`, and this parameter is not, buffered actions will be processed asynchronously.**
sink.max-retries	No	3	The maximum number of retries allowed when a write operation fails.

PostgreSQL Sinks

Last updated：2023-11-08 16:20:19

Overview

A PostgreSQL data sink can write data to the PostgreSQL database.

Notes

Primary key

All data generated by an ETL data source table is UPSERT data, so it's **important** that you correctly define the primary key for the PostgreSQL database table.
The primary key defined for the data sink table **must** be the same as that defined for the physical table. Otherwise, an error may occur after the task is started.

WITH parameters

PostgreSQL data sinks are developed based on JDBC and support the following sink-related configurations:

Parameter	Required	Default Value	Description
sink.buffer-flush.max-rows	No	100	The maximum size of buffered data records before flush. If this is set to `0`, no data will be buffered.
sink.buffer-flush.interval	No	1s	The maximum interval (ms) between flushes. **If `sink.buffer-flush.max-rows` is `0`, and this parameter is not, buffered actions will be processed asynchronously.**
sink.max-retries	No	3	The maximum number of retries allowed when a write operation fails.

ClickHouse Sinks

Last updated：2023-11-08 16:20:02

Overview

A ClickHouse data sink can write data to ClickHouse.

Note

The engine of ClickHouse data sink tables must be CollapsingMergeTree.

Mapping of common data types

For the data types supported by ClickHouse, see ClickHouse Data Types. The table below lists some of the common data types and their counterparts in Flink.

Flink	ClickHouse
VARCHAR	String/FixedString(N)
STRING	String/FixedString(N)
BOOLEAN	There isn't a dedicated ClickHouse type for boolean. You can use `UInt8` to store boolean data, limiting the valid values to `0` and `1`, or use the string type, limiting the valid values to `true` and `false`.
DECIMAL	Decimal32(S)/Decimal64(S)/Decimal128(S)
TINYINT	Int8
SMALLINT	Int16
INTEGER	Int32
BIGINT	Int64
FLOAT	Float32
DOUBLE	Float64
DATE	Date
TIMESTAMP	DateTime
TIMESTAMP WITH LOCAL	DateTime. Example: DateTime64(3, 'Asia/Shanghai').

TIME ZONE	

Notes

Primary key

To ensure that update and delete operations are synced as expected, make sure you define the primary key correctly using the CREATE TABLE statement.

Collapsed fields

The CollapsingMergeTree engine of ClickHouse adds the logic of rows collapsing to data parts merge algorithm. You can specify collapsed fields using `ENGINE = CollapsingMergeTree(Sign)` when creating a table. To learn more, see the ClickHouse document.

WITH parameters

ClickHouse data sinks are developed based on the data warehouse ClickHouse. The two have identical WITH parameters. For details, see ClickHouse.

Elasticsearch Sinks

Last updated：2023-11-08 16:20:45

Overview

An Elasticsearch data sink can write data to Elasticsearch.

Note

Currently, Elasticsearch data sinks only support Elasticsearch 6 and Elasticsearch 7.

Mapping of common data types

For the data types supported by Elasticsearch, see Elasticsearch Data Types. The table below lists some of the common data types and their counterparts in Flink.

Flink	Elasticsearch
text	STRING
match_only_text	STRING
binary	STRING
keyword	STRING
wildcard	STRING
search_as_you_type	STRING
ip	STRING
short	SMALLINT
integer	INT
long	BIGINT
unsigned_long	BIGINT
float	FLOAT
half_float	FLOAT
double	DOUBLE

boolean	BOOLEAN
date	TIMESTAMP(3)
date_nanos	TIMESTAMP(6)

Note

Data types not listed in the above table are not supported currently.

Notes

Primary key

A primary key is required for Elasticsearch. The field set as the primary key will be written into `_id`, and data with the same ID will be overwritten.

Version differences

Please note that if you use Elasticsearch 6, you need to configure `document-type`. This is not necessary with Elasticsearch 7.

WITH parameters

Elasticsearch data sinks are developed based on the data analytics engine Elasticsearch. The two have identical WITH parameters. For details, see Elasticsearch Service.