cloudera®

Cloudera Connector for Teradata

Important Notice

Cloudera, Inc.
1001 Page Mill Road, Bldg 3
Palo Alto, CA 94304
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Connectors 1.x
Date: November 6, 2017

Table of Contents

Cloudera Connector Powered by Teradata User Guide.....4

Version Scheme for Teradata Connector.....4

Cloudera Connector Powered by Teradata Release Notes.....4

New Features in Cloudera Connector Powered by Teradata.....4

Limitations for Cloudera Connector Powered by Teradata.....5

Known Issues and Workarounds.....6

Getting Support.....6

Prerequisites for Teradata Connector.....6

Installing the Teradata Connector.....6

Installation with CDH 5 and Cloudera Manager 5.....7

Installation without Cloudera Manager.....7

Upgrading the Teradata Connector.....8

Upgrading with CDH 5 and Cloudera Manager 5.....8

Upgrading without Cloudera Manager.....8

Using the Cloudera Connector Powered by Teradata.....9

Input Methods.....11

Output Methods.....12

Uninstalling Teradata Connectors.....12

Uninstallation with CDH 5 and Cloudera Manager 5.....13

Uninstallation without Cloudera Manager 5.....13

Cloudera Connector Powered by Teradata User Guide

Cloudera Connector Powered by Teradata provides high-speed data transfer between Teradata and CDH.

This connector allows various Sqoop tools, such as `sqoop-import` and `sqoop-export`, to operate in highly efficient direct modes, and expose options that are specific to Teradata.

This guide describes how to install, configure, and use the connector in a Sqoop 1 installation and provides reference information for connector operation. This guide is intended for:

- System and application programmers
- System administrators
- Database administrators
- Data analysts
- Data engineers

Version Scheme for Teradata Connector

This topic describes the versioning scheme used for Cloudera Connector Powered by Teradata. The version string consists of the following parts:

`<major_Sqoop_version>.<Connector_version>c<major_CDH_version>`

- `<major_Sqoop_version>`: Identifies the major version of Sqoop for which the connector has been compiled. Currently, its only values are `1` and `2`.
- `<Connector_version>`: Revision of the connector itself, possible values are `1`, `2`, `3`, `4`....
- `<major_CDH_version>`: The major CDH version for which the connector has been compiled and tested.

For example:

- 1.6c5 - Sixth revision of a Sqoop 1-based connector that is compatible with CDH 5.

Cloudera Connector Powered by Teradata Release Notes

This section summarizes the high level changes and most important new features in the Cloudera Connectors for Teradata.

New Features in Cloudera Connector Powered by Teradata

The following new features are included in Cloudera Connector Powered by Teradata.

New Features in Cloudera Connector Powered by Teradata Version 1.6.1c5

- Adds support for SLES 12.

New Features in Cloudera Connector Powered by Teradata Version 1.6c5

- Upgrades the JDBC driver to version 15.10.00.22 and the TDCH library to version 1.5.0. These libraries contain several bug fixes and improvements.
- Adds the `--schema` argument, used to override the `<td-instance>` value in the connection string of the Sqoop command. For example, if the connection string in the Sqoop command is `jdbc:teradata://<td-host>/DATABASE=database1`, but you specify *--schema database2*, your data is imported from database2 and not database1. If the connection string does not contain the DATABASE parameter — for example jdbc:teradata://<td-host>/CHARSET=UTF8) — you can also use the `--schema database` argument to have Sqoop behave as if you specified the `jdbc:teradata://<td-host>/DATABASE=databasename,CHARSET=UTF8` connection string.

New Features in Cloudera Connector Powered by Teradata Version 1.5c5

New features:

- Fixed compatibility issue with CDH 5.5.0 and higher.

New Features in Cloudera Connector Powered by Teradata Versions 1.4c5

New features:

- Added support for JDK 8.
- Added `--error-database` option.
- Added ability to specify format of date, time, and timestamp types when importing into CSV.
- Import method `split.by.amp` now supports views.
- Upgraded Teradata connector for Hadoop to version 1.3.4.

New Features and Changes in Cloudera Connector Powered by Teradata 1.3c5

New features:

- Upgraded Teradata Connector for Hadoop to version 1.3.3.
- Parcel distribution now contains Teradata JDBC driver; manual download no longer required.
- Added support for query import into Avro file format.

Changes:

- Export method `multiple.fastload` has been removed.

New Features in Cloudera Connector Powered by Teradata Versions 1.2c5

New features:

- Upgraded Teradata Connector for Hadoop to version 1.2.1.
- Added support for Avro.
- Added support for Incremental import.
- Added support for `--where` argument.
- Added support for Hive import.
- Added support for Importing all tables using `import-all-tables`.
- Added support for Query Bands.
- Added new import method `split.by.amp` (supported only on Teradata 14.10 and higher).

New Features in Cloudera Connector Powered by Teradata Version 1.0.0

This is the first release of this new connector. This connector features:

- Support for secondary indexes.
- Especially fast performance in most cases.

Limitations for Cloudera Connector Powered by Teradata

Limitations for Cloudera Connector Powered by Teradata has the following functional limitations.

- Does not support HCatalog.
- Does not support import into HBase.
- Does not support upsert functionality (parameter `--update-mode allowinsert`).
- Does not support the `--boundary-query` option.
- Does not support Parquet file format.
- Does not support export to Teradata VIEWs.
- Does not support Kerberos authentication.
- By default speculative execution is disabled for the Teradata Connector. This avoids placing redundant load on the Teradata database.

Known Issues and Workarounds

None.

Getting Support

Support for the Cloudera Connector for Teradata is available through Cloudera Enterprise Support. Refer to Cloudera Support for more details.

Prerequisites for Teradata Connector

To use the Cloudera Connector Powered by Teradata, you must have a functioning CDH installation, including Sqoop components. Depending on how Sqoop is installed, you may need administrative privileges to create or modify configuration files.

The Teradata connector uses catalog tables and views to look up metadata information. Therefore, the user making the connection must have the SELECT privilege on the DBC database. Check with your Teradata administrators or operators if you are not sure whether you have the required privileges. You must have SELECT privileges on at least one of the following DBC database object types:

- DBC.columns
- DBC.databases
- DBC.tables
- DBC.indices

Depending on the input method used, the Teradata connector might need to create temporary tables or temporary views to import data. Check with your Teradata administrators or operators to determine if you have the required privileges.

The Cloudera Connector Powered by Teradata requires the following additional permissions to use `*.fastload` data methods.

- DBC.table_levelconstraints
- DBC.triggers
- DBC.tvm
- DBC.dbase
- DBC.referencingtbls

For more information about how to install, configure, and use Sqoop, see the Sqoop documentation in the CDH 5 Installation Guide.

Cloudera Connector powered by Teradata versions 1.2c5, 1.3c5, 1.4c5, 1.5c5

- Teradata 13.00 and higher
- CDH 5.0 and higher
- Sqoop 1.4 and higher, but not compatible with Sqoop2

Cloudera Connector powered by Teradata versions 1.0.0, 1.2c4, 1.3c4, 1.4c4

- Teradata 13.00 and higher
- CDH 4.2 and higher
- Sqoop 1.4 and higher, but not compatible with Sqoop2

Installing the Teradata Connector

Use one of the following methods to install the Sqoop connector for Teradata:

- If you have a CDH 5 cluster managed by Cloudera Manager 5, see Installation with CDH 5 and Cloudera Manager 5 on page 7.
- If your cluster is not managed by Cloudera Manager, see Installation without Cloudera Manager on page 7.

Installation with CDH 5 and Cloudera Manager 5

Step 1: Adding the Sqoop Client Gateway

The Sqoop1 Client Gateway sets up and manages the connector configuration for the hosts where you execute Sqoop1 commands. If you do not already have the Sqoop1 Client Gateway deployed in your cluster, deploy it before proceeding.

Important: The Sqoop 1 Client Gateway is required for the Teradata Connector to work correctly. Cloudera recommends installing the Sqoop 1 Client Gateway role on any host used to execute the Sqoop CLI. If you do not already have the Sqoop Client service running on your cluster, see Managing the Sqoop 1 Client for instructions on how to add the service using the Cloudera Manager Admin Console.

Step 2: Download, Distribute, and Activate the Sqoop Parcels

Parcels for Sqoop connectors are prefixed by SQOOP_, followed by the name of the connector.

Follow the instructions in Managing Parcels to download, distribute, and activate Sqoop parcels.

Installation without Cloudera Manager

Note: For the sample file and directory names used in the instructions below, replace X with the CDH major version you are using. For example, for a CDH 5 setup, `sqoop-connector-teradata-1.2cX` should be replaced by `sqoop-connector-teradata-1.2c5`.

1. Install the Sqoop connector by opening the distribution archive in a convenient location such as `/usr/lib`. Opening the distribution creates a directory that contains the jar file of the compiled version of the connector. Note the path to this jar file. The directory that is created when the file is expanded varies according to which connector you are using. Examples of typical resulting paths include:
   - Cloudera Connector Powered by Teradata 1.2cX: `/usr/lib/sqoop-connector-teradata-1.2cX/sqoop-connector-teradata-1.2cX.jar`
2. Copy the Teradata JDBC drivers (`terajdbc4.jar` and `tdgssconfig.jar`) to the `lib` directory of the Sqoop installation. You can obtain these drivers from the Teradata download website: http://downloads.teradata.com/download/connectivity/jdbc-driver. Without these drivers, the connector will not function correctly.
3. Confirm that the `managers.d` directory exists in the Sqoop configuration directory.

   Note: Depending on how Sqoop is installed, its configuration directory can be in `/etc/sqoop/conf`, `/usr/lib/sqoop/conf`, or elsewhere if Sqoop was installed using the tarball distribution.

   If the `managers.d` directory does not exist, create it and ensure that the directory permissions are set to `755`.
4. Create a text file in the `managers.d` directory with a descriptive name such as `cldra_td_connector`. Ensure the file permissions are set to `644`.
5. The `cldra_td_connector` file must have the connector class name followed by the complete path to the directory where the connector jar is located.

   For example, for the Cloudera Connector powered by Teradata 1.2cX:

```
com.cloudera.connector.teradata.TeradataManagerFactory= \
/usr/lib/sqoop-connector-teradata-1.2cX/sqoop-connector-teradata-1.2cX.jar
```

> **Note:** The preceding command is shown on two lines but must be entered on a single line.

`TeradataManagerFactory` acts as a single point of delegation for invoking the connector bundled with this distribution. Alternatively, you can specify `TeradataManagerFactory` inside a `sqoop-site.xml` file, which must be inside a classpath directory.

If you are using Cloudera Connector powered by Teradata, use the following:

```
<configuration>
  <property>
    <name>sqoop.connection.factories</name>
    <value>com.cloudera.connector.teradata.TeradataManagerFactory</value>
  </property>
<configuration>
```

This configures a Sqoop action to use the Teradata connector inside Oozie.

Upgrading the Teradata Connector

Use these instructions if you are upgrading one of the connectors to a newer version (for example, if you need to upgrade Cloudera Connector Powered by Teradata from version 1.2c4 to 1.3c4).

Upgrading with CDH 5 and Cloudera Manager 5

Step 1: Distributing the Sqoop Connector Parcels

1. In the Cloudera Manager Admin Console, click **Hosts** in the top navigation bar and then go to the **Parcels** tab. Parcels for the Sqoop connectors are listed on this page, prefixed by "SQOOP_", followed by the name of the connector.
2. Click **Download** for the connectors you want to install.
3. After the parcels have been downloaded, click **Distribute** to distribute and unpack the connectors on all hosts in your Hadoop cluster.
4. After the parcels have been distributed, click **Activate** to make them available to the cluster. Sqoop connectors are listed as **Activated** on the **Parcels** page. You must redeploy the client configuration (Step 3) for activation to take effect.

Step 2: Deploying Client Configuration Files

1. In the Cloudera Manager Admin Console, go to the Sqoop Client service.
2. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
3. Click **Deploy Client Configuration** to confirm redeployment of the client configuration.

Upgrading without Cloudera Manager

> **Note:** For the sample file and directory names used in the instructions below, replace X with the CDH major version you are using. For example, for a CDH 5 setup, `sqoop-connector-teradata-1.2cX` should be replaced by `sqoop-connector-teradata-1.2c5`. Be sure to update all existing paths to new paths for your updated software.

1. Install the Sqoop connector by opening the distribution archive in a convenient location such as `/usr/lib`. Opening the distribution creates a directory that contains the jar file of the compiled version of the connector. Note the path to this jar file. The directory that is created when the file is expanded varies according to which connector you are using. Examples of typical resulting paths include:

- Cloudera Connector Powered by Teradata 1.2cX:
  `/usr/lib/sqoop-connector-teradata-1.2cX/sqoop-connector-teradata-1.2cX.jar`

2. Copy the Teradata JDBC drivers (`terajdbc4.jar` and `tdgssconfig.jar`) to the `lib` directory of the Sqoop installation. You can obtain these drivers from the Teradata download website: http://downloads.teradata.com/download/connectivity/jdbc-driver. Without these drivers, the connector will not function correctly.
3. Confirm that the `managers.d` directory exists in the Sqoop configuration directory.

   Note: Depending on how Sqoop is installed, its configuration directory can be in `/etc/sqoop/conf`, `/usr/lib/sqoop/conf`, or elsewhere if Sqoop was installed using the tarball distribution.

   If the `managers.d` directory does not exist, create it and ensure that the directory permissions are set to `755`.
4. Create a text file in the `managers.d` directory with a descriptive name such as `cldra_td_connector`. Ensure the file permissions are set to `644`.
5. The `cldra_td_connector` file must have the connector class name followed by the complete path to the directory where the connector jar is located.

   For example, for the Cloudera Connector powered by Teradata 1.2cX:

```
com.cloudera.connector.teradata.TeradataManagerFactory= \
/usr/lib/sqoop-connector-teradata-1.2cX/sqoop-connector-teradata-1.2cX.jar
```

   Note: The preceding command is shown on two lines but must be entered on a single line.

   `TeradataManagerFactory` acts as a single point of delegation for invoking the connector bundled with this distribution. Alternatively, you can specify `TeradataManagerFactory` inside a `sqoop-site.xml` file, which must be inside a classpath directory.

   If you are using Cloudera Connector Powered by Teradata, use the following:

```
<configuration>
  <property>
    <name>sqoop.connection.factories</name>
    <value>com.cloudera.connector.teradata.TeradataManagerFactory</value>
  </property>
<configuration>
```

   This configures a Sqoop action to use the Teradata connector inside Oozie.

Using the Cloudera Connector Powered by Teradata

After you have installed the connector and copied the JDBC drivers for Teradata to the `lib` directory of the Sqoop installation, use this connector by invoking Sqoop tools with the appropriate connection string.

The connection string format is `jdbc:teradata://<td-host>/DATABASE=<td-instance>`:

- `<td-host>` is the hostname of the machine on which the Teradata server runs.
- `<td-instance>` is the Teradata database instance name.

For example, the following command invokes the Sqoop import tool with three mappers:

```
$ sqoop import --connect jdbc:teradata://localhost/DATABASE=sqooptest \
--username sqooptest --password xxxxx --table MY_TABLE --num-mappers 3 \
--target-dir /user/sqooptest/MY_TABLE
```

The following command invokes the Sqoop export tool with three mappers:

```
$ sqoop export --connect jdbc:teradata://localhost/DATABASE=sqooptest \
--username sqooptest --password xxxxx --export-dir /user/sqooptest/MY_TABLE \
--table MY_TABLE_TARGET --num-mappers 3
```

You can control the behavior of the connector by using extra arguments. Extra arguments must appear at the end of the command. Use a double-dash separator (--) to indicate the end of the standard arguments and the beginning of the extra arguments. For example, the following command uses the double-dash (--) separator (in bold for emphasis) to separate the standard arguments `--table` and `--num-mappers` from the extra arguments `--input-method` and --query-band:

```
$ sqoop ... --table MY_TABLE --num-mappers 3 -- --input-method split.by.amp --query-band
 DC=BP\;Location=Europe
```

Table 1: Teradata Connector Feature Support

Parameter	Tool	Description
`--staging-table`	import and export	Override the default staging table name. This parameter applies only if staging tables are used during data transfer.
`--staging-database`	import and export	Override the default staging database name. This parameter applies only if staging tables are used during the data transfer.
`--staging-force`	import and export	Force the connector to create the staging table if the input/output method supports staging tables.
`--input-method`	import	Specify the input method used to transfer data from Teradata to Hadoop.
`--output-method`	export	Specify the output method used to transfer data from Hadoop to Teradata.
`--batch-size`	import and export	Specify the number of rows processed together in one batch.
`--access-lock`	import	Improve concurrency. When used, the import job is not blocked by concurrent accesses to the same table.
`--query-band`	import and export	Allow arbitrary query bands to be set for all queries that are run by the connector. The expected format is a semicolon-separated key=value pair list. A final semicolon is required after the last key=value pair. For example, `Data_Center=XO;Location=Europe;`.
`--error-table`	export (only for `internal.fastload`)	Specify a prefix for created error tables.
`--error-database`	export (only for `internal.fastload`)	Override the default error database name.
`--fastload-socket-hostname`	export (only for `internal.fastload`)	Hostname or IP address of the host on which you are running Sqoop, one that is visible from the Hadoop cluster. The connector can autodetect the interface. This parameter overrides the autodection routine.

Parameter	Tool	Description
`--keep-staging-table`	import	By default, the connector drops all automatically created staging tables when export fails. This option leaves the staging tables with partially imported data in the database.
`--num-partitions-for-staging-table`	import (only for `split.by.partition`)	Number of partitions to use for the automatically created staging table. The connector automatically generates the value based on the number of mappers used.
`--skip-xviews`	import and export	By default, the connector uses Teradata system views to obtain metadata. With this parameter, the connector switches to XViews instead.
`--date-format`	import and export	Use custom format for columns of `date` type. The parameter uses SimpleDateFormat formatting options.
`--time-format`	import and export	Use custom format for columns of `time` type. The parameter uses SimpleDateFormat formatting options.
`--timestamp-format`	import and export	Use custom format for columns of `timestamp` type. The parameter uses SimpleDateFormat formatting options.

Input Methods

Cloudera Connector Powered by Teradata supports the following methods for importing data from Teradata to Hadoop:

- `split.by.amp`
- `split.by.value`
- `split.by.partition`
- `split.by.hash`

`split.by.amp` Method

This optimal method retrieves data from Teradata. The connector creates one mapper per available Teradata AMP, and each mapper subsequently retrieves data from each AMP. As a result, no staging table is required. This method requires Teradata 14.10 or higher.

`split.by.value` Method

This method creates input splits as ranges on the split by column (usually the table's primary key). Each split is subsequently processed by a single mapper to transfer the data using SELECT queries. All splits can access all AMPs to retrieve data, so you should set the number of mappers between 20 and 30 because there is a limit for all-AMP concurrently running operations on the Teradata appliance. Ensure that users transferring data have sufficient spool space available for the SELECT queries.

`split.by.partition` Method

This method is preferred for extracting a large amount of data from the Teradata system. Behavior of this method depends whether source table is partitioned or not.

`split.by.hash` Method

This input method is similar to the `split.by.partition` method. Instead of directly operating on value ranges of one column, this method operates on the hash of the column. You can use importing by hash to extract data in situations where `split.by.value` and `split.by.partition` are not appropriate. Each mapper can access all AMPs available in the system, so set the number of mappers between 20 and 30 because there is a limit for all-AMP concurrent jobs on the Teradata appliance.

The following example shows import using input method `split.by.hash`:

```
$ sqoop import --connect jdbc:teradata://localhost/DATABASE=sqooptest \
--username sqooptest --password xxxxx --table MY_TABLE --num-mappers 3 \
--target-dir /user/sqooptest/MY_TABLE - --input-method split.by.hash
```

If your input table is not partitioned, the connector creates a partitioned staging table and runs an INSERT into SELECT query to move data from the source table into the staging table. Subsequently, each mapper transfers data from one partition, resulting in a single AMP operation. With a single AMP, you can use a large number of mappers to obtain optimal performance. The amount of available permanent space must be as large as your source table and the amount of spool space required to run the SELECT queries.

If your table is already partitioned, no extra staging table is created. However, you can force the connector to re-partition your data using the `--staging-force` parameter to achieve better performance. Without forcing repartition of the data, this method opens all-AMP operation, so you should use between 20 and 30 mappers. If your source table is a PI table, and your split by column is the table's primary key, the connector creates a single AMP operation, and you can use high number of mappers.

Output Methods

Cloudera Connector Powered by Teradata supports the following output methods to export data from Hadoop to Teradata:

- `batch.insert`
- `internal.fastload`

`batch.insert` Method

This method uses JDBC batch jobs to export data to the Teradata appliance. This method should be used only when other methods are not a viable. It creates a partitioned stating table before the export job, and then subsequently each mapper transfers data from Hadoop to one partition. After all mappers end, the INSERT into SELECT statement is called to transfer the data from staging to table to final destination. Ensure that you have sufficient permanent space for two copies of your data. You also need sufficient spool space for running the INSERT into SELECT query. The number of mappers that this method can use is limited only by the number of concurrent sessions allowed on the Teradata appliance.

`internal.fastload` Method

This method requires a partitioned staging table. Data is first exported by each mapper into a different partition and then moved to the target table, using the INSERT into SELECT statement. Make sure that you have sufficient permanent and spool space to store two copies of your data and to move them from the staging table to the target table. All mappers participate in one FastLoad job coordinated by an internal protocol. This is the fastest method for exporting data from Hadoop to a Teradata appliance. Because all mappers participate in the one FastLoad job, only one Teradata utility slot is used for the export job. The number of mappers is limited only by total number of AMPs available in your system.

The Teradata server is started on the machine where the sqoop command is running, and all mappers started by this sqoop command must connect to it. Because a mapper can be run on any hosts in the cluster, all hosts must have access to the machine running the sqoop command.

Uninstalling Teradata Connectors

You can use one of the following ways to uninstall the Sqoop connectors for Teradata:

- If you have a CDH 5 cluster managed by Cloudera Manager 5, see Uninstallation with CDH 5 and Cloudera Manager 5 on page 13.
- If your cluster is not managed by Cloudera Manager, see Uninstallation without Cloudera Manager 5 on page 13.

Uninstallation with CDH 5 and Cloudera Manager 5

Perform the following steps to uninstall the Sqoop connectors for Teradata using Cloudera Manager 5:

1. **Removing the Sqoop Connector Parcels:**
   a. In the Cloudera Manager Admin Console, click **Hosts** in the top navigation bar and then go to the **Parcels** tab. Parcels for the Sqoop connectors are listed on this page, prefixed by "SQOOP_", followed by the name of the connector.
   b. The Sqoop connectors are listed as **Activated**. To deactivate a parcel, click **Actions** on an activated parcel and select **Deactivate**.
   c. To remove the parcel, click the down arrow to the right of the **Activate** button and select **Remove from Hosts**.
2. **Redeploy client configuration:**
   a. In the Cloudera Manager Admin Console, go to the Sqoop Client service.
   b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
   c. Click **Deploy Client Configuration** to confirm redeployment of client configuration.

Uninstallation without Cloudera Manager 5

To remove the Cloudera Connector for Teradata:

- Delete the `cldra_td_connector` file from `managers.d` directory located under the Sqoop configuration directory.
- Remove the files from the connectors distribution. These files might be in a location under `/usr/lib/`.
- Remove the JDBC drivers for Teradata that you copied to the `lib` directory of the Sqoop installation. Sqoop no longer needs these drivers. The Teradata JDBC drivers are named `terajdbc4.jar` and `tdgssconfig.jar`.