Teradata Connector for Hadoop

(Apr 10, 2013)

docs.hortonworks.com

Teradata Connector for Hadoop

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

Using Teradata Connector for Hadoop

Introduction

Teradata Connector for Hadoop is an implementation of a Sqoop connector that enables those conversant with the Apache Sqoop tool to transfer data between the Teradata MPP DBMS and Apache Hadoop environments.

Background

Sqoop provides facilities for bulk transfer of data between external data stores and the Hadoop environment exploiting the Map Reduce paradigm. Sqoop depends on JDBC interfaces to access the external databases.

Most of the databases also have specialized access methods for high speed bulk data transfers for efficient batch processing needs, such as backups, etc.

To accommodate the varieties of database mechanisms to facilitate bulk transfer, Sqoop provides extensible base implementations of the data transfer functions utilizing the JDBC interface that can optionally be enhanced to suit a database-specific method of data transfer.

Sqoop has the notion of *Connectors,* which contain the specialized logic to read and write to external systems. The Teradata connector is a Sqoop Connector implementation for Teradata.

Supported Features

This section describes Teradata connector features that are available in the current release.

The Teradata connector has some features that are not directly supported by the Sqoop product; you must use the Teradata connector mechanism for such features by specifying additional `-D` command line options.

For example to use RCFiles or HCatalog features which are not supported by Sqoop, a command line option exposed by the Teradata connector is needed to enable the features. The option for RCFiles is `-Dteradata.db.input.file.format=rcfile` (see Teradata Connector Options below).

The Teradata connector supports the following features:

- Import/Export tools that run Hadoop MR jobs to transfer data.
- Support for Text, Sequence and RCFiles as the source for export and target for import operations.
- Import table or query data from Teradata to:
  - an existing partitioned or non-partitioned Hive table.
  - a new partitioned or non-partitioned Hive table created by the connector.
  - an HCatalog table.

- Export data from HDFS files, Hive or HCatalog tables to empty or non-empty Teradata tables.
- Facilities for mapping schemas between Teradata and Hive/HCatalog, including necessary data type conversions.

Connector Feature Checklist

Import all tables: Supported.

Incremental import: Not supported.

BLOB and CLOB: Limited to 64 KB.

Import data to Sqoop

- **TextFormat, delimited:** Supported.
- **SequenceFile:** Supported.
- **RCFile:** Supported.

Hive arguments: Support for all standard Hive arguments. All data types except Union are supported.

Export from / import to HCatalog table: Supported.

Automatic schema mapping to/from HCatalog: Supported.

Import using a query: Supported.

Update table: Not supported.

Compression: Not supported.

Software Versions and Installation

Connector Version

This document discusses the Teradata Connector for Hadoop ("Teradata connector") built on version 1.0.4 of the Teradata Hadoop Connector, a Teradata product.

Supported Product Versions

This section lists the product versions supported in the current release of the Teradata connector.

Teradata Database Versions

The following Teradata database versions are supported:

- Teradata Database 13.00
- Teradata Database 13.10

- Teradata Database 14.00

Hive Version

- Hive 0.10

Hadoop Version

- HDP 1.2.3 (versions prior to 1.2.3 are untested, but may also work)

Sqoop Versions

- Sqoop 1.4.2

Requirements and Dependencies

System Requirements

The Teradata connector requires JRE/JDK 1.6 or later versions.

Dependencies

1. Teradata GSS Client Driver 13.0 or later versions (tdgssconfig)
2. Teradata JDBC Driver 13.0 or later versions (terajdbc)
3. Teradata Hadoop Connector 1.0.4

Installation

Installation Dependencies

Sqoop must be installed first.

Installing the Software

1. Download the tarball from hortonworks.com/download.
2. Extract the contents of the tar archive to $SQOOP_HOME/lib.

Configuration

Database Connection Credentials

Refer to Sqoop documentation for the Teradata database connection credentials.

Documentation for Sqoop version 1.4.2 is available here: http://sqoop.apache.org/docs/1.4.2/index.html.

Teradata Connector Options

The Teradata connector defines a total of 42 connector-specific options. A good selection of them is also available as Sqoop options (although not all Sqoop options are directly translatable to Teradata connector options).

Teradata connector options can be specified using any of these techniques:

- a configuration file
- `–D` command line option
- Sqoop options (where applicable)

Therefore the following precedence is established:

1. The value in the configuration file is the default.
2. If `–D` command line options are provided, they override the configuration file values.
3. If any Sqoop command line options are provided which have purposes similar to the Teradata options, they override the `–D` option and the configuration file and have the highest precedence.

As an example, if the Teradata configuration has the job type set to HCatalog and the command line option (`–D com.teradata.db.input.job.type` option) has the value set to HDFS, but the Sqoop option `––as–sequence–file` is set, then the job type will be overridden to be of type Hive.

Note

Imports and exports are defined from the Hadoop perspective, that is, an import brings data into Hadoop from the database and an export moves data out of Hadoop into the database.

Import Options

All option names below are prefixed by "**teradata.db.input.**" when specified in the configuration files or in the `–D` command line option.

For example, the `job.type` option is specified as `teradata.db.input.job.type`.

Import Option (`teradata.db.input.*`)	**Description**	**Overrriding Sqoop Option**
job.type	The type of import job. Required: no Supported values: hcat, hive, hdfs Default value: hdfs	None
file.format	The format of a to-be-imported data file in HDFS. An 'hcat' or 'hive' job type supports 'rcfile', 'sequencefile', and 'textfile' file formats; and an 'hdfs' job type supports only 'textfile' format. Required: no	–as-sequencefile –as-textfile

Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
	Supported values: rcfile, sequencefile, textfile Default value: textfile	
target.paths	The directory with which to place the imported data. It is required for a 'hdfs' job, optional for a 'hive' job, and not valid for a 'hcat' job. For a 'hive' job, either specify this or the 'target.table' parameter but not both. Required: no Supported values: string Default value: The value of property 'mapred.output.dir'	–target-dir –warehouse-dir
method	The method that the Teradata connector uses to import data from a Teradata system. Required: no Supported values: split.by.hash, split.by.partition, split.by.value Default value: split.by.hash	
num.mappers	The number of mappers for the import job. It is also the number of splits the Teradata connector will attempt to create. Required: no Supported values: an integer greater than 0 Default value: 2	-m –num-mappers
source.query	The SQL query to select data from a Teradata database; either specify this or the 'source.table' parameter, but not both. Required: no Supported values: The select SQL query (Teradata database supported)	
source.count.query	A placeholder, not currently used.	N/A
source.table	The name of the source table in a Teradata system from which the data is imported. Either specify this or the 'source.query' parameter, but not both. Required: no Supported values: string	–table
source.field.names	The names of columns to import from the source table in a Teradata system, in comma-separated format. The order of the source field names must match exactly the order of the target field names for schema mapping. This parameter must be present when the 'target.field.names' parameter is specified. If not specified, then all columns from the source table will be retrieved. Required: no Supported values: string	–columns

Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
target.database	The name of the target database in Hive or HCatalog. It is optional with a 'hive' or 'hcat' job and not valid with an 'hdfs' job. Required: no Supported values: string Default value: default	N/A
target.table	The name of the target table in Hive or HCatalog. It is required with an 'hcat' job, optional with a 'hive' job, and not valid with an 'hdfs' job. For a 'hive' job, either specify this parameter or the 'target.paths' parameter, but not both. Required: no Supported values: string	–hive-table
target.table.schema	The column schema of the target table, including the partition schema, in comma-separated format. Required: no Supported values: string	
target.partition.schema	The partition schema of the target table in Hive or HCatalog, in comma-separated format. This parameter is applicable with 'hive' job only, and 'target.table.schema' must be specified with it. Required: no Supported values: string	N/A
target.field.names	The names of fields to write to the target file in HDFS, or to the target Hive or HCatalog table, in comma separated format. The order of the target field names must match exactly the order of the source field names for schema mapping. This parameter must be provided when the 'source.field.names' parameter is specified. Required: no Supported values: string	Driven by the imported columns
batch.size	The number of rows a Teradata connector fetches each time from the Teradata system, up to a 1 MB buffer size limit. Required: no Supported values: an integer greater than 0 Default value: 10000	–fetch-size
separator	The field separator to use with the imported files. This parameter is only applicable with the 'textfile' file format. Required: no Supported values: string Default value: \t	–fields-terminated-by
split.by.column	The name of a table column to be used for splitting import tasks. It is optional with the 'split.by.hash' and 'split.by.value' methods, and not valid with the	–split-by

Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
	'split.by.partition' method. If this parameter is not specified, the first column of the table's primary key or primary index will be used. Required: no Supported values: a valid table column name	
stage.force	If set to true, then staging is used even if the source table is a PPI table. It is valid with the 'split.by.partition' method only. Required: no Supported values: true, false Default value: false	N/A
stage.database	The database which the Teradata connector uses to create a staging table. Required: no Supported values: the name of a database in the Teradata system Default value: the current logon database in the JDBC connection	N/A
stage.table.name	The name which Teradata connector uses to create a staging table in the Teradata system, if staging is required. Its length cannot exceed 20 characters. It should be used when the source Teradata table has a name exceeding 24 characters. Required: no Supported values: string	N/A
teradata.db.hive.configuration.file	The path to the Hive configuration file in the HDFS. It is required for a 'hive' or 'hcat' job launched through remote execution or on data nodes. Required: no Supported values: string	

Unsupported Import Options

This section lists the Sqoop import options that the Teradata connector does not support.

Control Options

The Teradata connector does not support these Sqoop import control options:

```
--as-avrodatafile
--append
--compression-codec
--direct
--direct-split-size
--where
--compress, -z
```

Incremental Options

The Teradata connector does not support these Sqoop incremental options:

```
--check-column
--incremental
--last-value
```

Input Parsing Options

The Teradata connector does not support these Sqoop input parsing options:

```
--input-enclosed-by
--input-escaped-by
--input-lines-terminated-by
--input-optionally-enclosed-by
```

Output Formatting Options

The Teradata connector does not support these Sqoop output formatting options:

```
--enclosed-by
--escaped-by
--lines-terminated-by
--mysql-delimiters
--optionally-enclosed-by
```

Hive Support Options

The Teradata connector does not support these Sqoop import options for Hive:

```
--hive-delims-replacement
--hive-drop-import-delims
--hive-home
--hive-overwrite
--hive-partition-key
--hive-partition-value
--map-column-hive
```

HBase Support Options

The Teradata connector does not support these Sqoop import options for HBase:

```
--column-family
--hbase-create-table
--hbase-row-key
--hbase-table
```

Data Mapping Options

The Teradata connector does not support these Sqoop data mapping options:

```
--input-null-non-string
--input-null-string
--map-column-java
--null-non-string
--null-string
```

Export Options

All option names below are prefixed by "**teradata.db.output.**" when specified in the configuration files or in the `–D` command line option.

For example, `target.table` is specified as `teradata.db.output.target.table`.

Export Option (`teradata.db.output.*`)	**Description**	**Overrriding Sqoop Option**
target.table	The name of the target table in a Teradata system. Required: yes Supported values: string	–table
job.type	The type of export job. Required: no Supported values: hcat, hive, hdfs Default value: hdfs	N/A
file.format	The format of a to-be exported data file in HDFS. An 'hcat' or 'hive' job type supports 'rcfile', 'sequencefile', and 'textfile' formats; and an 'hdfs' job type supports only 'textfile' format. Required: no Supported values: rcfile, sequencefile, textfile Default value: textfile	N/A
source.paths	The directory of to-be exported source files in HDFS. It is required for an 'hdfs' job, optional with a 'hive' job, and not valid with an 'hcat' job. For a 'hive' job, either specify this or the 'source.table' parameter but not both. Required: no Supported values: string	–export-dir
method	The method that the Teradata connector uses to export data to a Teradata system. Required: no Supported values: batch.insert, multiple.fastload, internal.fastload Default value: batch.insert	N/A
num.mappers	The maximum number of output mapper tasks. If the value is zero, then the number of mappers will be the same as the number of file blocks in HDFS. Use either this parameter or 'num.reducers', but not both. Required: no Supported values: an integer greater than or equal to zero Default value: 2	-m –num-mappers
num.reducers	The maximum number of output reducer tasks if export is done in the reduce phase. Use either this parameter or 'num.mappers', but not both.	N/A

Export Option (`teradata.db.output.*`)	Description	Overrriding Sqoop Option
	Required: no Supported values: an integer greater than or equal to zero Default value: 0	
source.table	The name of the source table in Hive or HCatalog from which the data is exported. It is required for an 'hcat' job, optional with a 'hive' job, and not valid with an 'hdfs' job. For a 'hive' job, either specify this or the 'source.paths' parameter but not both. Required: no Supported values: string	N/A
source.table.schema	The full column schema of the source table in Hive or HCatalog, not including the partition schema if the table has any, in comma-separated format. When using the data mapping feature with an 'hdfs' job operating on a text file, use this parameter to describe the file content schema. Required: no Supported values: string	N/A
source.partition.schema	The full partition schema of the source table in Hive, in comma-separated format. It is valid with a 'hive' job only. When this parameter is used, the 'source.table.schema' parameter must also be specified. Required: no Supported values: string	N/A
source.field.names	The names of fields to export from the source HDFS files, or from the source Hive and HCatalog tables, in comma-separated format. The order of the source field names must match the order of the target field names for schema mapping. This parameter must be provided when the 'target.field.names' parameter is specified. Required: no Supported values: string	N/A
target.field.names	The names of fields to export to the target table in a Teradata system, in comma-separated format. The order of the target field names must match the order of the source field names for schema mapping. This parameter must be provided when the 'source.field.names' parameter is specified. Required: no Supported values: string	–columns
target.field.count	The number of fields to export to the target table in the Teradata system. Either specify this or the 'target.field.names' parameter, but not both. Required: no Supported values: integer Default value: 0	N/A

Export Option (`teradata.db.output.*`)	Description	Overrriding Sqoop Option
fastload.socket.host	The job client host name or IP address that fastload tasks communicate with to synchronize states. This parameter is valid with the 'internal.fastload' method only. If this parameter is not specified, the Teradata connector will automatically lookup for the node that the job is launched on and the configuration values of the 'dfs.datanode.dns.interface' parameter or the 'mapred.tasktracker.dns.interface' parameter, if these are configured. Otherwise, the Teradata connector will select the IP address of the node's first network interface. Required: no Supported values: resolveable host name or IP address	N/A
fastload.socket.port	The host port that fastload tasks will communicate with to synchronize states. This parameter is valid with the 'internal.fastload' method only. If this parameter is not specified, the Teradata connector will automatically select an available port starting from 8678. Required: no Supported values: integer	N/A
batch.size	The number of rows the Teradata connector will send each time to the Teradata system. Required: no Supported values: integer Default value: 10000	N/A
separator	The separator of fields in the source to-be-exported files. This parameter is only valid with 'textfile' file format. Required: no Supported values: string Default value: \t	–input-fields-terminated-by
stage.database	The database which the Teradata connector uses to create staging tables. Required: no Supported values: the name of a database in the Teradata system Default value: the current logon database in the JDBC connection	N/A
stage.table.name	The name which the Teradata connector uses to create a staging table in the Teradata system, if staging is required. Its length cannot exceed 20 characters. It should be used when the target Teradata table has a name exceeding 24 characters. Required: no Supported values: string less than 17 characters	

Export Option (`teradata.db.output.*`)	Description	Overrriding Sqoop Option
teradata.db.hive.configuration.file	The path to the Hive configuration file in the HDFS. It is required for a 'hive' or 'hcat' job launched through remote execution or on data nodes. Required: no Supported values: string	N/A

Unsupported Export Options

This section lists Sqoop export options that are unsupported by the Teradata connector.

Control Options

The Teradata connector does not support these Sqoop export control options:

```
--batch
--clear-staging-table
--direct
--update-key
--update-mode
```

Input Parsing Options

The Teradata connector does not support these Sqoop input parsing options:

```
--input-enclosed-by
--input-escaped-by
--input-lines-terminated-by
--input-optionally-enclosed-by
```

Output Formatting Options

The Teradata connector does not support these Sqoop output formatting options:

```
--enclosed-by
--escaped-by
--fields-terminated-by
--lines-terminated-by
--mysql-delimiters
--optionally-enclosed-by
```

Data Mapping Options

The Teradata connector does not support these Sqoop data mapping options:

```
--input-null-non-string
--input-null-string
--map-column-java
--null-non-string
--null-string
```

Data Type Support

Data types supported by the Teradata connector are directly based on Teradata types.

Teradata Data Types

BIGINT	TIME (n)	INTERVAL HOUR (n) TO SECOND (m)
BYTEINT	TIMESTAMP (n)	INTERVAL MINUTE (n)
INTEGER	PERIOD (DATE)	INTERVAL MINUTE (n) TO SECOND (m)
SMALLINT	PERIOD (TIME (n))	INTERVAL SECOND (n)
TINYINT	PERIOD (TIMESTAMP (n))	*The following data types are supported with some limitations:*
DOUBLE PRECISION	INTERVAL YEAR (n)	• BYTE (n) {1}
FLOAT	INTERVAL YEAR (n) TO MONTH	• VARBYTE (n) {1}
REAL	INTERVAL MONTH (n)	• BLOB {1}{2}
DECIMAL (n,m)	INTERVAL DAY (n)	• CLOB {2}
NUMERIC (n,m)	INTERVAL DAY (n) TO HOUR	• ARRAY {3}
CHAR (n)	INTERVAL DAY (n) TO MINUTE	{1} *Converted to HEX string*
VARCHAR (n)	INTERVAL DAY (n) TO SECOND (m)	{2} *BLOB and CLOB datatypes are limited to 64 KB in length*
LONG VARCHAR	INTERVAL HOUR (n)	{3} *Can be used for InputFormat only*
DATE	INTERVAL HOUR (n) TO MINUTE	

Hive Data Types

BIGINT	*The following Hive types are supported with some limitations:*
INT	• BINARY {a}
SMALLINT	• MAP {b}
TINYINT	• ARRAY {b}
DOUBLE	• STRUCT {b}
FLOAT	• TIMESTAMP {c}
STRING	{a} *Supported with Hive 0.10.0 or later*
BOOLEAN	{b} *Converted to/from VARCHAR in JSON format on the Teradata system*
	{c} *Custom formats are not supported*

Unsupported Data Types

These Teradata types are unsupported:	*This Hive type is unsupported:*
• GRAPHIC	• UNION
• VARGRAPHIC	
• LONG VARGRAPHIC	

Hive and HCatalog Support

Importing from Hive and HCatalog requires that HADOOP_CLASSPATH and LIB_JARS be specified before the **sqoop** command is run. For example, this shows the environment variable setup:

```
HADOOP_CLASSPATH=/usr/lib/hive/lib/hive-metastore-0.10.0.21.jar:/usr/lib/
hive/lib/libthrift-0.9.0.jar:/usr/lib/hive/lib/hive-exec-0.10.0.21.jar:/usr/
lib/hive/lib/libfb303-0.9.0.jar:/usr/lib/hive/lib/jdo2-api-2.3-ec.jar:/usr/
lib/hive/lib/antlr-runtime-3.0.1.jar:/usr/lib/hive/lib/mysql-connector-java.
jar:/usr/lib/hive/lib/commons-dbcp-1.4.jar:/usr/lib/hive/lib/commons-pool-1.
5.4.jar:/usr/lib/hive/lib/hive-cli-0.10.0.21.jar:/usr/lib/hive/lib/hive-
builtins-0.10.0.21.jar:/usr/lib/hive/conf:/usr/lib/hcatalog/share/hcatalog/
hcatalog-core.jar

LIB_JARS=usr/lib/hive/lib/hive-metastore-0.10.0.21.jar,/usr/lib/hive/lib/
libthrift-0.9.0.jar,/usr/lib/hive/lib/hive-exec-0.10.0.21.jar,/usr/lib/hive/
lib/libfb303-0.9.0.jar,/usr/lib/hive/lib/jdo2-api-2.3-ec.jar,/usr/lib/hive/
lib/hive-cli-0.10.0.21.jar,/usr/lib/hive/lib/hive-builtins-0.10.0.21.jar,/usr/
lib/hcatalog/share/hcatalog/hcatalog-core.jar

export HADOOP_CLASSPATH LIB_JARS
```

With these two environment variable settings, the Hive and HCatalog jobs can be run as shown in the following examples.

Sample Invocations

Import Data from Teradata to Hadoop and Hive

```
sqoop import \
    -libjars $LIB_JARS \
    -Dteradata.db.input.job.type=hive \
    -Dteradata.db.input.target.table=hive_table \
    -Dteradata.db.input.target.table.schema="emp_no int, birth_date string, \
      first_name string, last_name string, gender string, hire_date string" \
    --connect jdbc:teradata://td-host/Database=dbname \
    --connection-manager org.apache.sqoop.teradata.TeradataConnManager \
    --username tduser \
    --password tduserpass \
    --table tablename
```

Import Data from Teradata into an HCatalog Table

```
sqoop import \
    -libjars $LIB_JARS \
    -Dteradata.db.input.job.type=hcat \
    -Dteradata.db.input.target.table=hcat_table \
    --connect jdbc:teradata://td-host/Database=dbname \
    --connection-manager org.apache.sqoop.teradata.TeradataConnManager \
    --username tduser \
    --password tduserpass \
    --table tablename
```

Known Issues and Workarounds

Stage Tables

Issue: The export option `--stage-table` does not work.

Cause: The behavior of stage tables is different between Teradata connector and Sqoop, and this causes deadlocks during job cleanup if the Sqoop `-staging-table` option is used.

Workaround: Use the Teradata connector option `teradata.db.output.stage.table.name` for specifying the stage table name.

Fastload

Issue: The export option `'fastload.soclet.host'` does not work.

Cause: The `internal.fastload` method used for Teradata exports can cause resource exhaustion (running out of database AMPs) if the number of reducers exceeds the number of available AMPs.

Workaround: Use the option `teradata.db.output.num.reducers` to restrict the resource usage.

Query String

Issue: The Sqoop import option `-query` is not used by the Teradata connector, but currently the Sqoop connector needs a query string to be specified on the command line for query imports.

Cause: The query string used by the Sqoop connector has requirements that are invalid for the Teradata connector. Specifically, the Sqoop command line requirement expects a query string with a "$CONDITIONS" token to substitute data range clauses. However, the Teradata connector generates data ranges internally based on the query and so does not use the `-query` import option of Sqoop.

Workaround: Specify the query string to use with the Teradata connector option `teradata.db.input.source.query`, and specify the same query with "$CONDITIONS" added at the end to satisfy the Sqoop requirements.

For example, the following command line shows the use of the query string:

```
sqoop import \
-Dteradata.db.input.source.query="select * from mytable" \
--connect jdbc:teradata://td-host/Database=dbname \
--connection-manager org.apache.sqoop.teradata.TeradataConnManager \
--username tduser \
--password tduserpass \
--target-dir hdfs-dir -m 1 \
--query 'select * from mytable where $CONDITIONS'
```