Hortonworks Connector for Teradata

(Oct 8, 2015)

docs.hortonworks.com

Hortonworks Connector for Teradata

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain, free and open source.

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

1. Hortonworks Connector for Teradata

1.1. Introduction

Hortonworks Connector for Teradata (powered by the 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.

1.1.1. Background

Sqoop provides facilities for bulk transfer of data between external data stores and the Hadoop environment exploiting the Ma p 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.

1.1.1.1. Terminology

Sqoop has the notion of *Connectors,* which contain the specialized logic to read and write to external systems.

- The *Hortonworks Connector for Teradata* ("Hortonworks Connector") is a Sqoop Connector implementation for Teradata.
- It is built on the *Teradata Connector for Hadoop,* a Teradata product.

1.1.2. Supported Features

The Hortonworks Connector supports the following features:

- Import/Export tools that run Hadoop MR jobs to transfer data.
- Support for Text, Sequence, ORCFiles, Avro, and RCFiles as the source for export operations and target for im port operations.

  Note: If you will run Avro jobs, download avro-mapred-1.7.4-hadoop2.jar and place it under $SQOOP_HOME/lib.
- Importable 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.

1.1.2.1. Connector Feature Checklist

Import all tables: Supported.

Incremental import: Sqoop options are not supported but can be emulated, as specified in the sample invocation Incremental Import.

BLOB and CLOB: Limited to 64 KB.

Import data to Sqoop

- **TextFormat, delimited:** Supported.
- **SequenceFile:** Supported.
- **RCFile:** Supported.
- **ORCFile:** Supported with HDP 2.3.4 or later.
- **Avro file:** 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.

1.2. Software Versions and Installation

1.2.1. Connector Version

This document discusses the Hortonworks Connector for Teradata ("Hortonworks Connector") built on version 1.4.1 of the Teradata Connector for Hadoop.

1.2.2. Supported Product Versions

This section lists the product versions supported in the current release of the Hortonworks Connector.

1.2.2.1. Teradata Database Versions

The following Teradata database versions are supported:

- Teradata Database 13.00
- Teradata Database 13.10
- Teradata Database 14.00
- Teradata Database 14.10
- Teradata Database 15.00
- Teradata Database 15.10

1.2.2.2. Hive Version

- Hive 1.2.1

1.2.2.3. Hadoop Version

- • H D P 2.3.4.0 or later

1.2.2.4. Sqoop Versions

- Sqoop 1.4.6

1.2.3. Requirements and Dependencies

1.2.3.1. System Requirements

The Hortonworks Connector requires JRE/JDK 1.7.x or 1.8.x.

1.2.3.2. Dependencies

1. Teradata GSS Client Driver 15.00 or later versions (tdgssconfig)
2. Teradata JDBC Driver 15.00 or later versions (terajdbc)
3. Teradata Connector for Hadoop 1.4.1

1.2.4. Installation

1.2.4.1. Installation Dependencies

Sqoop must be installed first.

1.2.4.2. Installing the Software

1. Download the tarball from the "Add-Ons" for Hortonworks Data Platform 2.3.4 here: http://hortonworks.com/hdp/addons.
2. 2. Extract the contents of the tar archive to $SQOOP_HOME/lib. Sqoop will then distribute the contents of the tar to the necessary nodes.

1.3. Configuration

This section provides information about connection credentials and configuration options.

1.3.1. Database Connection Credentials

Refer to Sqoop documentation for the Teradata database connection credentials.

Documentation for Sqoop version 1.4.6 is available here: http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.3.4/bk_dataintegration/content/ch_using-sqoop.html.

1.3.2. Configuration Options

The Hortonworks Connector defines many connector-specific options. A good selection of them is also available as Sqoop options (although not all Sqoop options are directly translatable to Hortonworks Connector options).

1.3.2.1. Configuration Option Precedence

Options can be specified using any of these techniques:

- a configuration file
- `-D` command line option
- Sqoop options (where applicable): apart from standard Sqoop options, a few connector-specific options are supported

Therefore the following precedence is established:

1. 1. Sqoop connector-specific extra arguments have the highest precedence. (Sqoop command line options must match, or execution will fail.)
2. If `-D` command line options are provided, they override the configuration file values.
3. The value in the configuration file is the default.

As an example, if the configuration file sets the number of input mappers to 4 and the command line option (`-D com.teradata.db.input.num.mappers`) sets it to 5, but the Sqoop option `--num-mappers` is set to 6, then the import job will use 6 mappers.

In some cases, option constraints and the relationships between options affect the configuration value used. For example, import options `job.type` and `file.format` are interrelated. These options are described in Connector Import Options.

1.3.2.2. Sqoop Options

The Sqoop option `--connection-manager` must be set as follows to use the Hortonworks Connector for Teradata (see the Sample Invocations):

```
--connection-manager org.apache.sqoop.teradata.TeradataConnManager
```

Some of the Sqoop options are unsupported in the current release of the Hortonworks Connector for Hadoop. See the Appendix for a list of unsupported Sqoop options.

1.3.2.3. Hortonworks Connector Options

The Appendix describes the Hortonworks Connector options, including Connector Import Options and Connector-specific Extra Arguments.

1.4. Data Type Support

The Hortonworks Connector data types depend on Teradata database types.

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

1.4.2. Support for 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*

1.4.3. Unsupported Data Types

These Teradata types are unsupported:	*This Hive type is unsupported:*

• GRAPHIC • VARGRAPHIC • LONG VARGRAPHIC	• UNION

1.5. Hive and HCatalog Support

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

```
export HADOOP_CLASSPATH=$(hcat -classpath)
HIVE_HOME=/usr/hdp/current/hive-client
HCAT_HOME=/usr/hdp/current/hive-webhcat

export LIB_JARS=$HCAT_HOME/share/hcatalog/hcatalog-core-<version>.jar,
$HIVE_HOME/lib/hive-metastore-<version>.jar,
$HIVE_HOME/lib/libthrift-<version>.jar,\
$HIVE_HOME/lib/hive-exec-<version>.jar,\
$HIVE_HOME/lib/libfb303-<version>.jar,\
$HIVE_HOME/lib/jdo2-api-<version>.jar,\
$HIVE_HOME/lib/slf4j-api-<version>.jar,\
$HIVE_HOME/lib/hive-cli-<version>.jar,\
```

Note: Change the HIVE_HOME and HCAT_HOME variables as needed and change the versions of the jar to what is available under the directories mentioned.

Hive and HCatalog jobs can be run as shown in the next section.

1.6. Sample Invocations

The following examples assume that the SQOOP_HOME environment variable is set to the base directory of the Sqoop installation.

1.6.1. Import Data from Teradata to Hadoop and Hive

```
$SQOOP_HOME/bin/sqoop import \
-libjars $LIB_JARS \
--connection-manager org.apache.sqoop.teradata.TeradataconnManager \
--username tduser \
--password tduserpass \
--table tablename
--hcatalog-table hcat table
```

1.6.2. Import Data from Teradata into an HCatalog Table

```
$SQOOP_HOME/bin/sqoop import \
-libjars $LIB_JARS \
--connect jdbc:teradata://td-host/Database=dbname \
--connection-manager org.apache.sqoop.teradata.TeradataConnManage r \
--username tduser \
--password tduserpass \
--table tablename
--hcatalog-table hcat table
```

1.6.3. Incremental Import

Teradata incremental import emulates the check-column and last value options. Here is an example for a table which has 'hire_date' as the date column to check against and 'name' as the column that can be used to partition the data.

```
export USER=dbc
export PASS=dbc
export HOST=<dbhost>
export DB=<dbuser>
export TABLE=<dbtable>
export JDBCURL=jdbc: teradata: //$HOST/DATABASE=$DB
export IMPORT_DIR=<hdfs-dir to import>
export VERBOSE=--verbose
export MANAGER=org.apache.sqoop .teradata. TeradataConnManager
export CONN_MANAGER="--connection-manager $MANAGER"
export CONNECT="--connect $JDBCURL"
MAPPERS="--num-mappers 4"
DATE="'1990-12-31' "
FORMAT=" 'yyyy-rnrn-dd' "
LASTDATE="cast( $DATE as date format $FORMAT)"
SQOOPQUERY="select *  from employees where hire_date < $LASTDATE AND \
$CONDITIONS"
$SQOOP_HOME/bin/sqoop import $TDQUERY $TDSPLITBY $INPUTMETHOD $VERBOSE
$CONN_MANAGER $CONNECT -query "$SQOOPQUERY" --username $USER --password $PASS
--target-dir $IMPORT_DIR --split-by name
```

1.6.4. Export Data to Teradata

```
$SQOOP_HOME/bin/sqoop export \
    --connect jdbc:teradata://172.16.68.128/Database=employees \
    --connection-manager org.apache.sqoop.teradata.TeradataConnManager \
    --username dbc \
    --password dbc \
    --table employees2 \
    --export-dir /user/hrt_qa/test-sqoop/out \
    --batch
```

1.7. Known Issues and Workarounds

1.7.1. Stage Tables

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

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

Workaround: Use the Hortonworks Connector option `teradata.db.output.stage.table.name` for specifying the stage table name.

1.7.2. 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.

1.8. Appendix: Configuration Options

This appendix describes the Hortonworks Connector configuration options and lists the Sqoop options that are currently unsupported.

- Sqoop Options
- Hortonworks Connector Options

1.8.1. Sqoop Options

To use the Hortonworks Connector, you must set the Sqoop option `--connection-manager` to `org.apache.sqoop.teradata.TeradataConnManager` as shown in the Sample Invocations.

Some of the Sqoop options are unsupported in the current release of the Hortonworks Connector for Hadoop. The tables below list the unsupported import and export options.

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.

1.8.1.1. Unsupported Sqoop Import Options

Import Category	Unsupported Options

Control Options	`--append`
	`--compression-codec`
	`--direct`
	`--direct-split-size`
	`--where`
	`--compress, -z`

Incremental Options	`--check-column`
	`--incremental`
	`--last-value`

Output Formatting Options	`--mysql-delimiters` `--optionally-enclosed-by`
Hive Support Options	`--hive-delims-replacement` `--hive-drop-import-delims`

Import Category	Unsupported Options

	`--hive-home`
	`--hive-overwrite`
	`--hive-partition-key`
	`--hive-partition-value`
	`--map-column-hive`

HBase Support Options	`--column-family`
	`--hbase-create-table`
	`--hbase-row-key`
	`--hbase-table`

Data Mapping Options	`--map-column-java`

1.8.1.2. Unsupported Sqoop Export Options

Export Category	Unsupported Options

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

Input Parsing Options	`--input-lines-terminated-by` `--input-optionally-enclosed-by`
Data Mapping Options	`--map-column-java`

1.8.2. Hortonworks Connector Options

This section describes configuration options provided by the Hortonworks Connector.

- Connector Import Options
- Connector Export Options
- Connector-specific Extra Arguments

For information about how the options can be specified, see Configuration Option Precedence.

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.

1.8.2.1. Connector 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`.

Connector 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 for 'hcat' and 'hive' settings; also none for 'hdfs' when the file format is 'textfile'. But for file formats other than 'textfile' the 'hdfs' job type is reset to 'hive', therefore the following Sqoop option overrides a job.type of 'hdfs': `--as-sequencefile`

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
	
	Supported values: orcfile, refile, sequencefile, textfile
	
	Default value: textfile
		`--as-sequencefile`
		
		`--as-textfile`

target.paths	The directory with which to place the imported data. It is required for an 'hdfs' job, optional for a 'hive' job, and not valid for an '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`

num.mappers	The number of mappers for the import job. It is also the number of splits the Hortonworks 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)	`--query`
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	`--columns`

Connector Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
	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	
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.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 Hortonworks 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.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	`--split-by`

1.8.2.2. Connector 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`.

Connector 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`
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`

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`

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`
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`

1.8.2.3. Connector-specific Extra Arguments

The Hortonworks connector for Teradata has the following connector-specific extra arguments:

Type of Argument	Argument	Description
Common Options		
	jobtype	The job type: hdfs, hive or hcat.

Type of Argument	Argument	Description
	fileformat	File format: sequencefile, textfile, avrofile, orcfile or refile. Default is textfile.
	usexview	Use X views for metadata queries. (X views take security into consideration.)
	stagedatabase	Database to use for creating stage tables.
	stagetablename	Stage table name to use; if blank, a default name is generated.
	batchsize	Fetch size or insert batch size.
	queryband	Query band for the session.
Import-specific Options	numpartitionshating	Number of partitions to be created in the staging table.
	method	One of split.by.{value | hash | partition | amp}
	accesslock	Row lock is used for fetching rows.
	avroschemafile	Avro schema file for Avro imports.
	targettableschema	Schema for the partitioning columns. Needed when Hive table is to be created.
	targetpartitionschema	Schema for the partitioning columns. Needed when Hive table is to be created.
	targetfieldnames	Field names for the target fields. Needed when Hive table is to be created.
Export Options	sourcetableschema	Schema for the source hive table.
	sourcepartitionschema	Schema for the partitioning columns.
	sourcefieldnames	Field names for the source fields to export.
	fastloadsockethost	Host for Fastload exports.
	fastloadsocketport	Port for the Fastload exports.
	fastloadsockettimeout	Timeout for the Fastload export operation.
	errortablename	Error table name for use with Fast load.
	keepstagetable	Keep stage table after export. (If not present, stage table is dropped after export.)
	forcestage	Force creation of a stage table.