Hortonworks Data Platform

Teradata Connector User Guide

(December 17, 2019)

docs.cloudera.com

Hortonworks Data Platform: Teradata Connector User Guide

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. You can contact us directly to discuss your specific needs.

Table of Contents

1. What's New in Hortonworks Connector for Teradata

The Hortonworks Connector for Teradata 1.6 supports HDP 3.0 and later and the following capabilities:

- Importing from or exporting to HDFS
- Importing from or exporting to a Hive table
- Importing a table from Teradata or exporting to a Teradata table

2. Hortonworks Connector for Teradata

2.1. Introduction

Hortonworks Connector for Teradata (powered by the Teradata Connector for Hadoop) 1.6.3 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. You can use the connector with HDP 3.x and later. It is not backwards compatible with previous HDP versions, MapR versions and/or CDH versions. Please see below for additional details. Download the Teradata v1.6.3 connector from HDP Add-Ons.

2.1.1. 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.

2.1.1.1. 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.

2.1.2. 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.

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

2.1.2.1. 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.
- **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.

2.2. Software Versions and Installation

2.2.1. Connector Version

This document discusses the Hortonworks Connector for Teradata ("Hortonworks Connector") built on version 1.6.3 of the Teradata Connector for Hadoop.

2.2.2. Supported Product Versions

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

2.2.2.1. HDP Version

- HDP 3.1.5 or later

2.2.2.2. Teradata Database Versions

The following Teradata database versions are supported:

- Teradata Database 16.00
- Teradata Database 15.10
- Teradata Database 15.00
- Teradata Database 14.10
- Teradata Database 14.00

2.2.2.3. Hive Version

- Hive 3.x and later

2.2.2.4. Sqoop Versions

- Sqoop 1.4.7

2.2.3. Requirements and Dependencies

2.2.3.1. System Requirements

The Hortonworks Connector requires:

- OracleJRE/OracleJDK 1.8 or later versions
- OpenJRE/OpenJDK 1.8 or later versions

2.2.3.2. Dependencies

1. Teradata GSS Driver 16.20 (tdgssconfig)
2. Teradata JDBC Driver 16.20 (terajdbc)
3. Teradata Connector for Hadoop 1.6.3

2.2.4. Installation

2.2.4.1. Installation Dependencies

Sqoop must be installed first.

2.2.4.2. Installing the Software

1. Download the tarball from the "Add-Ons" for the latest version of Hortonworks Data Platform (HDP) here: https://hortonworks.com/downloads/.

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.

2.3. Configuration

This section provides information about connection credentials and configuration options.

2.3.1. Database Connection Credentials

Refer to Sqoop documentation for the Teradata database connection credentials.

2.3.2. 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).

2.3.2.1. 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. 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.

2.3.2.2. 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.

2.3.2.3. Hortonworks Connector Options

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

2.4. Data Type Support

The Hortonworks Connector data types depend on Teradata database types listed in Section 2.7, "1.6.3 Connector Release Content" [13].

2.4.1. Unsupported Data Types

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

2.5. Sample Invocations

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

Important

When importing data from Teradata Connector for Hadoop version 1.6.3 or later into Hive in a non-TDE cluster, the user must be in the same group as the group with access to the Hive table warehouse directory.

2.5.1. 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
```

2.5.2. Specify a Connector-Extra Argument

For example, to use numpartitionsinstaging extra arguments you need to pass in the information at the very end of the Sqoop command:

```
$SQOOP_HOME/bin/sqoop import \
    -libjars $LIB_JARS \
    --connection-manager org.apache.sqoop.teradata.TeradataConnManager \
```

```
    --query "SELECT * FROM TEST" \
    --target-dir /tmp/td_sqoop_test \
    -connect "jdbc:teradata://localhost/ENCRYPTDATA=OFF,DATABASE=local" \
    -username xxx \
    -password xxx \
    --split-by IID \
    --num-mappers \
    --verbose
    – --numpartitionsinstaging
```

2.5.3. 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-mm-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
```

2.5.4. 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
```

2.6. Appendix: Configuration Options

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

- Hortonworks Connector Options [9]

2.6.1. 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.

2.6.1.1. 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`
	`--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`

2.6.1.2. 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`

2.6.2. Hortonworks Connector Options

This section describes configuration options provided by the Hortonworks Connector.

- Connector Import Options [9]
- Connector Export Options [11]
- Connector-specific Extra Arguments [12]

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.

2.6.2.1. 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.	`--as-sequencefile` `--as-textfile`

Connector Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
	Required: no Supported values: orcfile, refile, sequencefile, textfile Default value: 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 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`
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	`--hive-table`

Connector Import Option (`teradata.db.input.*`)	Description	Overrriding Sqoop Option
	Supported values: string	
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`

2.6.2.2. 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,	`--export-dir`

Connector Export Option (`teradata.db.output.*`)	Description	Overrriding Sqoop Option
	either specify this or the 'source.table' parameter but not both. Required: no Supported values: string	

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`

2.6.2.3. 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.
	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.

Type of Argument	Argument	Description
	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.

2.7. 1.6.3 Connector Release Content

- Data Transfer
  - Teradata
    - i. Import table from Teradata method: split.by.hash, split.by.value, split.by.partition split.by.amp, internal.fastexport
    - ii. Import query from Teradata method: split.by.partition
    - iii. Export to empty/non-empty Teradata table method: batch.insert, internal.fastload
  - HDFS
    - i. Import data from HDFS
    - ii. Export data to HDFS
  - Hive

- Import data from Hive table
- Export data to Hive table

  1) Create non-existing Hive table.

  2) Add partitions to existing Hive table.

- HCat
  - i. Import data from HCat not supported
  - ii. Export data to HCat not supported

- **Schema Mapping**

  All fields (columns)

  Selected fields (columns)

  Null / not null values

  Data type conversions

  Data format conversions

- **Remote Execution**

  Templeton

  Oozie

- **Data Types**
  - **Teradata Data Types**

    BIGINT

    BYTEINT

    INTEGER

    SMALLINT

    DOUBLE PRECISION

    FLOAT

    REAL

    DECIMAL (n,m)

    NUMERIC (n,m)

    NUMBER (n,m)

CHAR (n)

VARCHAR (n)

LONG VARCHAR

DATE

TIME (n)

TIME (n) WITH TIME ZONE

TIMESTAMP (n)

TIMESTAMP (n) WITH TIME ZONE

PERIOD (DATE)

PERIOD (TIME (n))

PERIOD (TIMESTAMP (n))

INTERVAL YEAR (n)

INTERVAL YEAR (n) to MONTH

INTERVAL MONTH (n)

INTERVAL DAY (n)

INTERVAL DAY (n) to HOUR

INTERVAL DAY (n) to MINUTE

INTERVAL DAY (n) to SECOND (m)

INTERVAL HOUR (n)

INTERVAL HOUR (n) to MINUTE

INTERVAL HOUR (n) to SECOND (m)

INTERVAL MINUTE (n)

INTERVAL MINUTE (n) to SECOND (m)

INTERVAL SECOND (n)

BYTE (n) (See Limitations)

VARBYTE (n) (See Limitations)

BLOB (See Limitations)

CLOB (See Limitations)

ARRAY (See Limitations)

XML

JSON

- **Hive Data Types**

  BIGINT

  SMALLINT

  TINYINT

  INT

  DECIMAL

  FLOAT

  DOUBLE

  CHAR

  VARCHAR

  STRING

  BOOLEAN

  MAP (See Limitations)

  ARRAY (See Limitations)

  STRUCT (See Limitations)

  BINARY

  DATE

  TIMESTAMP

- **Avro Data Types**

  LONG

  INT

  FLOAT

  DOUBLE

  STRING

  BOOLEAN

BYTES

NULL

RECORDS (See Limitations)

ENUMS (See Limitations)

MAPS (See Limitations)

ARRAYS (See Limitations)

UNIONS (See Limitations)

FIXED (See Limitations)

- Parquet Data Types

  INT

  TINYINT

  SMALLINT

  BIGINT

  FLOAT

  DOUBLE (See Limitations)

  BINARY (See Limitations)

  TIMESTAMP (See limitations)

  STRING

  BOOLEAN

  CHAR

  VARCHAR

- File Storage Format
  - HDFS: TextFile, AvroFile
  - Hive: SequenceFile

    TextFile

    RCFile

    ORCFile

    AvroFile

Parquet (See Limitations)

- Compression
  - Import -> Hive/HDFS

    i. Intermediate (always Snappy)

    ii. Output: Snappy, LZO, GZip, BZip2
  - Export -> Teradata: Only Snappy intermediate compression
- Character Sets: ASCII, UTF8, UTF16

2.8. Included features in the 1.6.3 release

TDCH-1790: Upgrade TDJDBC jar from 16.20.00.08 to 16.20.00.12

Included features in the 1.6.2 release

TDCH-1768: Queryband Parameter support in Data Mover workflows

Included features in the 1.6.1 release

TDCH-1480: Hive Decimal data type support for Parquet

TDCH-1753: Add more diagnostics messages to internal.fastexport and internal.fastload methods

TDCH-1747: Provide emptytablecheck option to disable empty source table check

Included features in the 1.6.0 release

TDCH-1727: Implement Orc File Format schema changes.

TDCH-1732: Change configureOozie script for HDP 3.0

Included features in the 1.5.5 release

TDCH-1697: Upgraded TDJDBC to 16.20 from 16.10

TDCH-1471: Add ability to define staging tables/views in different database

TDCH-1327: Support Avro with Hive

Included features in the 1.5.4 release

TDCH-1669: Upgraded TDJDBC to 16.10 from 16.00

TDCH-1671: Added tdch installation scripts from the teradata-hadoop-builder to TDCH rpm

Included features in the 1.5.3 release

TDCH-1099: Update TD wallet to version 16.10

TDCH-1527: Implement exception logging for Export

TDCH-1552: Add new Oozie workflows to configureOozie script for TD Studio

Included features in the 1.5.2 release

TDCH-1153: Support JDBC Unicode Passthrough for TD DB 16.00+

TDCH-1285: Add Compression capability into TDCH

TDCH-1352: Add command-line argument (upt) for Unicode Passthrough parameter

TDCH-1402: Add new JDBC driver JAR to support Teradata Database 16.00

TDCH-1435: Address ResourceManager HA for TDCH jobs using Oozie

TDCH-1489: Modify calls to HiveMetaStoreClient class to provide support in future versions of Hive

Included features in the 1.5.0 release

TDCH-1356: Extend configureOozie to support MapR 4.x

TDCH-1292: configureOozie script has been modified to copy needed jars from HDFS instead of from local system for HDP 2.x

TDCH-1269: Add the ability to truncate the Teradata table on export to Teradata

TDCH-922: TDCH support for distributed JDBC FastExport

TDCH-717: Support for Hive CHAR type

TDCH-715: Support for Hive BINARY type

TDCH-701: TDCH support for Hive Parquet file format

TDCH-528: Support for Hive VARCHAR type

TDCH-443: Add JSON support between Teradata and Hadoop

TDCH-442: Add XML support between Teradata and Hadoop

TDCH-332: Support for Hive DATA type

Included features in the 1.4.3 release

TDCH-1199: Improve performance of retrieving table partitions for split.by.partition

TDCH-1161: Add support for conversion from TD TIME to Hive TIMESTAMP on import

TDCH-1133: Add TDCH argument to use INSERT OVERWRITE instead of INSERT INTO to overwrite the data that is already in the hive table during an import

TDCH-1044: Make datablocksize tunable for TD staging tables

Included features in the 1.4.2 release

TDCH-1059: Have configureOozie utilize latest certified configuration for non-cert'd platforms (IE use CDH 5.4 config for CDH 5.5)

TDCH-870: Extend mapper throttle functionality to support retry/timeout

TDCH-360: Update TDCH Tutorial to reflect TDCH 1.3.x architecture

Included features in the 1.4.1 release

TDCH-947: Extend configureOozie script to assign default ports for Resource Manager based on distribution.

TDCH-905: Extend user defined converters to remove package requirement

TDCH-848: Batch.insert to support preemption

TDCH-615: Provide users with a mechanism to signal that TDCH should error out when CHAR/VARCHAR truncation occurs

Included features in the 1.4.0 release

TDCH-861: Workflows generated by configureOozie script support HA and Kerberos enabled clusters

TDCH-836: TDCH jobs only use slots available for the given queue when new '-throttlemappers true' argument is supplied

TDCH-729: Support Hive RCFile tables which utilize the non-default RCFile serde

TDCH-331: Support for the Hive Decimal datatype

TDCH-298: Merge all distribution-specific TDCH jars into two hadoop-specific TDCH jars (hadoop1.x / hadoop2.x)

Included features in the 1.3.4 release

a) TDCH jobs can use credential information from Teradata Wallet

b) Support for Kerberos-enabled clusters

c) Users can define database where internal.fastload error tables reside via -errortabledatabase command line parameter or tdch.output.teradata.error.table.database

d) Split.by.amp can be run against views (will utilize spool)

Included features in the 1.3.1 release

a) RPM based distribution with support for multiple TDCH installations in both the Linux filesystem and in HDFS

Included features in the 1.3 release

a) Change the name/package of the tool class, and the old name may be deprecated in a future release.

b) Support new plugin architecture, see section 5.0 for more information.

c) Add -errorlimit parameter to support error data check for internal.fastload job. If the number of error rows exceeds the specified value, the job will fail.

d) Support data format conversion of String type when DATE/TIME/TIMESTAMP data type conversions are involved.

e) Enhance logic to classify internal.fastload job as JDBC job classified by TASM rule.

Included features in the 1.2 release

a) Support various URI schema path specification for Avro schema file and Hive configuration file.

b) Support Teradata 14.0 Number data type.

c) Support Teradata 14.10 Extended Object Names. With Teradata 14.10, object names can have up to and including 128 characters. Nearly the complete repertoire of Unicode characters are allowed in an object name. (See Limitations)

d) Display the current TDCH version number in console output.

Included features in the 1.1 release

a) Add ORC file format support.

b) Add Avro file format support.

Included features in the 1.0.10 release

a) Add -numpartitionsinstaging parameter for split.by.partition method to specify different partition number from mapper number.

b) Report exception for unrecognized parameters.

Included features in the 1.0.9b release

a) Add -usexviews parameter for users that do not have system view and table accessing privileges.

Included features in the 1.0.9 release

a) Provide the progress report of job for internal.fastload method.

b) Record the output of job client into an HDFS file. (For Studio Integration)

Included features in the 1.0.8 release

a) Support importing data into existing non empty partitions of hive partitioned table.

b) Add -queryband parameter to support session level query band.

c) Add the following data types support for partition column: TINYINT,

SMALLINT, INT, BIGINT, FLOAT, DOUBLE, TIMESTAMP, BOOLEAN.

d) Support more characters in string value of a partition column of hive table,(e.g., '%', ':', '/', '#'). (see section 8.1 for not-supported characters)

e) Allow user to specify error table name prefix in internal.fastload method. User can provide a name as error table name's prefix with -errortablename parameter.

f) Add -keepstagetable parameter. If the parameter is set to true, the staging table will be kept when export job fails during inserting data from staging table to target table.

Included features in the 1.0.7 release

a) Add -accesslock parameter for importing data from Teradata to improve concurrency. If the parameter is set to true, the import job will not be blocked by other concurrent accesses against the same table.

b) Add the support for importing data into non existing partitions of an existing hive partitioned table.

c) Allow a Hive configuration file path to be specified by the -hiveconf parameter, so the connector can access it in either HDFS or a local file System. This feature would enable users to run hive import/export jobs on any node of a Hadoop cluster (see section 8.3)

d) After Teradata 14.10 release, split.by.amp import method is supported. (see section 7.1(d))

Included features in the 1.0.6 release

a) Add the support for user specified text format parameters including: escapedby, enclosedby, nullstring and nullnonstring(see section 9.4).

b) Add the support for using a non-printing character as the field|line separator(see section 9.5)

Included features in the 1.0.1 - 1.0.5 releases

a) Add split.by.hash import method

b) Add split.by.value import method

Included features in the 1.0.0 release

a) Support remote execution by Templeton and Oozie

b) Support quoting reserved words and non-standard ASCII characters in Teradata database/table/column names

c) Support Hive Map, Array and Struct data type

d) Import to existing Hive table

e) Create new Hive table at end of import (if table does not exist)

f) Import to Hive partitioned tables

g) Export from Hive partitioned tables

h) Retrieve automatically Hive table's metadata

Included features in the 0.80 release

a) Schema mapping and type conversion with import and export

b) Import to Hive RCFile, SequenceFile, TextFile storage format

c) Import to HCatalog table with RCFile, SequenceFile, TextFile storage format

d) Import to Hive partitioned files

e) Export from Hive RCFile, SequenceFile, TextFile storage format

f) Export from HCatalog table with RCFile, SequenceFile,TextFile storage format

g) Export from Hive partitioned files

Included features in the 0.40 - 0.50 releases

a) Use TeradataCombineFileInputFormat to limit number of map tasks

Included features in the 0.30 release

a) JDBC Internal Fastload implementation

Included features in the 0.20 - 22 releases

a) Insert HDFS delimited text file into Teradata database table via JDBC Fastload

b) Move table data or select query results from Teradata database into HDFS

c) Insert HDFS delimited text file into Teradata database table via JDBC Batch Insert

2.9. Problems Fixed in the Release

Included fixes in the 1.6.2 release

TDCH-1780: Customer Specific Issue

TDCH-1785: Customer Specific Issue

T/DCH-1776: A Socket exception on data sessions do not terminate the load - as it should

Included fixes in the 1.6.1 release

TDCH-1762: Conflicting usage of temporary MapR directory by TDCH

TDCH-1758: Removal of assert statements in TDCH source code

TDCH-1635: TDCH installation(tdch_install.sh) script fails on kerberos enabled cluster

TDCH-1633: Implement alternate behavior for input source table empty

Included fixes in the 1.6.0 release

TDCH-1730: Fix Error: java.lang.NumberFormatException: For input string: "true"

TDCH-1729: Fix : java.lang.ClassNotFoundException: org.apache.hadoop.hive.common.type.Timestamp

TDCH-1723: ClassCastException: org.apache.hadoop.hive.common.type.Timestamp cannot be cast to java.sql.Timestamp

TDCH-1721: Avro Tests failed with Managed Avro table has externally defined schema.

TDCH-1720: Export from Hive Managed table to TD table failing on HDP 3.0

TDCH-1711: tool.ConnectorExportTool: com.teradata.connector.common.exception.ConnectorException

TDCH-1708: Error: java.lang.IllegalArgumentException: Compression codec com.hadoop.compression.lzo.LzoCodec was not found.

TDCH-1707: Error: java.lang.ClassCastException: java.sql.Date cannot be cast to org.apache.hadoop.hive.common.type.Date

TDCH-1706: java.lang.ClassCastException: java.sql.Timestamp cannot be cast to org.apache.hadoop.hive.common.type.Timestamp

TDCH-1701: [HDP] TDCH install failing on HDP 3.0 Bug

TDCH-1705: Fix internal.fastload method issues on HDP 3.0

Included fixes in the 1.5.5 release

TDCH-1691: Precision is lost while importing the time stamp data from Teradata to Hadoop

TDCH-1689: TDCH job fails to export data from Hadoop when ORC and internal.fastload used

TDCH-1688: TDCH job fails if there are zero number of rows to be copied from Teradata to Hive

TDCH-1685: Compression is not applied for the target hive table for Parquet

TDCH-1683: Error during import Parquet into already existing Hive table with TDCH

TDCH-1677: TDCH failed to export Hive ORC format table with date format

TDCH-1676: TDCH jobs configured to load a Teradata table with a TIMESTAMP(6) and TIME(0)

column fail to define their import statement correctly

TDCH-1652: Remove dependency of "serialization.format"

TDCH-1649: TDCH should Filter out unsupported Unicode characters with TD 15.10

TDCH-1634: configureOozie.sh script fails with javax.security.sasl.SaslException: GSS initiate failed

TDCH-1630: Multiple input paths separated by commas can fail

Included fixes in the 1.5.4 release

TDCH-1452: TDCH job fails if we have Hive table with 'field.delim'='\\'

TDCH-1556: Make sure that the column order is returned exactly from td while create table in hive.

TDCH-1580: Hive BOOLEAN TO TERADATA BYTEINT gives NULL values always.

TDCH-1651: With TD 15.10 internal.fastexport failing when two conditions are in sql query

TDCH-1655: ConfigureOozie.sh updated to copy jars from atlas directory for latest hdp versions.

TDCH-1678: JDBC upgraded to 16.10 then TDCH internal.fastload with queryband test failed

TDCH-1679: configureOozie.sh failed on cdh when hive path not found

TDCH-1680: Update ConfigureOozie.sh to work for Mapr 6.0

TDCH-1656: [HDP] Integrate new rmHA command-line argument into any installation scripts that use TDCH and configureOozie.sh

TDCH-1657: [CDH] Integrate new rmHA command-line argument into any installation scripts that use TDCH and configureOozie.sh

Included fixes in the 1.5.3 release

TDCH-1442: split.by.value with VARCHAR column has problems with string conversions from BigDecimal

TDCH-1487: AVRO from HDFS into Teradata - silently succeeded with truncated data

TDCH-1533: TDCH silently succeeds with wrong values while using HDFS export for non-bounded NUMERIC data type in target (Teradata DB)

Included fixes in the 1.5.2 release

TDCH-685: Internal.fastload export jobs with target tables that include VARCHAR cols with length >= 64k result in SQLException

TDCH-1398: Timestamp giving trimmed output while using Import tool with idatastream method

TDCH-1401: VARBYTE and BYTES doesn't export or import VARBYTES when using Avro

TDCH-1433: Modify the failing early when no data exists in table to successfully exit

TDCH-1434: FLOAT values in "-sourcerecordschema" causing exception

TDCH-1436: Decimal data type is being converted to scientific notation with importing into HDFS

TDCH-1451: Parquet import not working with split.by.amp method for NULL VARCHAR values

TDCH-1454: Modify internal.fastload method to handle varying fractional seconds in target table

TDCH-1465: ORC import giving NULL value instead of expected 0 length with split.by.amp method for 0 length VARCHAR values

Included fixes in the 1.5.1 release

TDCH-1410: Remove 0 length files before transferring from HDFS to Hive during import into Hive

TDCH-1399: Parquet fixes for HDP2.5

Included fixes in the 1.5.0 release

TDCH-1272: TDCH Hive table creating disregards empty nullstring values

TDCH-1261: split.by.amp does not transfer data when numMappers == 1

TDCH-1259: TDCH gives index out of bounds for string to timestamp export if timeformat uses more than 3 digits in nanoseconds

TDCH-1369: split.by.partition exception when number of mappers is 1 and TD table is not partitioned

Included fixes in the 1.4.4 release

TDCH-1260: Update configureOozie to support HDP 2.4

TDCH-1245: Timestamp to timestamp support for nanoseconds

Included fixes in the 1.4.3 release

TDCH-1176: Failure with split.by.partition when using ANSI transaction mode

TDCH-860: TDCH doesn't support separators with backslash characters

TDCH-816: Split.by.hash/value should utilize first column in table whentable is NOPI and nummappers > 1

TDCH-789: Import method quietly overridden when using sourcequery option

TDCH-742: Null pointer exception thrown when column names that includespaces are double quoted in targettableschema value

TDCH-577: Method used for map tasks to connect to client forinternal.fastload can pick wrong address

Included fixes in the 1.4.2 release

TDCH-1096: Avro union to decimal conversions result in data corruption

TDCH-689: String to Timestamp(6) conversions lose nanosecond precision

TDCH-550: TDCH utilizes deprecated sun.* classes

TDCH-296: Split.by.partition utilizes staging table even when source table is partitioned

Included fixes in the 1.4.1 release

TDCH-934: Task fails with connection reset when launching jobs with 300 mappers

TDCH-923: Oozie based TDCH jobs fail with class not found exception on HDP 2.3

TDCH-908: Epoch to DATE conversion doesn't take daylight savings into account

TDCH-903: ORC imports fail on HDP 2.3 (resolved by Hortonworks BUG-42834)

TDCH-775: User-defined converters with zero-arg constructors cause 'cant find matching constructor' exceptions

Included fixes in the 1.4.0 release

TDCH-872: Empty columns in hdfs files are now treated as nulls

instead of empty strings

TDCH-827: Failed insert/select during internal.fastload jobs now

return non-zero error code

TDCH-759: ConfigureOozie script support for MapR

TDCH-754: Jobclientoutput argument support for maprfs

TDCH-692: Unrecognized command line arguments now displayed in error message

TDCH-686: String to Decimal conversions no longer result in data corruption

TDCH-680: Split column no longer needed for jobs using split.by.hash or split.by.value and a single mapper

Included fixes in the 1.3.4 release

TDCH-726: Reading ORC tables with Timestamp columns no longer ends in String cannot be cast to Timestamp exception

TDCH-720: TDCH returns non-zero error code after throwing unsupported datatype exception

TDCH-700: TDCH no longer fails to compile with JDK 8

TDCH-697: Multiple calls to TeradataInternalFastloadRecordWriter.close() no longer cause empty batch exceptions

TDCH-616: Speculative execution is properly disabled in Hadoop 2.x

TDCH-598: Null values in Time and Timestamp columns no longer cause null pointer exceptions

TDCH-510: Avro jars no longer required when reading/writing textfiles in HDFS

TDCH-256: .template files now reference TDCH 1.3.x config names

Included fixes in the 1.3.3 release

TDCH-519: TDCH jobs using the internal.fastload method that attempt to export to a table already having so many rows that the row count cannot be stored in a 32-bit two's complement integer no longer result in a numeric overflow error.

TDCH-515: TDCH jobs will not experience a slow logon to the Teradata Database issue that was causing jobs to stall more than 30 seconds before beginning to send data.

TDCH-427: The diagnostic message printed when the Teradata output postprocessor routine starts is correctly labeled as coming from the "output postprocessor".

TDCH-420: TDCH will attempt to run internal.fastload jobs where the user specified number of mappers exceeds the value returned by ClusterStatus.getMaxMapTasks().

TDCH-419: TDCH jobs no longer erroneously fail claiming that the names of error tables or staging tables are too long.

TDCH-342: TDCH jobs that export from a hive table partitioned by a date column no longer result in a NullPointerException.

TDCH-335: TDCH internal.fastload jobs that attempt to load LOB values produce an appropriate error message.

TDCH-314: An error message regarding the reason for a job failure will be output even in the case that an error also occurs during job cleanup.

TDCH-289: TDCH internal.fastload jobs no longer fail with IllegalArgumentException on MapR 3.1, HA enabled clusters.

TDCH-288: Teradata Database logon information is not visible via the job_conf.xml file.

TDCH-273: Reading from hdfs files with blank columns no longer results in IndexOutOfBoundsExceptions.

Included fixes in the 1.3.2 release

TDCH-353: Conversions from Timestamp with Timezone to long (and vice versa) return incorrect values

TDCH-352: Exports from hive tables with binary columns backed by rcfiles fail with class cast exception

TDCH-309: ConfigurationMappingUtils doesn't overwrite values

TDCH-307: FileNotFoundException thrown when hive table exists

TDCH-306: Avro schema file required

TDCH-305: Nulls in avro files get converted to empty strings

TDCH-303: Issue with escapecharacter / nullstring

TDCH-302: Issue with nullnonstring

TDCH-287: Export hive tables partitioned by smallint cols fails with ClassCastException

TDCH-286: Data corruption when exporting float cols from hive rctable in HDP 2.0

TDCH-255: Import to partitioned hcat table causes output directory already exists error

Included fixes in the 1.3.1 release

TDCH-295: Split.by.partition fails when nummappers > num physical parts

TDCH-281: NoSuchObjectException thrown when hive table does not exist

TDCH-277: Data corruption when exporting hive table backed by rcfile

TDCH-266: Fix TIME/TIMESTAMP w/ TZ NULL support; don't try to parse empty TIME/TIMESTAMP w/ TZ strings

Included fixes in the 1.3 release

a) Remove hidden 'teradata.db.custom_parse' parameter to make the logic of csv handling simple.

Included fixes in the 1.2.1 release

a) Fix error with encrypted data by updating JDBC version from 14.10.00.17 to 14.10.00.26.

b) Add "sel" keyword support for source query.

c) Fix incorrect return value of internal.fastload job when importing hive table is empty.

d) Fix the bug that "-nummappers" parameter doesn't take effect when exporting Avro file format.

Included fixes in the 1.2 release

a) Provide meaningful error messages for NPE (Null Pointer Exception), when the -targetpaths parameter is missing in a HDFS import job.

b) Fix dead lock issues caused by open JDBC control session when internal.fastload job fails.

c) Send keep-alive message between job client and finished mappers to avoid socket connection being killed by firewall.

Included fixes in the 1.1.1 release

a) Fix RCFile SerDe incompatibility between TDCH1.1 and Hive0.12.

b) Fix support of BLOB data type in HDFS text file format for import jobs.

Included fixes in the 1.1 release

a) Modified the logic which checks if a Teradata target table is empty for export methods.

b) Modified the staging/error table naming logic to prevent duplicate table names generated when multiple concurrent jobs.

c) Modified the logic which gets current user name for a Hive job, to prevent invalid path generated in a secure Hadoop environment.

Included fixes in the 1.0.10 release

a) Incorrect nullable property setting of a column selected from a source query.

b) Ambiguous table name reference for Teradata metadata tables or views.

c) JDBC 14.10 incompatible problem.

Included fixes in the 1.0.9b release

a) The special characters of Regex cannot be included in separator parameter.

Included fixes in the 1.0.9a release

a) The characters presented by Unicode format cannot be included in separator parameter, such as '\u001a'.

Included fixes in the 1.0.9 release

a) Ambiguous partition column reference for split.by.partition method.

Included fixes in the 1.0.8 release

a) Unified case-sensitivity checking for all parameters. Please see section 5.3.

b) A partition columns mismatch issue for query based import job. This problem occurs when following conditions are satisfied:

1) A source query is provided for importing data from Teradata to Hadoop.

2) Specified source fields and target fields are subset of the selected columns of the query.

3) Source and target field names for schema mapping are provided.

c) Inappropriate warning reported for export job failure caused by inserting into target table from staging table.

d) The error tables generated by internal fastload are not correctly dropped at the end of job.

e) Incorrect process of redundant white space(more than one adjacent white space, white space between column definition and comma, etc.)

in user provided hive table and partition schema option.(Temporarily do not support redundant white space in complex types definition)

f) Fix unexpected syntax error when adding partitions to existing hive table with hive table's database specified.

Included fixes in the 1.0.7 release

a) Inappropriate exceptions reported from a query-based import job.

Only the split.by.partition method supports a query as an import

source. A proper exception will be thrown if a non split.by.partition

import job is issued with the "sourcequery" parameter.

b) One gets an error when the user account used to start Templeton is different from the user account used by Templeton to run a connector job.

c) A time-out issue for large data import jobs. In the case of a large-size data import, the Teradata database may need a long time to produce the results in a spool table before the subsequent data transfer. If this exceeds the time-out limitation of a mapper before the data transfer starts, the mapper would be killed.

With this fix, the mapper would be kept alive instead.

d) A timeout issue for export jobs using internal.fastload. The internal.fastload export method requires synchronization of all mappers at the end of their execution.

If one mapper finishes its data transfer earlier than some others, it has to wait for other mappers to complete their work. If the wait exceeds the time-out of an idle task, the mapper would be killed by its task tracker. With this fix, that mapper would be kept alive instead.

e) Fix the limitation that the user should have authorization to create local directory while executing Hive job on one node without Hive configuration (hive-site.xml) file. Before the bug fixing, the TDCH need to copy the file from HDFS to local file system.

f) Case-sensitivity problems with the following parameters: "-jobtype", "-fileformat", and "-method". With this fix, values of these parameters do not have to case-sensitive any more.

g) Incorrect delimiters used by an export job for Hive tables in RCFileFormat.

Included fixes in the 1.0.6 release

a) Hive table owner attribute for an import job was not set properly

b) JDBC URL CHARSET set to lower case utf8/utf16 gets the exception character set not supported by JDBC driver

c) Issues with column name case sensitivity cause the exception field name is not found in schema

d) Incorrect calculation of target table column count for a Hive export job with source table schema and source partition schema gets the exception source and target field count are different

e) getListTableSQL() returns not only tables but also views

Included fixes in the 1.0.5 release

a) Cannot load data into a non-default Hive database

Included fixes in the 1.0.1 - 1.0.4 releases

a) New split.by methods do not support where condition

b) BLOB import gets an "String cannot be cast to [B" exception

c) Make TeradataConnection's getCurrentDatabase() public

d) Columns with Period and Interval data type gets an data type not convertible exception

e) Exit code exceeds Linux 0-255 value limit

f) partition.stage method is renamed to split.by.partition to be consistent with new import method naming

g) No data is imported into Hive table with more than one partition columns

h) split.by.partition imports only subset of rows

Included fixes in the 1.0.0 release

a) Error attempting to run with HDFS job type and RCFile format

b) Cannot auto get values for fastload socket host and socket port

c) num.partitions, combined.file.num.splits, num.mappers are confusing and not intuitive

d) Fastload job id output is not consistent

e) Interval, BLOB, CLOB and Binary data type gets an inconvertible data type error

Included fixes in the 0.50 release

a) count(*) on table is returning numeric overflow when row count exceeds INTEGER_MAX

b) Internal Fastload mapper times out when one mapper finishes much earlier than others

c) Configuration parameters are in multiple classes

d) Use TeradataObjectArrayWritable to support different type of file formats

e) Inconsistent naming convention for classes

f) CombineInputFormat's overflown block splits are non-local

Included fixes in the 0.40 - 0.45 releases

a) Staging table name contains null when query is provided to TeradataInputFormat

b) Does not import when field count is 0

c) Multiple Fastload does not support fully qualified table name

d) Batch size is not set properly

e) Improved logging and error handling

f) Split sql is using partition in where clause, should use PI column

g) InputFormat key and value is reversed

Included fixes in the 0.30 - 0.32 releases

a) Null values in HDFS file cause PreparedStatement insert exception

b) StageTablePrefix is changed to StageTableSuffix

c) Writable moved to samples program

Included fixes in the 0.20 - 0.22 releases

a) Use generic Writable interface in TeradataInputFormat and TeradataOutputFormat

b) TeradataOutputFormat supports usage in both mappers and reducers

c) Stage table names exceed 30-character limit of JDBC Fastload

d) Concurrent database insert sessions blocked each other until explicit commit

e) Proper clean up of inserts by failed mapper/reducer task attempts

2.10. Performance Enhancements in the Release

Include performance enhancements in the 1.5.0 release

TDCH-1020: Modify Teradata/Hive plug-ins to fail early in certain circumstances

Included performance enhancements in the 1.0.0 - 1.0.4 releases

- With split.by.hash and split.by.value, staging is no longer required with import methods
- Split.by.partition no longer require staging when import source table is already a PPI table.

Included performance enhancements in the 0.50 release

- Determine how to parse objects from database metadata

Included performance enhancements in the 0.40 - 0.45 releases

- InputFormat Split SQL query on PI instead of on partition
- Add TeradataAsciiTextParser for faster parsing
- Initialize Writable only once for TeradataTextWritable4FL

Included performance enhancements in the 0.30 - 0.32 releases

- Skip staging when output method is batch.insert and target table is NOPI table
- Skip staging when output method is internal.fastload and target table is fastloadable