TeradataBinarySerde

Availability

> **Earliest version CSVSerde is available**
>
> The TeradataBinarySerDe is available in Hive 2.4 or greater.

Overview

Teradata can use TPT(Teradata Parallel Transporter) or BTEQ(Basic Teradata Query) to export and import data files compressed by gzip in very high speed. However such binary files are encoded in Teradata's proprietary format and can't be directly consumed by Hive without a customized SerDe.

The TeradataBinarySerde enables users to read or write Teradata binary data as Hive Tables:

- Directly consume the Teradata Binary data file which is exported by TPT/BTEQ and then registered in Hive
- Generate the Teradata Binary data file in Hive and directly load it via TPT/BTEQ into Teradata

How to export

The TeradataBinarySerde supports data files in 'Formatted' or 'Formatted4' mode with the restrictions

- INDICDATA format is used (please don't use DATA)
- Maximum decimal digits = 38 (please don't use 18)
- Date format = integer (please don't use ANSI)

Using TPT FastExport

Here is a bash script example for how to call TPT FastExport:

TPT FastExport script example

```
query_table=foo.teradata_binary_table
data_date=20180108
select_statement="SELECT * FROM $query_table WHERE transaction_date BETWEEN DATE '2018-01-01' AND DATE '2018-01-08' AND is_deleted=0"
select_statement=${select_statement//\'/\'\'}  # Do not put double quote here
output_path=/data/foo/bar/${data_date}
num_chunks=4

tbuild -C -f $td_export_template_file -v ${tpt_job_var_file} \
  -u "ExportSelectStmt='${select_statement}',FormatType=Formatted,DataFileCount=${num_chunks},
  FileWriterDirectoryPath='${output_path},FileWriterFileName='${query_table}.${data_date}.teradata',
  SourceTableName='${query_table}'"
```

The **td_export_template_file** looks like below with proper **Format, MaxDecimalDigits,** and **DateForm** in place:

td_export_template_file

```
USING CHARACTER SET UTF8
DEFINE JOB EXPORT_TO_BINARY_FORMAT
DESCRIPTION 'Export to the INDICDATA file'
(
  /* https://www.info.teradata.com/HTMLPubs/DB_TTU_16_00/Load_and_Unload_Utilities/B035-2436%E2%80%90086K
/2436ch03.05.3.html */
  APPLY TO OPERATOR ($FILE_WRITER()[@DataFileCount] ATTR
    (
    IndicatorMode = 'Y',
    OpenMode      = 'Write',
    Format        = @FormatType,
    DirectoryPath = @FileWriterDirectoryPath,
    FileName      = @FileWriterFileName,
    IOBufferSize  = 2048000
    )
  )
  SELECT * FROM OPERATOR ($EXPORT()[1] ATTR
    (
    SelectStmt        = @ExportSelectStmt,
    MaxDecimalDigits  = 38,
    DateForm          = 'INTEGERDATE',  /* ANSIDATE is hard to load in BTEQ */
    SpoolMode         = 'NoSpool',
    TdpId             = @SourceTdpId,
    UserName          = @SourceUserName,
    UserPassword      = @SourceUserPassword,
    QueryBandSessInfo = 'Action=TPT_EXPORT;SourceTable=@SourceTableName;Format=@FormatType;'
    )
  );
);
```

The login credential is supplied in tpt_job_var_file instead of via command line:

tpt_job_var_file

```
 SourceUserName=<td_use>
,SourceUserPassword=<td_pass>
,SourceTdpId=<td_pid>
```

Using BTEQ

The BTEQ script looks like below with proper **INDICDATA**, **Format**, **MaxDecimalDigits**, and **DateForm** in place and by default, **recordlength=max64** (**Formatted**) is applied, so **MAX1MB** must be explicitly specified when '**Formatted4**' mode is required.

BTEQ script example

```
SET SESSION DATEFORM=INTEGERDATE;
.SET SESSION CHARSET UTF8
.decimaldigits 38

.export indicdata recordlength=max1mb file=td_data_with_1mb_rowsize.dat
select * from foo.teradata_binary_table order by test_int;
.export reset
```

How to import

Using BTEQ

When **unicode** is used, the CHAR(n) column must be specified as CHAR(n x 3) in the **USING()** section. For example, `test_char` is defined as `CHAR(1) CHARACTER SET UNICODE` in DDL, when loading via BTEQ, it needs to occupy up to 3 bytes, therefore it appears as `CHAR(3)` in the USING().

If n x 3 rule is not applied, BTEQ can encounter the error like "Failure 2673 The source parcel length does not match data that was defined."

Here is the sample BTEQ script with proper **INDICDATA**, **Format**, **MaxDecimalDigits**, and **DateForm** to load a '**RECORDLENGTH=MAX1MB**' data file:

BTEQ script example

```
SET SESSION DATEFORM=INTEGERDATE;
.SET SESSION CHARSET UTF8
.decimaldigits 38

.IMPORT INDICDATA RECORDLENGTH=MAX1MB FILE=td_data_with_1mb_rowsize.teradata
.REPEAT *
USING(
      test_tinyint BYTEINT,
      test_smallint SMALLINT,
      test_int INTEGER,
      test_bigint BIGINT,
      test_double FLOAT,
      test_decimal DECIMAL(15,2),
      test_date DATE,
      test_timestamp TIMESTAMP(6),
      test_char CHAR(3),  -- CHAR(1) will occupy 3 bytes
      test_varchar VARCHAR(40),
      test_binary VARBYTE(500)
)
INSERT INTO foo.stg_teradata_binary_table
(
 test_tinyint, test_smallint, test_int, test_bigint, test_double, test_decimal,
 test_date, test_timestamp, test_char, test_varchar, test_binary
)
values (
:test_tinyint,
:test_smallint,
:test_int,
:test_bigint,
:test_double,
:test_decimal,
:test_date,
:test_timestamp,
:test_char,
:test_varchar,
:test_binary
);

.IMPORT RESET
```

Using TPT FastLoad

tbuild can load multiple gzip files in parallel, this makes TPT the best choice to bulk load big data files.

Here is a bash script example for how to call TPT FastLoad.

TPT FastLoad script example

```
staging_database=foo
staging_table=stg_table_name_up_to_30_chars
table_name_less_than_26chars=stg_table_name_up_to_30_c
file_dir=/data/foo/bar
job_id=<my_job_execution_id>

tbuild -C -f $td_import_template_file -v ${tpt_job_var_file} \
  -u "TargetWorkingDatabase='${staging_database}',TargetTable='${staging_table}',
     SourceDirectoryPath='${file_dir}',SourceFileName='*.teradata.gz',
     FileInstances=8,LoadInstances=1,
     Substr26TargetTable='${table_name_less_than_26chars}',
     TargetQueryBandSessInfo='TptLoad=${staging_table};JobId=${job_id};'"
```

The **td_import_template_file** looks like:

td_import_template_file

```
USING CHARACTER SET @Characterset
DEFINE JOB LOAD_JOB
DESCRIPTION 'Loading Data From File To Teradata Table'
(
set LogTable=@TargetWorkingDatabase||'.'||@Substr26TargetTable||'_LT';
set ErrorTable1=@TargetWorkingDatabase||'.'||@Substr26TargetTable||'_ET';
set ErrorTable2=@TargetWorkingDatabase||'.'||@Substr26TargetTable||'_UT';
set WorkTable=@TargetWorkingDatabase||'.'||@Substr26TargetTable||'_WT';
set ErrorTable=@TargetWorkingDatabase||'.'||@Substr26TargetTable||'_ET';

set LoadPrivateLogName=@TargetTable||'_load.log'
set UpdatePrivateLogName=@TargetTable||'_update.log'
set StreamPrivateLogName=@TargetTable||'_stream.log'
set InserterPrivateLogName=@TargetTable||'_inserter.log'
set FileReaderPrivateLogName=@TargetTable||'_filereader.log'

STEP PRE_PROCESSING_DROP_ERROR_TABLES
(
APPLY
('release mload '||@TargetTable||';'),
('drop table '||@LogTable||';'),
('drop table '||@ErrorTable||';'),
('drop table '||@ErrorTable1||';'),
('drop table '||@ErrorTable2||';'),
('drop table '||@WorkTable||';')
TO OPERATOR ($DDL);
);


STEP LOADING
(
    APPLY $INSERT TO OPERATOR ($LOAD() [@LoadInstances])
    SELECT * FROM OPERATOR ($FILE_READER() [@FileInstances]);
);
);
```

Please set the correct values in **tpt_job_var_file**, such as **SourceFormat**, **DateForm**, **MaxDecimalDigits**. Here is an example:

tpt_job_var_file

```
 Characterset='UTF8'
,DateForm='integerDate'
,MaxDecimalDigits=38

,TargetErrorLimit=100
,TargetErrorList=['3807','2580', '3706']
,TargetBufferSize=1024
,TargetDataEncryption='off'
,SourceOpenMode='Read'
,SourceFormat='Formatted'
,SourceIndicatorMode='Y'
,SourceMultipleReaders='Y'

,LoadBufferSize=1024
,UpdateBufferSize=1024
,LoadInstances=1

,TargetTdpId=<td_pid>
,TargetUserName=<td_user>
,TargetUserPassword=<td_pass>
```

Usage

Table Creating

Create table with specific Teradata properties

```
CREATE TABLE `teradata_binary_table_1mb`(
  `test_tinyint` tinyint,
  `test_smallint` smallint,
  `test_int` int,
  `test_bigint` bigint,
  `test_double` double,
  `test_decimal` decimal(15,2),
  `test_date` date,
  `test_timestamp` timestamp,
  `test_char` char(1),
  `test_varchar` varchar(40),
  `test_binary` binary
 )
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.teradata.TeradataBinarySerde'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.TeradataBinaryFileInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.TeradataBinaryFileOutputFormat'
TBLPROPERTIES (
  'teradata.timestamp.precision'='6',
  'teradata.char.charset'='UNICODE',
  'teradata.row.length'='1MB'
);
```

Default Teradata properties

```
'teradata.timestamp.precision'='6',
'teradata.char.charset'='UNICODE',
'teradata.row.length'='64KB'
```

Table Properties

Property Name	Property Value Set	Default Property Value	Note
teradata.row.length	(64KB, 1MB)	64KB	**64KB** corresponds to **Formatted** mode **1MB** corresponds to **Formatted4** mode
teradata.char.charset	(UNICODE, LATIN)	UNICODE	This decides how many bytes per char for CHAR data type 3 bytes per char for UNICODE 2 bytes per char for LATIN All the fields with CHAR type are controlled by this property (no separate specifying supported)
teradata.timestamp.precision	0-6	6	This decides how many bytes for TIMESTAMP data type. More details is here. All the fields with TIMESTAMP are controlled by this property (no separate specifying supported)

Teradata to Hive Type Conversion

Teradata Data Type	Teradata Data Type Definition	Hive Type	Hive Data Type Definition	Note
DATE	*DATE*	**DATE**	*DATE*	
TIMESTAMP	*TIMESTAMP(X)*	**TIMESTAMP**	*TIMESTAMP*	The decoding of TIMESTAMP precision is controlled by the table property **teradata.timestamp.precision**
BYTEINT	*BYTEINT*	**TINYINT**	*TINYINT*	
SMALLINT	*SMALLINT*	**SMALLINT**	*SMALLINT*	
INTEGER	*INTEGER|INT*	**INT**	*INT|INTEGER*	
BIGINT	*BIGINT*	**BIGINT**	*BIGINT*	
FLOAT	*FLOAT*	**DOUBLE**	*DOUBLE*	
DECIMAL	*DECIMAL(N,M)* *--Default DECIMAL(5, 0)*	**DECIMAL**	*DECIMAL(N,M)* *--Default DECIMAL (10, 0)*	
VARCHAR	*VARCHAR(X)*	**VARCHAR**	*VARCHAR(X)*	
CHAR	*CHAR(X)*	**CHAR**	*CHAR(X)*	The decoding of each CHAR is controlled by the table property **teradata.char.charset**
VARBYTE	*VARBYTE(X)*	**BINARY**	*BINARY*	

Serde Restriction

The TeradataBinarySerde has several restrictions:

- Only support simple data type listed above, other data type like INTERVAL, TIME, NUMBER, CLOB, BLOB in Teradata are not yet supported.
- Doesn't support the complex data type such as ARRAY, MAP.