Using Impala to query external JDBC data sources (Preview)

Date published: 2024-07-26
Date modified: 2024-07-26

Legal Notice

This document has been released as part of a technical preview for features described herein. Technical preview components are provided as a convenience to our customers for their evaluation and trial usage. These components are provided ‘as is’ without warranty or support. Further, Cloudera assumes no liability for the usage of technical preview components, which should be used by customers at their own risk.

Contents

Legal Notice **2**
Contents **3**
Impala support for external JDBC tables **4**
Prerequisites 4
Syntax 4
Table Properties 6
Supported Data Types 6
Limitations 7
Securing the JDBC password 7
Support for case-sensitive table and column names 8
Modifying the external JDBC table **8**
Querying external JDBC tables **8**
Query options for external JDBC tables **9**

Impala support for external JDBC tables

Apache Impala now supports reading from external JDBC data sources. An external JDBC table represents a table or a view in a remote RDBMS database or another Impala cluster. Using external JDBC tables, you can connect Impala to a database, such as MySQL, PostgreSQL, or another Impala cluster and read the data in the remote tables.

***Note**: This feature is in technical preview and not recommended for use in production deployments. Cloudera recommends that you try this feature in test and development environments.*

Prerequisites

The Impala package does not include JDBC drivers of the remote databases. Ensure that the MySQL, Postgres, and Impala JDBC drivers are uploaded to a location in HDFS, Ozone, or Amazon S3 that can be read by Impala.

Syntax

To connect to a remote database, you create an external JDBC table with the appropriate table properties, such as the database type, JDBC URL, driver class, driver file location, JDBC username and password, and name of the remote table to be mapped to the Impala external table.

Syntax:

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [db_name.]table_name
  (col_name data_type,
   ....)
  STORED BY JDBC
  TBLPROPERTIES (
    "database.type"="[***VALUE***]",
    "jdbc.url"="[***VALUE***]",
    "jdbc.driver"="[***VALUE***]",
    "driver.url"="[***VALUE***]",
    "dbcp.username"="[***VALUE***]",
    "dbcp.password"="[***VALUE***]",
    "table"="[***VALUE***]");
```

Examples:

- Creating an external JDBC table to map a table in a remote PostgreSQL database:

```
CREATE EXTERNAL TABLE student_jdbc (
      id INT,
      bool_col BOOLEAN,
      tinyint_col TINYINT,
      smallint_col SMALLINT,
      int_col INT,
      bigint_col BIGINT,
      float_col FLOAT,
      double_col DOUBLE,
      date_col DATE,
      string_col STRING,
      timestamp_col TIMESTAMP)
STORED BY JDBC
TBLPROPERTIES (
      "database.type"="POSTGRES",
      "jdbc.url"="jdbc:postgresql://[***IP_ADDRESS***]:[***PORT***]
   /[***DATABASE NAME***]",
      "jdbc.driver"="org.postgresql.Driver",
      "driver.url"="/test-warehouse/data-sources/jdbc-drivers/postg
   resql-jdbc.jar",
"dbcp.username"="[***USERNAME***]",
"dbcp.password"="[***PASSWORD***]",
"table"="student");
```

- Creating an external JDBC table to map a table in another Impala cluster:

```
CREATE EXTERNAL TABLE student_jdbc (
  id INT,
  bool_col BOOLEAN,
  tinyint_col TINYINT,
  smallint_col SMALLINT,
  int_col INT,
  bigint_col BIGINT,
  float_col FLOAT,
  double_col DOUBLE,
  date_col DATE,
  string_col STRING,
  timestamp_col TIMESTAMP)
STORED BY JDBC
TBLPROPERTIES (
  "database.type"="IMPALA",
  "jdbc.url"="jdbc:impala://[***IP
ADDRESS***]:[***PORT***]/[***DATABASE NAME***]",
  "jdbc.auth"="AuthMech=3",
  "jdbc.properties"="MEM_LIMIT=1000000000, MAX_ERRORS = 10000",
```

```
  "jdbc.driver"="com.cloudera.impala.jdbc.Driver",
"driver.url"="hdfs://test-warehouse/data-sources/jdbc-drivers/I
mpalaJDBC42.jar",
  "dbcp.username"="[***USERNAME***]",
"dbcp.password.keystore"="jceks://hdfs/test-warehouse/data-sour
ces/test.jceks",
  "dbcp.password.key"="[***KEY***]",
  "table"="student");
```

Table Properties

While creating an external JDBC table, you are required to specify the following table properties:

- `database.type`: POSTGRES, MYSQL, IMPALA
- `jdbc.url`: JDBC connection string with the required parameters — database type, hostname, port number, and database name.

  Example: “jdbc:impala://10.96.132.138:21050/sample_db”.

- `jdbc.driver`: Class name of the JDBC driver
- `driver.url`: URL to download the JAR file package that is used to access the external database
- table: Name of the table in the remote database that you want to map in Impala

Besides the above required properties, you can also specify optional parameters that allow you to use different authentication methods, allow case sensitive column names in remote tables, or to specify additional database properties:

- `jdbc.auth`: Authentication mechanism of the JDBC driver
- `dbcp.username`: JDBC username
- `dbcp.password`: JDBC password in clear text.

  Note: Storing JDBC passwords in clear text is not recommended in production environments. The recommended way is to store the password in a Java keystore file.

- `dbcp.password.key`: Key of the Java keystore
- `dbcp.password.keystore`: Location of the keystore file
- `jdbc.properties`: Additional properties applied to database engines, like Impala Query options. The properties are specified as comma-separated "key-value" pairs.
- `jdbc.fetch.size`: Number of rows to fetch in a batch
- `column.mapping`: Mapping of column names between external table and Impala JDBC table.

Supported Data Types

The following column data types are supported for an external JDBC table:

- Numeric data type: boolean, tinyint, smallint, int, bigint, float, double
- Decimal with scale and precision
- String type: string
- Date
- Timestamp

Note: The following data types are currently not supported — char, varchar, and binary. Complex data type: struct, map, array and nested type.

Limitations

- Unsupported column data types: char, varchar, binary. Complex data type - struct, map, array, and nested type
- JDBC tables have to be defined one table at a time
- Writing to a JDBC table is not supported
- Only support binary predicates with operators =, !=, <=, >=, <, > to be pushed to RDBMS

Securing the JDBC password

The `dbcp.password` table property stores the JDBC password in clear text. To avoid the risk of a password leak, the `SHOW CREATE TABLE <table-name>` and `DESCRIBE FORMATTED | EXTENDED <table-name>` statements mask the value of the `dbcp.password` table property in their outputs.

In production environments, it is recommended that you do not store the JDBC password in clear text using the `dbcp.password` table property. Instead, you can store the password in a Java Keystore file on HDFS or on cloud storage like Amazon S3 using the following command:

- Creating a Java keystore file on HDFS with the key as "host1.password" and password as "passwd1":

  ```
  hadoop credential create host1.password -provider
  jceks://hdfs/user/foo/test.jceks -v passwd1
  ```

- Creating a Java keystore file on Amazon S3 with the key as "impala" and password as "passwd2":

  ```
  hadoop credential create impala -provider
  jceks://s3a@dw-impala-test/jceks/demo.jceks -v passwd2
  ```

For more information, see the Apache Hadoop CredentialProvider API guide.

Support for case-sensitive table and column names

The column names of tables in the remote database can be different from the external JDBC table schema. For example, Postgres allows case-sensitive column names, however, Impala saves column names in lowercase. In such situations, you can set the `column.mapping` table property to map column names between Impala external JDBC tables and the remote tables.

Example:

```
"column.mapping"="id=id, bool_col=Bool_col, tinyint_col=Tinyint_col,
smallint_col=Smallint_col, int_col=Int_col, bigint_col=Bigint_col,
float_col=Float_col, double_col=Double_col, date_col=date_col,
string_col=String_col, timestamp=Timestamp");
```

Modifying the external JDBC table

You can use the `ALTER TABLE` statement to add, drop, or modify columns, or modify the table properties of existing external JDBC tables. The syntax is the same as the other Impala tables.

Example:

- Using ALTER TABLE statement to add, drop, or modify columns:

  ```
  ALTER TABLE student_jdbc ADD COLUMN IF NOT EXISTS date_col DATE;
  ALTER TABLE student_jdbc DROP COLUMN int_col;
  ALTER TABLE student_jdbc CHANGE COLUMN date_col timestamp_col
  TIMESTAMP;
  ```

- Using ALTER TABLE statement to modify table properties:

  ```
  ALTER TABLE student_jdbc
  SET TBLPROPERTIES ("dbcp.username"="impala",
  "dbcp.password"="password");
  ```

Querying external JDBC tables

Querying or reading external JDBC tables is the same as querying regular tables in Impala. You can use SELECT statements to query data and can also join the external table with other tables across databases. However, do note that the metadata for the external tables is not persisted in Hive Metastore (HMS).

Example:

```
SELECT * from student_jdbc;
```

Query options for external JDBC tables

A new query option, **CLEAN_DBCP_DS_CACHE** is added to save the DBCP SQL DataSource objects in the cache for a longer period of time. This allows the DBCP connection pools to be reused across multiple queries. When the value is set to false, the DBCP SQL DataSource object is not closed when its reference count is 0. The SQL DataSource object is kept in cache until the object is idle for more than 5 minutes.

Type: BOOLEAN
Default: True (1)