MySQL Foreign Data Wrapper Guide

Version 2.5.5

1 What's New

The following features are added to create MySQL Foreign Data Wrapper 2.5.5 :

- Support for EDB Postgres Advanced Server 13.
- Support for Ubuntu 20.04 LTS platform.

2 Requirements Overview

Supported Versions

The MySQL Foreign Data Wrapper is certified with EDB Postgres Advanced Server 9.5 and above.

Supported Platforms

The MySQL Foreign Data Wrapper is supported on the following platforms:

Linux x86-64

- RHEL 8.x/7.x
- CentOS 8.x/7.x
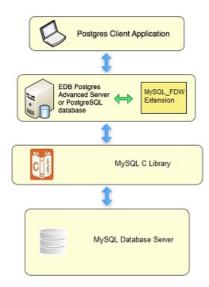- OEL 8.x/7.x
- Ubuntu 20.04/18.04 LTS

- Debian 10.x/9.x

Linux on IBM Power8/9 (LE)

- RHEL 7.x

3 Architecture Overview

The MySQL data wrapper provides an interface between a MySQL server and a Postgres database. It transforms a Postgres statement (SELECT / INSERT / DELETE / UPDATE) into a query that is understood by the MySQL database.

4 Installing the MySQL Foreign Data Wrapper

The MySQL Foreign Data Wrapper can be installed with an RPM package. During the installation process, the installer will satisfy software prerequisites.

Installing the MySQL Foreign Data Wrapper using an RPM Package

You can install the MySQL Foreign Data Wrapper using an RPM package on the following platforms:

- RHEL 7
- RHEL 8
- CentOS 7
- CentOS 8

On RHEL 7

Before installing the MySQL Foreign Data Wrapper, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Enable the optional, extras, and HA repositories:

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the MySQL Foreign Data Wrapper.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-
$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing MySQL Foreign Data Wrapper

After saving your changes to the configuration file, use the following command to install the MySQL Foreign Data Wrapper:

```
yum install edb-as<xx>-mysql<x>_fdw
```

where xx is the server version number, and x is the supported release version number of MySQL. For example, to install MySQL 5.0 on RHEL 7:

```
yum install edb-as<xx>-mysql5_fdw
```

Note

MySQL 8.0 and MySQL 5.0 RPMs are available for RHEL 7 platform.

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

On RHEL 8

Before installing the MySQL Foreign Data Wrapper, you must install the following prerequisite packages, and request credentials from EDB:

Install the `epel-release` package:

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Enable the `codeready-builder-for-rhel-8-\*-rpms` repository:

```
ARCH=$( /bin/arch )
subscription-manager repos --enable "codeready-builder-for-rhel-
8-${ARCH}-rpms"
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the MySQL Foreign Data Wrapper.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
dnf -y https://yum.enterprisedb.com/edbrepos/edb-repo-
latest.noarch.rpm
```

The repository configuration file is named edb.repo . The file resides in /etc/yum.repos.d .

Modifying the file, providing your user name and password

After creating the edb.repo file, use your choice of editor to ensure that the value of the enabled parameter is 1 , and replace the username and password placeholders in the baseurl specification with the name and password of a registered EDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-
```

```
$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing MySQL Foreign Data Wrapper

After saving your changes to the configuration file, use the below command to install the MySQL Foreign Data Wrapper:

```
dnf install edb-as<xx>-mysql8_fdw
```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter y , and press Return to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

On CentOS 7

Before installing the MySQL Foreign Data Wrapper, you must install the following prerequisite packages, and request credentials from EDB:

Install the epel-release package:

```
 yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Note

You may need to enable the [extras] repository definition in the

CentOS-Base.repo file (located in /etc/yum.repos.d).

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the MySQL Foreign Data Wrapper.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
 yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

The repository configuration file is named edb.repo . The file resides in /etc/yum.repos.d .

Modifying the file, providing your user name and password

After creating the edb.repo file, use your choice of editor to ensure that the value of the enabled parameter is 1 , and replace the username and password placeholders in the baseurl specification with the name and password of a registered EDB user.

```
 [edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-$basearch
enabled=1
```

```
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing MySQL Foreign Data Wrapper

After saving your changes to the configuration file, use the following command to install the MySQL Foreign Data Wrapper:

```
yum install edb-as<xx>-mysql<x>_fdw
```

where xx is the server version number, and x is the supported release version number of MySQL. For example, to install MySQL 5.0 on CentOS 7:

```
yum install edb-as<xx>-mysql5_fdw
```

Note

MySQL 8.0 and MySQL 5.0 RPMs are available for CentOS 7 platform.

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter y , and press Return to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

On CentOS 8

Before installing the MySQL Foreign Data Wrapper, you must install the following prerequisite packages, and request credentials from EDB:

Install the epel-release package:

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-
latest-8.noarch.rpm
```

Enable the `PowerTools` repository:

```
dnf config-manager --set-enabled PowerTools
```

You must also have credentials that allow access to the EDB repository. For information about requesting credentials, visit:

https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%2004-09-2019.pdf

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the MySQL Foreign Data Wrapper.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-
latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EDB user.

```
[edb]
```

```
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:
<password>@yum.enterprisedb.com/edb/redhat/rhel-$releasever-
$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

Installing MySQL Foreign Data Wrapper

After saving your changes to the configuration file, use the following command to install the MySQL Foreign Data Wrapper:

```
dnf install edb-as<xx>-mysql8_fdw
```

where xx is the server version number.

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

Installing the MySQL Foreign Data Wrapper on a Debian or Ubuntu Host

To install the MySQL Foreign Data Wrapper on a Debian or Ubuntu host, you must have credentials that allow access to the EDB repository. To request credentials for the repository, visit the EDB website.

The following steps will walk you through on using the EDB apt repository to install a DEB package. When using the commands, replace the username and password with the credentials provided by EDB.

1. Assume superuser privileges:

sudo su –

1. Configure the EnterpriseDB repository:

On Debian 9:

```
sh -c 'echo "deb
https://username:password@apt.enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release -cs) main" >
/etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
```

On Debian 10:

1. Set up the EDB repository:

```
sh -c 'echo "deb [arch=amd64]
https://apt.enterprisedb.com/$(lsb_release -cs)-edb/
$(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-
$(lsb_release -cs).list'
```

1. Substitute your EDB credentials for the username and password in the following command:

```
sh -c 'echo "machine apt.enterprisedb.com login <username>
password <password>" > /etc/apt/auth.conf.d/edb.conf'
```

2. Add support to your system for secure APT repositories:

apt-get install apt-transport-https

1. Add the EBD signing key:

```
wget -q -O - https://username:password
@apt.enterprisedb.com/edb-deb.gpg.key | apt-key add -
```

1. Update the repository metadata:

```
apt-get update
```

1. Install DEB package:

```
apt-get install edb-as<xx>-mysql-fdw
```

where xx is the server version number.

5   Features of the MySQL Foreign Data Wrapper

The key features of the MySQL Foreign Data Wrapper are listed below:

Writable FDW

MySQL Foreign Data Wrapper provides the write capability. Users can insert, update and delete data in the remote MySQL tables by inserting, updating and deleting the data locally in the foreign tables. MySQL foreign data wrapper uses the Postgres type casting mechanism to provide opposite type casting between MySQL and Postgres data types.

Note

The first column of MySQL table must have unique/primary key for DML to work.

See also:

Example: Using the MySQL Foreign Data Wrapper <example_using_the_mysql_data_adapter>

Data Type Mappings

Connection Pooling

MySQL_FDW establishes a connection to a foreign server during the first query that uses a foreign table associated with the foreign server. This connection is kept and reused for subsequent queries in the same session.

Where Clause Push-down

MySQL Foreign Data Wrapper allows the push-down of WHERE clause to the foreign server for execution. This fearure optimizes remote queries to reduce the number of rows transferred from foreign servers.

Column Push-down

MySQL Foreign Data Wrapper supports the column push-down. As a result, the query brings back only those columns that are a part of the select target list.

Prepared Statement

MySQL Foreign Data Wrapper supports Prepared Statement. The select queries uses prepared statements instead of simple query protocol.

Import Foreign Schema

MySQL Foreign Data Wrapper supports Import Foreign Schema which enables the local host to import table definitions on the EDB Postgres Advanced Server from the MySQL server. The new foreign tables are created with the corresponding column types and same table name as that of remote tables in the existing local schema.

See also:

Example: Import Foreign Schema <example_import_foreign_schema>

Automated Cleanup

MySQL Foreign Data Wrappper allows the cleanup of foreign tables in a single operation using DROP EXTENSION command. This feature is specifically useful when a foreign table is set for a temporary purpose, as in case of data migration. The syntax:

```
DROP EXTENSION mysql_fdw CASCADE;
```

For more information, see DROP EXTENSION.

6   Configuring the MySQL Foreign Data Wrapper

Before using the MySQL Foreign Data Wrapper, you must:

1. Use the CREATE EXTENSION command to create the MySQL Foreign Data Wrapper extension on the Postgres host.
2. Use the CREATE SERVER command to define a connection to the MySQL server.
3. Use the CREATE USER MAPPING command to define a mapping that associates a Postgres role with the server.
4. Use the CREATE FOREIGN TABLE command to define a single table in the Postgres database that corresponds to a table that resides on the MySQL server or use the IMPORT FOREIGN SCHEMA command to import multiple remote tables in the local schema.

CREATE EXTENSION

Use the CREATE EXTENSION command to create the mysql_fdw extension. To invoke the command, use your client of choice (for example, psql) to connect to the Postgres database from which you will be querying the MySQL server, and invoke the command:

```
CREATE EXTENSION [IF NOT EXISTS] mysql_fdw [WITH] [SCHEMA schema_name];
```

Parameters

IF NOT EXISTS

Include the IF NOT EXISTS clause to instruct the server to issue a notice instead of throwing an error if an extension with the same name already exists.

`schema_name`

> Optionally specify the name of the schema in which to install the extension's objects.

Example

The following command installs the MySQL foreign data wrapper:

> `CREATE EXTENSION mysql_fdw;`

For more information about using the foreign data wrapper `CREATE EXTENSION` command, see:

> https://www.postgresql.org/docs/current/static/sql-createextension.html.

CREATE SERVER

Use the `CREATE SERVER` command to define a connection to a foreign server. The syntax is:

```
CREATE SERVER server_name FOREIGN DATA WRAPPER
mysql_fdw
   [OPTIONS (option 'value' [, ...])]
```

The role that defines the server is the owner of the server; use the `ALTER SERVER` command to reassign ownership of a foreign server. To create a foreign server, you must have `USAGE` privilege on the foreign-data wrapper specified in the `CREATE SERVER` command.

Parameters

`server_name`

> Use `server_name` to specify a name for the foreign server. The

server name must be unique within the database.

FOREIGN_DATA_WRAPPER

Include the FOREIGN_DATA WRAPPER clause to specify that the server should use the mysql_fdw foreign data wrapper when connecting to the cluster.

OPTIONS

Use the OPTIONS clause of the CREATE SERVER command to specify connection information for the foreign server. You can include:

Option	Description
host	The address or hostname of the MySQL server. The default value is 127.0.0.1.
port	The port number of the MySQL Server. The default is 3306.
secure_auth	Use to enable or disable secure authentication. The default value is true.
init_command	The SQL statement to execute when connecting to the MySQL server.
ssl_key	The path name of the client private key file.
ssl_cert	The path name of the client public key certificate file.
ssl_ca	The path name of the Certificate Authority (CA) certificate file. This option, if used, must specify the same certificate used by the server.
ssl_capath	The path name of the directory that contains trusted SSL CA certificate files.
ssl_cipher	The list of permissible ciphers for SSL encryption.

Option	Description
use_remote_estimate	Include the use_remote_estimate to instruct the server to use EXPLAIN commands on the remote server when estimating processing costs. By default, use_remote_estimate is false.

Example

The following command creates a foreign server named `mysql_server` that uses the `mysql_fdw` foreign data wrapper to connect to a host with an IP address of `127.0.0.1`:

```
CREATE SERVER mysql_server FOREIGN DATA WRAPPER
mysql_fdw OPTIONS (host '127.0.0.1', port '3306');
```

The foreign server uses the default port (`3306`) for the connection to the client on the MySQL cluster.

For more information about using the `CREATE SERVER` command, see:

https://www.postgresql.org/docs/current/static/sql-createserver.html

CREATE USER MAPPING

Use the `CREATE USER MAPPING` command to define a mapping that associates a Postgres role with a foreign server:

```
CREATE USER MAPPING FOR role_name SERVER server_name
    [OPTIONS (option 'value' [, ...])];
```

You must be the owner of the foreign server to create a user mapping for that server.

Parameters

role_name

Use role_name to specify the role that will be associated with the foreign server.

server_name

Use server_name to specify the name of the server that defines a connection to the MySQL cluster.

OPTIONS

Use the OPTIONS clause to specify connection information for the foreign server.

username : the name of the user on the MySQL server.

password : the password associated with the username.

Example

The following command creates a user mapping for a role named enterprisedb ; the mapping is associated with a server named mysql_server :

```
CREATE USER MAPPING FOR enterprisedb SERVER mysql_server;
```

If the database host uses secure authentication, provide connection credentials when creating the user mapping:

```
CREATE USER MAPPING FOR public SERVER mysql_server OPTIONS (username 'foo', password 'bar');
```

The command creates a user mapping for a role named public that is associated with a server named mysql_server . When connecting to the MySQL server, the server will authenticate as foo , and provide a

password of `bar`.

For detailed information about the `CREATE USER MAPPING` command, see:

> https://www.postgresql.org/docs/current/static/sql-createusermapping.html

CREATE FOREIGN TABLE

A foreign table is a pointer to a table that resides on the MySQL host. Before creating a foreign table definition on the Postgres server, connect to the MySQL server and create a table; the columns in the table will map to columns in a table on the Postgres server. Then, use the `CREATE FOREIGN TABLE` command to define a table on the Postgres server with columns that correspond to the table that resides on the MySQL host. The syntax is:

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ OPTIONS ( option 'value' [, ... ] ) ] [
COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
  SERVER server_name [ OPTIONS ( option 'value' [, ... ] ) ]
```

where `column_constraint` is:

```
[ CONSTRAINT constraint_name ]
{ NOT NULL | NULL | CHECK (expr) [ NO INHERIT ] | DEFAULT
default_expr }
```

and `table_constraint` is:

```
[ CONSTRAINT constraint_name ] CHECK (expr) [ NO INHERIT ]
```

Parameters

table_name

Specifies the name of the foreign table; include a schema name to specify the schema in which the foreign table should reside.

IF NOT EXISTS

Include the IF NOT EXISTS clause to instruct the server to not throw an error if a table with the same name already exists; if a table with the same name exists, the server will issue a notice.

column_name

Specifies the name of a column in the new table; each column should correspond to a column described on the MySQL server.

data_type

Specifies the data type of the column; when possible, specify the same data type for each column on the Postgres server and the MySQL server. If a data type with the same name is not available, the Postgres server will attempt to cast the data type to a type compatible with the MySQL server. If the server cannot identify a compatible data type, it will return an error.

COLLATE collation

Include the COLLATE clause to assign a collation to the column; if not specified, the column data type's default collation is used.

INHERITS (parent_table [, ...])

Include the INHERITS clause to specify a list of tables from which the new foreign table automatically inherits all columns. Parent tables can be plain tables or foreign tables.

CONSTRAINT constraint_name

Specify an optional name for a column or table constraint; if not specified, the server will generate a constraint name.

NOT NULL

Include the NOT NULL keywords to indicate that the column is not allowed to contain null values.

NULL

Include the NULL keywords to indicate that the column is allowed to contain null values. This is the default.

CHECK (expr) [NO INHERIT]

Use the CHECK clause to specify an expression that produces a Boolean result that each row in the table must satisfy. A check constraint specified as a column constraint should reference that column's value only, while an expression appearing in a table constraint can reference multiple columns.

A CHECK expression cannot contain subqueries or refer to variables other than columns of the current row.

Include the NO INHERIT keywords to specify that a constraint should not propagate to child tables.

DEFAULT default_expr

Include the DEFAULT clause to specify a default data value for the column whose column definition it appears within. The data type of the default expression must match the data type of the column.

SERVER server_name [OPTIONS (option 'value' [, ...])]

To create a foreign table that will allow you to query a table that

resides on a MySQL file system, include the SERVER clause and specify the server_name of the foreign server that uses the MySQL data adapter.

Use the OPTIONS clause to specify the following options and their corresponding values:

option	value
dbname	The name of the database on the MySQL server; the database name is required.
table_name	The name of the table on the MySQL server; the default is the name of the foreign table.
max_blob_size	The maximum blob size to read without truncation.

Example

To use data that is stored on MySQL server, you must create a table on the Postgres host that maps the columns of a MySQL table to the columns of a Postgres table. For example, for a MySQL table with the following definition:

```
CREATE TABLE warehouse (
 warehouse_id     INT PRIMARY KEY,
 warehouse_name   TEXT,
 warehouse_created TIMESTAMP);
```

You should execute a command on the Postgres server that creates a comparable table on the Postgres server:

```
CREATE FOREIGN TABLE warehouse
(
 warehouse_id     INT,
 warehouse_name   TEXT,
 warehouse_created TIMESTAMP
)
```

```
SERVER mysql_server
      OPTIONS (dbname 'db', table_name 'warehouse');
```

Include the SERVER clause to specify the name of the database stored on the MySQL server and the name of the table (warehouse) that corresponds to the table on the Postgres server.

For more information about using the CREATE FOREIGN TABLE command, see:

https://www.postgresql.org/docs/current/static/sql-createforeigntable.html

Note

MySQL foreign data wrapper supports the write capability feature.

Data Type Mappings

When using the foreign data wrapper, you must create a table on the Postgres server that mirrors the table that resides on the MySQL server. The MySQL data wrapper will automatically convert the following MySQL data types to the target Postgres type:

MySQL	Postgres
BIGINT	BIGINT/INT8
BOOLEAN	SMALLINT
BLOB	BYTEA
CHAR	CHAR
DATE	DATE
DOUBLE	DOUBLE PRECISION/FLOAT8
FLOAT	FLOAT/FLOAT4
INT/INTEGER	INT/INTEGER/INT4

MySQL	Postgres
LONGTEXT	TEXT
SMALLINT	SMALLINT/INT2
TIMESTAMP	TIMESTAMP
VARCHAR()	VARCHAR()/CHARCTER VARYING()

Note

For `ENUM` data type:

MySQL accepts `enum` value in string form. You must create exactly same `enum` listing on Advanced Server as that is present on MySQL server. Any sort of inconsistency will result in an error while fetching rows with values not known on the local server.

Also, when the given `enum` value is not present at MySQL side but present at Postgres/Advanced Server side, an empty string (`"`) is inserted as a value at MySQL side for the `enum` column. To select from such a table having enum value as `"`, create an `enum` type at Postgres side with all valid values and `"`.

IMPORT FOREIGN SCHEMA

Use the `IMPORT FOREIGN SCHEMA` command to import table definitions on the Postgres server from the MySQL server. The new foreign tables are created with the same column definitions as that of remote tables in the existing local schema. The syntax is:

```
IMPORT FOREIGN SCHEMA remote_schema
    [ { LIMIT TO | EXCEPT } ( table_name [, ...] ) ]
    FROM SERVER server_name
    INTO local_schema
    [ OPTIONS ( option 'value' [, ... ] ) ]
```

Parameters

remote_schema

Specifies the remote schema (MySQL database) to import from.

LIMIT TO (table_name [, ...])

By default, all views and tables existing in a particular database on the MySQL host are imported. Using this option, you can limit the list of tables to a specified subset.

EXCEPT (table_name [, ...])

By default, all views and tables existing in a particular database on the MySQL host are imported. Using this option, you can exclude specified foreign tables from the import.

SERVER server_name

Specify the name of server to import foreign tables from.

local_schema

Specify the name of local schema where the imported foreign tables must be created.

OPTIONS

Use the OPTIONS clause to specify the following options and their corresponding values:

Y{0.6}|

Option	Description

Option	Description
import_default	Controls whether column DEFAULT expressions are included in the definitions of foreign tables imported from a foreign server. The default is false.
import_not_null	Controls whether column NOT NULL constraints are included in the definitions of foreign tables imported from a foreign server. The default is true.

Example

For a MySQL table created in the `edb` database with the following definition:

```
CREATE TABLE color(cid INT PRIMARY KEY, cname TEXT);
INSERT INTO color VALUES (1, 'Red');
INSERT INTO color VALUES (2, 'Green');
INSERT INTO color VALUES (3, 'Orange');

CREATE TABLE fruit(fid INT PRIMARY KEY, fname TEXT);
INSERT INTO fruit VALUES (1, 'Orange');
INSERT INTO fruit VALUES (2, 'Mango');
```

You should execute a command on the Postgres server that imports a comparable table on the Postgres server:

```
IMPORT FOREIGN SCHEMA edb FROM SERVER mysql_server
INTO public;

SELECT * FROM color;

 cid | cname
-----+--------
   1 | Red
```

```
   2 | Green
   3 | Orange
(3 rows)


SELECT * FROM fruit;


fid | fname
-----+--------
   1 | Orange
   2 | Mango
(2 rows)
```

The command imports table definitions from a remote schema `edb` on server `mysql_server` and then creates the foreign tables in local schema `public`.

For more information about using the `IMPORT FOREIGN SCHEMA` command, see:

> https://www.postgresql.org/docs/current/static/sql-importforeignschema.html

7   Example: Using the MySQL Foreign Data Wrapper

Access data from Advanced Server and connect to psql. Once you are connected to psql, follow the below steps:

```
-- load extension first time after install
CREATE EXTENSION mysql_fdw;
```

```
-- create server object
CREATE SERVER mysql_server
      FOREIGN DATA WRAPPER mysql_fdw
      OPTIONS (host '127.0.0.1', port '3306');


-- create user mapping
CREATE USER MAPPING FOR postgres
   SERVER mysql_server OPTIONS (username 'foo', password 'bar');


-- create foreign table
CREATE FOREIGN TABLE warehouse
(
 warehouse_id     INT,
 warehouse_name    TEXT,
 warehouse_created TIMESTAMP
)
SERVER mysql_server
      OPTIONS (dbname 'db', table_name 'warehouse');


-- insert new rows in table
INSERT INTO warehouse values (1, 'UPS', current_date);
INSERT INTO warehouse values (2, 'TV', current_date);
INSERT INTO warehouse values (3, 'Table', current_date);


-- select from table
SELECT * FROM warehouse ORDER BY 1;

warehouse_id | warehouse_name | warehouse_created
-------------+----------------+------------------
        1 | UPS          | 10-JUL-20 00:00:00
        2 | TV           | 10-JUL-20 00:00:00
        3 | Table        | 10-JUL-20 00:00:00


-- delete row from table
```

```
DELETE FROM warehouse where warehouse_id = 3;

-- update a row of table
UPDATE warehouse set warehouse_name = 'UPS_NEW' where
warehouse_id = 1;


-- explain a table with verbose option
EXPLAIN VERBOSE SELECT warehouse_id, warehouse_name
FROM warehouse WHERE warehouse_name LIKE 'TV' limit 1;

                             QUERY PLAN
--------------------------------------------------------------------------------------
-------------
Limit  (cost=10.00..11.00 rows=1 width=36)
     Output: warehouse_id, warehouse_name
     -> Foreign Scan on public.warehouse  (cost=10.00..1010.00
rows=1000 width=36)
            Output: warehouse_id, warehouse_name
            Local server startup cost: 10
            Remote query: SELECT `warehouse_id`, `warehouse_name`
FROM `db`.`warehouse` WHERE ((`warehouse_name` LIKE BINARY
'TV'))
```

8   Example: Import Foreign Schema

Access data from Advanced Server and connect to psql. Once you are
connected to psql, follow the below steps:

```
 -- load extension first time after install
CREATE EXTENSION mysql_fdw;
```

```
-- create server object
CREATE SERVER mysql_server
      FOREIGN DATA WRAPPER mysql_fdw
      OPTIONS (host '127.0.0.1', port '3306');


-- create user mapping
CREATE USER MAPPING FOR postgres
    SERVER mysql_server OPTIONS (username 'foo', password 'bar');


-- import foreign schema
IMPORT FOREIGN SCHEMA edb FROM SERVER mysql_server
INTO public;

 SELECT * FROM color;
 cid | cname
-----+--------
   1 | Red
   2 | Green
   3 | Orange

SELECT * FROM fruit;
 fid | fname
-----+--------
   1 | Orange
   2 | Mango
```

9   Identifying the MySQL Foreign Data Wrapper Version

The MySQL Foreign Data Wrapper includes a function that you can use to identify the currently installed version of the .so file for the data wrapper. To use the function, connect to the Postgres server, and enter:

```
SELECT mysql_fdw_version();
```

The function returns the version number:

```
mysql_fdw_version
-----------------
<xxxxx>
```

10   Troubleshooting

In case you are experiencing issues with using MySQL 8 and MySQL_FDW, below is a list of solutions to some frequently seen issues:

Authentication plugin 'caching_sha2_password' Error

```
ERROR:  failed to connect to MySQL: Authentication plugin
'caching_sha2_password' cannot be loaded:
/usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open
shared  object file: No such file or directory
```

Specify the authentication plugin as mysql_native_password and set a cleartext password value. The syntax:

```
ALTER USER 'username'@'host' IDENTIFIED WITH
mysql_native_password BY '<password>';
```

Note

Refer to MySQL 8 documentation for more details on the above error.