MySQL Foreign Data Wrapper

Version 2

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Built at 2025-12-12T14:43:59

1 MySQL Foreign Data Wrapper 3
2 Release notes 4
2.1 MySQL Foreign Data Wrapper 2.9.3 release notes 5
2.2 MySQL Foreign Data Wrapper 2.9.2 release notes 6
2.3 MySQL Foreign Data Wrapper 2.9.1 release notes 7
2.4 MySQL Foreign Data Wrapper 2.9.0 release notes 8
2.5 MySQL Foreign Data Wrapper 2.8.0 release notes 9
2.6 MySQL Foreign Data Wrapper 2.7.0 release notes 10
2.7 MySQL Foreign Data Wrapper 2.6.0 release notes 11
2.8 MySQL Foreign Data Wrapper 2.5.5 release notes 12
2.9 MySQL Foreign Data Wrapper 2.5.3 release notes 13
2.10 MySQL Foreign Data Wrapper 2.5.1 release notes 14
3 Supported platforms 15
4 Architecture overview 16
5 Key features 17
6 Installing MySQL Foreign Data Wrapper on Linux 20
6.1 Installing MySQL Foreign Data Wrapper on Linux x86 (amd64) 21
6.1.1 Installing MySQL Foreign Data Wrapper on RHEL 9 or OL 9 x86_64 22
6.1.2 Installing MySQL Foreign Data Wrapper on RHEL 8 or OL 8 x86_64 23
6.1.3 Installing MySQL Foreign Data Wrapper on AlmaLinux 9 or Rocky Linux 9 x86_64 24
6.1.4 Installing MySQL Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64 25
6.1.5 Installing MySQL Foreign Data Wrapper on SLES 15 x86_64 26
6.1.6 Installing MySQL Foreign Data Wrapper on Ubuntu 24.04 x86_64 28
6.1.7 Installing MySQL Foreign Data Wrapper on Ubuntu 22.04 x86_64 29
6.1.8 Installing MySQL Foreign Data Wrapper on Debian 12 x86_64 30
6.1.9 Installing MySQL Foreign Data Wrapper on Debian 11 x86_64 32
6.2 Installing MySQL Foreign Data Wrapper on Linux AArch64 (ARM64) 33
6.2.1 Installing MySQL Foreign Data Wrapper on RHEL 9 or OL 9 arm64 34
6.2.2 Installing MySQL Foreign Data Wrapper on Debian 12 arm64 35
6.3 Installing MySQL Foreign Data Wrapper on Linux IBM Power (ppc64le) 37
6.3.1 Installing MySQL Foreign Data Wrapper on RHEL 9 ppc64le 38
6.3.2 Installing MySQL Foreign Data Wrapper on RHEL 8 ppc64le 39
6.3.3 Installing MySQL Foreign Data Wrapper on SLES 15 ppc64le 40
7 Initial configuration 42
8 Upgrading 51
9 Uninstalling 52
10 Example: End-to-end 53
11 Example: Import foreign schema 55
12 Example: Join pushdown 56
13 Example: Aggregate pushdown 58
14 Example: ORDER BY pushdown 60
15 Example: LIMIT OFFSET pushdown 62
16 Example: Using a connection file 64
17 Identifying the version 65
18 Authentication plugin ‘caching_sha2_password’ error 66

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

1 MySQL Foreign Data Wrapper

The MySQL Foreign Data Wrapper (`mysql_fdw`) is a Postgres extension that lets you access data that resides on a MySQL database from EDB Postgres Advanced Server. It's a writable foreign data wrapper that you can use with Postgres functions and utilities or with other data that resides on a Postgres host.

You can install the MySQL Foreign Data Wrapper with an RPM package. You can download an installer from the EDB website.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2 Release notes

The MySQL Foreign Data Wrapper documentation describes the latest version of MySQL Foreign Data Wrapper 5, including minor releases and patches. The release notes provide information on what was new in each release. For new functionality introduced in a minor or patch release, the content also includes indicators about the release that introduced the feature.

Version	Release Date
2.9.3	13 Oct 2025
2.9.2	18 Jul 2024
2.9.1	20 Jul 2023
2.9.0	06 Jan 2023
2.8.0	26 May 2022
2.7.0	02 Dec 2021
2.6.0	18 May 2021
2.5.5	23 Nov 2020
2.5.3	10 Dec 2019
2.5.1	28 Nov 2018

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.1 MySQL Foreign Data Wrapper 2.9.3 release notes

> **Deprecation**
>
> Removed support for version 12 of EDB Postgres Advanced Server, EDB Postgres Extended Server and PostgreSQL.

Released: 13 Oct 2025

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.9.3 include:

Type	Description
Enhancement	Added support for PostgreSQL, EDB Postgres Advanced Server*[, and EDB Postgres Extended Server]* 18.
Bug fix	Improved the handling of `NULL` values when executing parameterized queries or execution-timed parameters to prevent unpredictable behavior due to dangling pointers.
Bug fix	Changed behavior where planner internal parameters were pushed to the remote server, which could cause the server to crash. Internal parameters are no longer pushed to the remote server.

[*] Support for EDB Postgres Advanced Server and EDB Postgres Extended Server 18 was added on 25 Nov 2025.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.2 MySQL Foreign Data Wrapper 2.9.2 release notes

Released: 18 Jul 2024

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.9.2 include:

Type	Description
Enhancement	Added support for PostgreSQL 17 and upcoming EDB Postgres Advanced Server 17.
Enhancement	Added `MYSQL_READ_DEFAULT_FILE` option to read the connection details from the default file.
Enhancement	Mapped TIMESTAMP to TIMESTAMPTZ in IMPORT SCHEMA processing.
Bug fix	Fixed an issue for server crash in left join with the whole row reference.
Bug fix	Fixed an issue for MySQL connection by setting timezone to UTC.
Bug fix	Fixed an issue related to pushdown of `ScalarArrayOpExpr` (ANY and ALL).
Bug fix	Fixed an issue while de-parsing the multi-dimensional array in `ScalarArrayOpExpr` (ANY and ALL).
Deprecation	Removed support for EDB Postgres Advanced Server 11 and PostgreSQL 11.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.3 MySQL Foreign Data Wrapper 2.9.1 release notes

Released: 20 Jul 2023

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.9.1 include:

Type	Description
Enhancement	Added support for PostgreSQL 16.
Enhancement	Added operators and functions entries in the default `mysql_fdw_pushdown.config` file.
Bug	Fixed the incorrect quoting when generating EXPLAIN plan of the query.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.4 MySQL Foreign Data Wrapper 2.9.0 release notes

Released: 06 Jan 2023

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.9.0 include:

Type	Description
Feature	Add a configuration file approach to control the pushdowns. We can now provide the list of operators and functions to be pushed down to the remote server safely.
Feature	Add support for the `import_generated` option in `IMPORT FOREIGN SCHEMA` to include columns generated from expressions in the definitions of foreign tables imported from a foreign server.
Feature	Add support for the `TRUNCATE` command to truncate foreign tables.
Feature	Add support for `ON CONFLICT DO NOTHING` in `INSERT`. It is mapping to `INSERT IGNORE`.
Feature	Add support for `IS [ NOT ] DISTINCT FROM` operator.
Bug Fix	Fix an oversight in assessing the pushdown `ORDER BY` clause. Don't push when underneath `query_pathkeys` is not safe to push down.
Bug Fix	Fix unstable ordering in the `SQL/select` test.
Bug Fix	Push down the `LIMIT n` clause when `OFFSET` is `NULL`.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.5 MySQL Foreign Data Wrapper 2.8.0 release notes

Released: 26 May 2022

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.8.0 include:

Type	Description
Feature	Push down ORDER BY to remote MySQL servers: When possible, consider a path that adds the ORDER BY clause to the remote SQL to get an ordered result set from the foreign server. This helps with an efficient merge join.
Feature	Push down LIMIT/OFFSET to remote MySQL servers: Wherever applicable, perform LIMIT and OFFSET operations on the remote server. This reduces network traffic between local PostgreSQL and remote MySQL servers.
Enhancement	Add character_set option. Users can now set it to an appropriate value per MySQL database.
Enhancement	Add sql_mode option. MySQL server can operate in different SQL modes, this option allows it to match that up.
Bug Fix	Introduce import_enum_as_text IMPORT FOREIGN SCHEMA option. When true, mysql_fdw implicitly maps the ENUM column to a TEXT type to import the table successfully without any failure.
Bug Fix	Force correct type modifiers while converting MySQL value to PostgreSQL.
Bug Fix	Fix text column data truncation when reading very long data.
Bug Fix	Do not push aggregates with VARIADIC array as MySQL does not support them.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.6 MySQL Foreign Data Wrapper 2.7.0 release notes

Released: 02 Dec 2021

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.7.0 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 14.
Enhancement	MySQL Foreign Data Wrapper now supports min, max, sum, avg, and count aggregate function pushdown. You can now push aggregates to the remote MySQL server instead of fetching all of the rows and aggregating them locally. This improves performance for the cases where aggregates can be pushed down. Aggregate filters and orders are not pushed down as MySQL does not support them.
Bug fix	Function expression deparsing implicit/explicit coercion sending wrong query to the MySQL server.
Bug fix	Import foreign schema failure due to SET type.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.7 MySQL Foreign Data Wrapper 2.6.0 release notes

Released: 18 May 2021

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.6.0 include:

Type	Description
Feature	Support for push down joins to the remote MySQL servers.
Enhancement	Support for EDB Postgres Advanced Server 13.
Enhancement	Support for Ubuntu 20.04 LTS platform.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.8 MySQL Foreign Data Wrapper 2.5.5 release notes

Released: 23 Nov 2020

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.5.5 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 13.
Enhancement	Support for Ubuntu 20.04 LTS platform.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.9 MySQL Foreign Data Wrapper 2.5.3 release notes

Released: 10 Dec 2019

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.5.3 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 12.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

2.10 MySQL Foreign Data Wrapper 2.5.1 release notes

Released: 28 Nov 2018

Enhancements, bug fixes, and other changes in MySQL Foreign Data Wrapper 2.5.1 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 11.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

3 Supported platforms

The MySQL Foreign Data Wrapper is supported on the same Linux platforms as EDB Postgres Advanced Server. To determine the platform support for the MySQL Foreign Data Wrapper, you can either refer to the platform support for EDB Postgres Advanced Server on the Platform Compatibility page on the EDB website or refer to Installing MySQL Foreign Data Wrapper.

Supported database versions

This table lists the latest MySQL Foreign Data Wrapper versions and their supported corresponding EDB Postgres Advanced Server (EPAS), EDB Postgres Extended Server (PGE), and PostgreSQL (PG) versions.

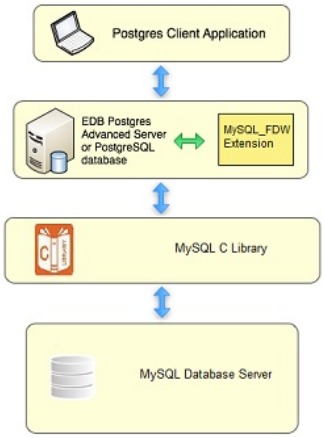
MySQL Foreign Data Wrapper	EPAS, PGE and PG 18	EPAS, PGE and PG 17	EPAS, PGE and PG 16	EPAS, PGE and PG 15	EPAS, PGE and PG 14	EPAS, PGE and PG 13
2.9.3	Y	Y	Y	Y	Y	Y
2.9.2	N	Y	Y	Y	Y	Y
2.9.1	N	N	Y	Y	Y	Y
2.9.0	N	N	N	Y	Y	Y
2.8.0	N	N	N	Y	Y	Y
2.7.0	N	N	N	N	Y	Y
2.6.1	N	N	N	N	Y	Y
2.6.0	N	N	N	N	N	Y
2.5.5	N	N	N	N	N	Y
2.5.4	N	N	N	N	N	Y

> **Note**
>
> While the source code for EDB's MySQL Foreign Data Wrapper could be used with EDB Postgres Extended Server instances, official support and dedicated packages (for easier installation and management) became available starting with version 2.9.2 and later.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

4 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.



Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

5 Key features

These are the key features of the MySQL Foreign Data Wrapper.

Writable foreign data wrapper

MySQL Foreign Data Wrapper provides the write capability. You 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
- Data type mappings

Connection pooling

MySQL Foreign Data Wrapper 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 pushdown

MySQL Foreign Data Wrapper allows the pushdown of a `WHERE` clause to the foreign server for execution. This feature optimizes remote queries to reduce the number of rows transferred from foreign servers.

Column pushdown

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

Join pushdown

MySQL Foreign Data Wrapper supports join pushdown. It pushes the joins between the foreign tables of the same remote MySQL server to that remote MySQL server, enhancing the performance.

> **Note**
>
> - Currently, joins involving only relational and arithmetic operators in join clauses are pushed down to avoid any potential join failure.
> - Only the INNER and LEFT/RIGHT OUTER joins are supported.

See also:

- Example: Join pushdown
- Blog: Join Pushdown - covers performance improvements and partition-wise join pushdowns

Aggregate pushdown

MySQL Foreign Data Wrapper supports aggregate pushdown for min, max, sum, avg, and count aggregate functions, allowing you to push aggregates to the remote MySQL server instead of fetching all of the rows and aggregating them locally. Aggregate filters and aggregate orders aren't pushed down as MySQL doesn't support them.

See also:

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

For more information, see Example: Aggregate pushdown Also, see Blog: Aggregate Pushdown - covers performance improvements, using join and aggregate pushdowns together, and pushing down aggregates to the partition table

ORDER BY pushdown

MySQL Foreign Data Wrapper supports ORDER BY pushdown. If possible, push ORDER BY clause to the remote server so that we get the ordered result set from the foreign server itself. It might help us to have an efficient merge join. NULLs behavior is opposite on the MySQL server. Thus to get an equivalent result, we add the "expression IS NULL" clause at the beginning of each of the ORDER BY expressions.

For more information, see Example: ORDER BY pushdown

LIMIT OFFSET pushdown

MySQL Foreign Data Wrapper supports limit offset push-down. Wherever possible, perform `LIMIT` and `OFFSET` operations on the remote server. This reduces network traffic between local PostgreSQL and remote MySQL servers. `ALL/NULL` options are not supported on the MySQL server, and thus they are not pushed down. Also, `OFFSET` without `LIMIT` is not supported on the MySQL server hence queries having that construct are not pushed.

For more information, see Example: LIMIT OFFSET pushdown

Configuration file to restrict pushdowns

MySQL 2.9.0 and later provides the `mysql_fdw_pushdown.config` configuration file to restrict the pushdowns. You can define the list of functions and operators in this file that can pushdown to the remote server. You can easily add or modify the list as per the requirements.

This file lists the objects as aggregates, functions, and operators allowed to push down to the remote server. Each entry should be on a single line. Each entry must have two columns:

- Object type that can be ROUTINE (functions, aggregates, and procedures) or OPERATOR
- The second column is the schema-qualified object names with their arguments

The exact form of the second column can be formatted using the following query:

For ROUTINE:

```
SELECT pronamespace::regnamespace || '.' || oid::regprocedure FROM pg_proc
WHERE proname = '<routine_name>'
```

For OPERATOR:

```
SELECT oprnamespace::regnamespace || '.' || oid::regoperator FROM
pg_operator
WHERE oprname = '<operator_name>'
```

Example of `mysql_fdw_pushdown.config` file:

```
ROUTINE pg_catalog.sum(bigint)
ROUTINE pg_catalog.sum(smallint)
ROUTINE pg_catalog.to_number(text)
ROUTINE pg_catalog.to_number(text,text)
OPERATOR pg_catalog.=(integer,integer)
OPERATOR pg_catalog.=(text,text)
OPERATOR pg_catalog.=(smallint,integer)
OPERATOR pg_catalog.=(bigint,integer)
OPERATOR pg_catalog.=(numeric,numeric)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Using a connection file

MySQL Foreign Data Wrapper 2.9.2 and later provides the `MYSQL_DEFAULT_FILE` option to read the connection details from the default file. You can provide the filename using this option with connection details like username, password, and so on. Set the `MYSQL_DEFAULT_FILE` option at a server level to the default connection filename. If this option is set to a wrong filename or the file contents aren't in the required format, then an error is raised.

If both `MYSQL_DEFAULT_FILE` and the other connection details are provided in the `CREATE SERVER/USER MAPPING` command, then the `MYSQL_DEFAULT_FILE` option takes precedence.

For more information, see Example: Using a connection file .

IS [NOT] DISTINCT FROM operator

MySQL Foreign Data Wrapper 2.9.0 and later supports the `IS [NOT] DISTINCT FROM` operator. MySQL uses the `<=>` operator corresponding to the `IS NOT DISTINCT FROM` operator and the `NOT <=>` operator corresponding to the `IS DISTINCT FROM` operator.

Prepared Statement

MySQL Foreign Data Wrapper supports Prepared Statement. The select queries use 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 to EDB Postgres Advanced Server from the MySQL server. The new foreign tables are created with the corresponding column types and same table name as the remote tables in the existing local schema. From version 2.9.0 and later, it supports the `import_generated` option to include columns generated from expressions in the definitions of foreign tables imported from a foreign server.

For more information, see Example: Import foreign schema for an example.

Automated cleanup

MySQL Foreign Data Wrapper allows the cleanup of foreign tables in a single operation using the `DROP EXTENSION` command. This feature is useful when a foreign table is set for a temporary purpose. The syntax is:

```
DROP EXTENSION mysql_fdw CASCADE;
```

For more information, see DROP EXTENSION.

Truncate table

From version 2.9.0 and later, MySQL Foreign Data Wrapper supports the `TRUNCATE TABLE` command on the foreign tables. However, the `CASCADE` option isn't supported with the `TRUNCATE TABLE` command.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6 Installing MySQL Foreign Data Wrapper on Linux

Select a link to access the applicable installation instructions:

Linux x86-64 (amd64)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9, RHEL 8
- Oracle Linux (OL) 9, Oracle Linux (OL) 8
- Rocky Linux 9, Rocky Linux 8
- AlmaLinux 9, AlmaLinux 8

SUSE Linux Enterprise (SLES)

- SLES 15

Debian and derivatives

- Ubuntu 24.04, Ubuntu 22.04
- Debian 12, Debian 11

Linux IBM Power (ppc64le)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9, RHEL 8

SUSE Linux Enterprise (SLES)

- SLES 15

Linux AArch64 (ARM64)

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- Oracle Linux (OL) 9

Debian and derivatives

- Debian 12

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1 Installing MySQL Foreign Data Wrapper on Linux x86 (amd64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- RHEL 8
- Oracle Linux (OL) 9
- Oracle Linux (OL) 8
- Rocky Linux 9
- Rocky Linux 8
- AlmaLinux 9
- AlmaLinux 8

SUSE Linux Enterprise (SLES)

- SLES 15

Debian and derivatives

- Ubuntu 24.04
- Ubuntu 22.04
- Debian 12
- Debian 11

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.1 Installing MySQL Foreign Data Wrapper on RHEL 9 or OL 9 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

  ```
  sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.2 Installing MySQL Foreign Data Wrapper on RHEL 8 or OL 8 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

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

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.3 Installing MySQL Foreign Data Wrapper on AlmaLinux 9 or Rocky Linux 9 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

  ```
  sudo dnf -y install epel-release
  ```
- Enable additional repositories to resolve dependencies:

  ```
  sudo dnf config-manager --set-enabled crb
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.4 Installing MySQL Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

  ```
  sudo dnf -y install epel-release
  ```
- Enable additional repositories to resolve dependencies:

  ```
  sudo dnf config-manager --set-enabled powertools
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.5 Installing MySQL Foreign Data Wrapper on SLES 15 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  zypper lr -E | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the MySQL community repository:

  ```
  sudo wget https://dev.mysql.com/get/mysql80-community-release-sles12-5.noarch.rpm
  rpm --import /etc/RPM-GPG-KEY-mysql-2022
  ```
- Enable the MySQL8 repository and disable the MySQL 5 repository:

  ```
  sudo zypper modifyrepo -e mysql80-community
  sudo zypper modifyrepo -d mysql57-community
  ```
- Activate the required SUSE module:

  ```
  sudo SUSEConnect -p PackageHub/15.7/x86_64
  ```
- Refresh the metadata:

  ```
  sudo zypper refresh
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo zypper -n install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo zypper -n install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo zypper -n install mysql_fdw_<X>
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.6 Installing MySQL Foreign Data Wrapper on Ubuntu 24.04 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
# For EDB Postgres Advanced Server:
sudo apt-get -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo apt-get -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo apt-get -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.7 Installing MySQL Foreign Data Wrapper on Ubuntu 22.04 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.

Install the package

```
# For EDB Postgres Advanced Server:
sudo apt-get -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo apt-get -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo apt-get -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.8 Installing MySQL Foreign Data Wrapper on Debian 12 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Address other prerequisites:

```
# Download the GPG key to your APT keyring directly using the apt-key utility:
sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29

# Install and configure the MySQL repo:
sudo echo "deb http://repo.mysql.com/apt/debian/bullseye mysql-8.0" | sudo tee  /etc/apt/sources.list.d/mysql.list

# Get the most up-to-date package information from the MySQL APT repository:
sudo apt-get update
```

- Address other prerequisites:

```
# Download the GPG key to your APT keyring directly using the apt-key utility:
sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29

# Install and configure the MySQL repo:
sudo echo "deb http://repo.mysql.com/apt/debian/bullseye mysql-8.0" | sudo tee  /etc/apt/sources.list.d/mysql.list

# Get the most up-to-date package information from the MySQL APT repository:
sudo apt-get update
```

Install the package

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

```
# For EDB Postgres Advanced Server
sudo apt-get -y install edb-as15-mysql8-fdw

# For EDB Postgres Extended Server:
sudo apt-get -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo apt-get -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.1.9 Installing MySQL Foreign Data Wrapper on Debian 11 x86_64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Address other prerequisites:

  ```
  # Download the GPG key to your APT keyring directly using the apt-key utility:
  sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29

  # Install and configure the MySQL repo:
  sudo echo "deb http://repo.mysql.com/apt/debian/bullseye mysql-8.0" | sudo tee  /etc/apt/sources.list.d/mysql.list

  # Get the most up-to-date package information from the MySQL APT repository:
  sudo apt-get update
  ```

Install the package

```
# For EDB Postgres Advanced Server
sudo apt-get -y install edb-as15-mysql8-fdw

# For EDB Postgres Extended Server:
sudo apt-get -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo apt-get -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.2 Installing MySQL Foreign Data Wrapper on Linux AArch64 (ARM64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- RHEL 9
- Oracle Linux (OL) 9

Debian and derivatives

- Debian 12

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.2.1 Installing MySQL Foreign Data Wrapper on RHEL 9 or OL 9 arm64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

  ```
  sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.2.2 Installing MySQL Foreign Data Wrapper on Debian 12 arm64

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  apt-cache search enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Address other prerequisites:

```
# Download the GPG key to your APT keyring directly using the apt-key utility:
sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29

# Install and configure the MySQL repo:
sudo echo "deb http://repo.mysql.com/apt/debian/bullseye mysql-8.0" | sudo tee  /etc/apt/sources.list.d/mysql.list

# Get the most up-to-date package information from the MySQL APT repository:
sudo apt-get update
```

- Address other prerequisites:

```
# Download the GPG key to your APT keyring directly using the apt-key utility:
sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29

# Install and configure the MySQL repo:
sudo echo "deb http://repo.mysql.com/apt/debian/bullseye mysql-8.0" | sudo tee  /etc/apt/sources.list.d/mysql.list

# Get the most up-to-date package information from the MySQL APT repository:
sudo apt-get update
```

Install the package

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

```
# For EDB Postgres Advanced Server
sudo apt-get -y install edb-as15-mysql8-fdw

# For EDB Postgres Extended Server:
sudo apt-get -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo apt-get -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.3 Installing MySQL Foreign Data Wrapper on Linux IBM Power (ppc64le)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL)

- RHEL 9
- RHEL 8

SUSE Linux Enterprise (SLES)

- SLES 15

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.3.1 Installing MySQL Foreign Data Wrapper on RHEL 9 ppc64le

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

  ```
  sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
  ```
- Refresh the cache:

  ```
  sudo dnf makecache
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.3.2 Installing MySQL Foreign Data Wrapper on RHEL 8 ppc64le

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  dnf repolist | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the EPEL repository:

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

  ```
  sudo dnf makecache
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo dnf -y install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo dnf -y install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo dnf -y install mysql_fdw_<X>
```

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

6.3.3 Installing MySQL Foreign Data Wrapper on SLES 15 ppc64le

Prerequisites

Before you begin the installation process:

- Install Postgres on the same host. See:
  - Installing EDB Postgres Advanced Server
  - Installing PostgreSQL
- Set up the EDB repository.

  Setting up the repository is a one-time task. If you already set up your repository, you don't need to perform this step.

  To determine if your repository exists, enter:

  ```
  zypper lr -E | grep enterprisedb
  ```

  If no output is generated, the repository is installed.

  To set up the EDB repository:

  1. Go to EDB repositories.
  2. Select the button that provides access to the EDB repository.
  3. Select the platform and software that you want to download.
  4. Follow the instructions for setting up the EDB repository.
- Install the MySQL community repository:

  ```
  sudo wget https://dev.mysql.com/get/mysql80-community-release-sles12-5.noarch.rpm
  rpm --import /etc/RPM-GPG-KEY-mysql-2022
  ```
- Enable the MySQL8 repository and disable the MySQL 5 repository:

  ```
  sudo zypper modifyrepo -e mysql80-community
  sudo zypper modifyrepo -d mysql57-community
  ```
- Activate the required SUSE module:

  ```
  sudo SUSEConnect -p PackageHub/15.7/ppc64le
  ```
- Refresh the metadata:

  ```
  sudo zypper refresh
  ```

Install the package

```
# For EDB Postgres Advanced Server:
sudo zypper -n install edb-as<X>-mysql8_fdw

# For EDB Postgres Extended Server:
sudo zypper -n install edb-postgresextended<X>-mysql8_fdw

# For PostgreSQL:
sudo zypper -n install mysql_fdw_<X>
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Where `<X>` is the version of your Postgres and `8` is the version of MySQL to be installed.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

7 Initial configuration

Before using the MySQL Foreign Data Wrapper:

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 theIMPORT 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're 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 returning 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 the PostgreSQL documentation.

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 in the database.

`FOREIGN_DATA_WRAPPER`

Include the `FOREIGN_DATA_WRAPPER` clause to specify for the server to use the `mysql_fdw` foreign data wrapper when connecting to the cluster.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

`OPTIONS`

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

Option	Description
character_set	The character set to use for MySQL connection. The default value is `auto` which means autodetect based on the operating system setting. Before the introduction of the `character_set` option, the character set was set similar to the PostgreSQL database encoding. To get this older behavior set the `character_set` to special value `PGDatabaseEncoding`.
fetch_size	This option specifies the number of rows mysql_fdw should get in each fetch operation. It can be specified for a foreign table or a foreign server. The option specified on a table overrides an option specified for the server. The default value is `100`.
host	The address or hostname of the MySQL server. The default value is `127.0.0.1`.
init_command	The SQL statement to execute when connecting to the MySQL server.
port	The port number of the MySQL Server. The default is `3306`.
reconnect	Enable or disable automatic reconnection to the MySQL server if the existing connection is found to have been lost. The default value is `false`.
secure_auth	Use to enable or disable secure authentication. The default is `true`.
sql_mode	Set MySQL `sql_mode` for established connection. Default is `ANSI_QUOTES`.
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.
use_remote_estimate	Include `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 the PostgreSQL documentation.

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 to associate 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`

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

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

`username` is the name of the user on the MySQL server.

`password` is 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's associated with a server named `mysql_server`. When connecting to the MySQL server, the server authenticates as `foo` and provides a password of `bar`.

For detailed information about the `CREATE USER MAPPING` command, see the PostgreSQL documentation.

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 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' [, ... ] )
]
```

`column_constraint` is:

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

`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 resides.

`IF NOT EXISTS`

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Include the `IF NOT EXISTS` clause to instruct the server to not return an error if a table with the same name already exists. If a table with the same name exists, the server issues a notice.

`column_name`

Specifies the name of a column in the new table. Each column must 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 isn't available, the Postgres server attempts to cast the data type to a type compatible with the MySQL server. If the server can't identify a compatible data type, it returns 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 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 generates a constraint name.

`NOT NULL`

Include the `NOT NULL` keywords to indicate that the column isn't 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 references that column's value only, while an expression appearing in a table constraint can reference multiple columns.

A `CHECK` expression can't contain subqueries or refer to variables other than columns of the current row.

Include the `NO INHERIT` keywords to specify that a constraint must 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 in. 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 allows you to query a table that resides on a MySQL file system, include the `SERVER` clause and specify `server_name` for 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.
fetch_size	This option specifies the number of rows mysql_fdw should get in each fetch operation. It can be specified for a foreign table or a foreign server. The option specified on a table overrides an option specified for the server. The default value is `100`.
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.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Example

To use data that's stored on a 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);
```

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 the PostgreSQL documentation.

> **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 converts 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
LONGTEXT	TEXT
SMALLINT	SMALLINT/INT2
TIMESTAMP	TIMESTAMP
VARCHAR()	VARCHAR()/CHARCTER VARYING()

> **Note**
>
> For `ENUM` data type:
>
> MySQL accepts an `enum` value in string form. You must create exactly the same `enum` listing on EDB Postgres Advanced Server as is present on the MySQL server. Any sort of inconsistency causes an error while fetching rows with values not known on the local server.
>
> Also, when the given `enum` value isn't present at the MySQL side but is present at the EDB Postgres Advanced Server side, an empty string (`''`) is inserted as a value at the MySQL side for the `enum` column. To select from a table having the `enum` value as `''` , create an `enum` type at the Postgres side with all valid values and `''` .

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

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 you want to creat the imported foreign tables.

`OPTIONS`

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

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`.
import_enum_as_text	Can be used to map MySQL `ENUM` type to `TEXT` type in the definitions of foreign tables, otherwise emit a warning for type to be created. The default is `false`.

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');
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

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 the PostgreSQL documentation.

DROP EXTENSION

Use the `DROP EXTENSION` command to remove an extension. To invoke the command, use your client of choice (for example, psql) to connect to the Postgres database from which you're dropping the MySQL server, and run the command:

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT
];
```

Parameters

`IF EXISTS`

Include the `IF EXISTS` clause to instruct the server to issue a notice instead of returning an error if an extension with the specified name doesn't exist.

`name`

Optionally, specify the name of the installed extension.

`CASCADE`

Drop objects that depend on the extension. It drops all the other dependent objects too.

`RESTRICT`

Don't allow to drop extension if any objects, other than its member objects and extensions listed in the same `DROP` command, depend on it.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

Example

The following command removes the extension from the existing database:

```
DROP EXTENSION mysql_fdw;
```

For more information about using the foreign data wrapper `DROP EXTENSION` command, see the PostgreSQL documentation.

DROP SERVER

Use the `DROP SERVER` command to remove a connection to a foreign server. The syntax is:

```
DROP SERVER [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT
]
```

The role that drops the server is the owner of the server. Use the `ALTER SERVER` command to reassign ownership of a foreign server. To drop a foreign server, you must have `USAGE` privilege on the foreign data wrapper specified in the `DROP SERVER` command.

Parameters

`IF EXISTS`

Include the `IF EXISTS` clause to instruct the server to issue a notice instead of returning an error if a server with the specified name doesn't exists.

`name`

Optionally, specify the name of the installed server.

`CASCADE`

Drop objects that depend on the server. Drop all the other dependent objects too.

`RESTRICT`

Don't allow to drop the server if any objects depend on it.

Example

The following command removes a foreign server named `mysql_server`:

```
DROP SERVER mysql_server;
```

For more information about using the `DROP SERVER` command, see the PostgreSQL documentation.

Use the `DROP USER MAPPING` command to remove a mapping that associates a Postgres role with a foreign server. You must be the owner of the foreign server to remove a user mapping for that server.

```
DROP USER MAPPING [ IF EXISTS ] FOR { user_name | USER | CURRENT_USER | PUBLIC } SERVER
server_name;
```

Parameters

`IF EXISTS`

Include the `IF EXISTS` clause to instruct the server to issue a notice instead of throwing an error if the user mapping doesn't exist.

`user_name`

Specify the user name of the mapping.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

`server_name`

Specify the name of the server that defines a connection to the MySQL cluster.

Example

The following command drops a user mapping for a role named `enterprisedb`. The mapping is associated with a server named `mysql_server`.

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

For detailed information about the `DROP USER MAPPING` command, see the PostgreSQL documentation.

DROP FOREIGN TABLE

A foreign table is a pointer to a table that resides on the MySQL host. Use the `DROP FOREIGN TABLE` command to remove a foreign table. Only the owner of the foreign table can drop it.

```
DROP FOREIGN TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT
]
```

Parameters

`IF EXISTS`

Include the `IF EXISTS` clause to instruct the server to issue a notice instead of returning an error if the foreign table with the specified name doesn't exist.

`name`

Specify the name of the foreign table.

`CASCADE`

Drop objects that depend on the foreign table. Drop all the other dependent objects too.

`RESTRICT`

Don't allow to drop foreign table if any objects depend on it.

Example

```
DROP FOREIGN TABLE warehouse;
```

For more information about using the `DROP FOREIGN TABLE` command, see the PostgreSQL documentation.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

8 Upgrading

If you have an existing installation of MySQL Foreign Data Wrapper that you installed using the EDB repository, you can update your repository configuration file and then upgrade MySQL Foreign Data Wrapper to a more recent product version.

To perform the process, open a terminal window and enter the commands that apply to the operating system and package manager used for the installation:

To update your repository configuration file:

```
sudo <package-manager> upgrade edb-repo
```

Where `<package-manager>` is the package manager used with your operating system:

Package manager	Operating system
dnf	RHEL 8/9 and derivatives
zypper	SLES
apt-get	Debian and Ubuntu

To upgrade to the latest product version, enter one of the following commands:

Operating system	Upgrade command
RHEL 8/9 and derivatives	`sudo dnf -y upgrade edb-as<xx>-mysql<y>_fdw* mysql-community-devel`
SLES	`sudo zypper -y upgrade edb-as<xx>-mysql<y>_fdw* mysql-community-devel`
Debian and Ubuntu	`sudo apt-get upgrade edb-as<xx>-mysql<y>-fdw`

Where

- `<package-manager>` is the package manager used with your operating system.
- `<xx>` is the EDB Postgres Advanced Server version number.
- `<y>` is the version of MySQL to be upgraded.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

9 Uninstalling

You can use the `remove` command to uninstall MySQL Foreign Data Wrapper packages. To uninstall, open a terminal window, assume superuser privileges, and enter the command that applies to the operating system and package manager used for the installation:

- On RHEL or Rocky Linux or AlmaLinux 8:

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

  Where `<xx>` is the EDB Postgres Advanced Server version.

- On SLES:

  ```
  zypper remove edb-as<xx>-mysql<x>_fdw
  ```

  Where:

  - `<xx>` is the EDB Postgres Advanced Server version.
  - `<x>` is the supported release version number of MySQL.

- On Debian or Ubuntu

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

  Where `<xx>` is the EDB Postgres Advanced Server version.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

10 Example: End-to-end

Access data from EDB Postgres Advanced Server and connect to psql. Once you're connected to psql, follow these 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 enterprisedb
    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)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

```
                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'))

-- drop foreign table
DROP FOREIGN TABLE warehouse;

-- drop user mapping
DROP USER MAPPING FOR enterprisedb SERVER
mysql_server;

-- drop
server
DROP SERVER
mysql_server;
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

11 Example: Import foreign schema

Access data from EDB Postgres Advanced Server and connect to psql. Once you're connected to psql, follow these 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
```

See IMPORT FOREIGN SCHEMA for more information.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

12 Example: Join pushdown

This example shows join pushdown between two foreign tables: `warehouse` and `sales_records`.

Table on MySQL server:

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

CREATE TABLE sales_records
(
warehouse_id      INT PRIMARY KEY,
qty           INT
);
```

Table on Postgres server:

```
CREATE EXTENSION mysql_fdw;
CREATE SERVER mysql_server FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host '127.0.0.1', port
'3306');
CREATE USER MAPPING FOR public SERVER mysql_server OPTIONS (username 'edb', password
'edb');

CREATE FOREIGN TABLE warehouse
(
warehouse_id      INT,
warehouse_name    TEXT,
warehouse_created TIMESTAMP
)
SERVER mysql_server OPTIONS (dbname 'edb', table_name 'warehouse');
INSERT INTO warehouse values (1, 'UPS',
current_date);
INSERT INTO warehouse values (2, 'TV',
current_date);
INSERT INTO warehouse values (3, 'Table',
current_date);

CREATE FOREIGN TABLE sales_records
(
warehouse_id      INT,
qty           INT
)
SERVER mysql_server OPTIONS (dbname 'edb', table_name 'sales_records');
INSERT INTO sales_records values (1,
100);
INSERT INTO sales_records values (2,
75);
INSERT INTO sales_records values (3,
200);
```

The output:

```
--inner join
edb=# EXPLAIN VERBOSE SELECT t1.warehouse_name, t2.qty FROM warehouse t1 INNER JOIN sales_records t2 ON (t1.warehouse_id =
t2.warehouse_id);
```

```
output
                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
--------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=36)
   Output: t1.warehouse_name, t2.qty
   Relations: (edb.warehouse t1) INNER JOIN (edb.sales_records t2)
   Local server startup cost: 10
   Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`warehouse` r1 INNER JOIN `edb`.`sales_records` r2 ON
(((r1.`warehouse_id` = r2.`warehouse_id`))))
(5 rows)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

```
--left join
edb=# EXPLAIN VERBOSE SELECT t1.warehouse_name, t2.qty FROM warehouse t1 LEFT JOIN sales_records t2 ON (t1.warehouse_id =
t2.warehouse_id);
```

```
output
                                                      QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
-------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=36)
   Output: t1.warehouse_name, t2.qty
   Relations: (edb.warehouse t1) LEFT JOIN (edb.sales_records t2)
   Local server startup cost: 10
   Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`warehouse` r1 LEFT JOIN `edb`.`sales_records` r2 ON
(((r1.`warehouse_id` = r2.`warehouse_id`))))
(5 rows)
```

```
--right join
edb=# EXPLAIN VERBOSE SELECT t1.warehouse_name, t2.qty FROM warehouse t1 RIGHT JOIN sales_records t2 ON (t1.warehouse_id =
t2.warehouse_id);
```

```
output
                                                      QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
-------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=36)
   Output: t1.warehouse_name, t2.qty
   Relations: (edb.sales_records t2) LEFT JOIN (edb.warehouse t1)
   Local server startup cost: 10
   Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`sales_records` r2 LEFT JOIN `edb`.`warehouse` r1 ON
(((r1.`warehouse_id` = r2.`warehouse_id`))))
(5 rows)
```

```
--cross join
edb=# EXPLAIN VERBOSE SELECT t1.warehouse_name, t2.qty FROM warehouse t1 CROSS
JOIN
sales_records
t2;
```

```
output
                                        QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=1000000 width=36)
   Output: t1.warehouse_name, t2.qty
   Relations: (edb.warehouse t1) INNER JOIN (edb.sales_records t2)
   Local server startup cost: 10
   Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`warehouse` r1 INNER JOIN `edb`.`sales_records` r2 ON (TRUE))
(5 rows)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

13 Example: Aggregate pushdown

MySQL Foreign Data Wrapper supports pushdown for the following aggregate functions:

- AVG - Calculates the average of a set of values.
- COUNT - Counts rows in a specified table or view.
- MIN - Gets the minimum value in a set of values.
- MAX - Gets the maximum value in a set of values.
- SUM - Calculates the sum of values.

Table on MySQL server:

```
CREATE FOREIGN TABLE sales_records
(
warehouse_id      INT PRIMARY KEY,
qty           INT
);
```

Table on Postgres server:

```
CREATE EXTENSION mysql_fdw;
CREATE SERVER mysql_server FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host '127.0.0.1', port
'3306');
CREATE USER MAPPING FOR public SERVER mysql_server OPTIONS (username 'edb', password
'edb');

CREATE FOREIGN TABLE sales_records
(
warehouse_id      INT,
qty           INT
)
SERVER mysql_server OPTIONS (dbname 'edb', table_name 'sales_records');
INSERT INTO sales_records values (1,
100);
INSERT INTO sales_records values (2,
75);
INSERT INTO sales_records values (3,
200);
```

The output:

```
edb=# EXPLAIN VERBOSE SELECT avg(qty) FROM
sales_records;
```

output

```
                          QUERY PLAN
-------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=32)
   Output: (avg(qty))
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT avg(`qty`) FROM `edb`.`sales_records`
(5 rows)
```

```
edb=# EXPLAIN VERBOSE SELECT COUNT(qty) FROM
sales_records;
```

output

```
                           QUERY PLAN
-----------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=8)
   Output: (count(qty))
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT count(`qty`) FROM `edb`.`sales_records`
(5 rows)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

```
edb=# EXPLAIN VERBOSE SELECT MIN(qty) FROM
sales_records;
```

output

```
                         QUERY PLAN
-------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=4)
   Output: (min(qty))
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT min(`qty`) FROM `edb`.`sales_records`
(5 rows)
```

```
edb=# EXPLAIN VERBOSE SELECT MAX(qty) FROM
sales_records;
```

output

```
                         QUERY PLAN
-------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=4)
   Output: (max(qty))
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT max(`qty`) FROM `edb`.`sales_records`
(5 rows)
```

```
edb=# EXPLAIN VERBOSE SELECT SUM(qty) FROM
sales_records;
```

output

```
                         QUERY PLAN
-------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=8)
   Output: (sum(qty))
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT sum(`qty`) FROM `edb`.`sales_records`
(5 rows)
```

```
edb=# EXPLAIN VERBOSE SELECT SUM(qty) FROM sales_records GROUP BY warehouse_id HAVING SUM(qty) = 75;
```

output

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=200 width=12)
   Output: (sum(qty)), warehouse_id
   Relations: Aggregate on (edb.sales_records)
   Local server startup cost: 10
   Remote query: SELECT sum(`qty`), `warehouse_id` FROM `edb`.`sales_records` GROUP BY 2 HAVING ((sum(`qty`) = 75))
(5 rows)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

14 Example: ORDER BY pushdown

This example shows ORDER BY pushdown on foreign table: `sales_records`.

Table on MySQL server:

```
CREATE TABLE
sales_records(
warehouse_id      INT PRIMARY KEY,
qty           INT);
```

Table on Postgres server:

```
-- 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 public SERVER mysql_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE
sales_records(
warehouse_id      INT,
qty           INT)
SERVER mysql_server OPTIONS (dbname 'edb', table_name 'sales_records');

-- insert into
table
INSERT INTO sales_records values (1,
100);
INSERT INTO sales_records values (2,
75);
INSERT INTO sales_records values (3,
200);
```

The Output:

```
-- ORDER BY
ASC
edb=#EXPLAIN VERBOSE SELECT * FROM sales_records WHERE qty > 80 ORDER BY
warehouse_id;
```

output

```
                                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
---------------
 Foreign Scan on public.sales_records  (cost=10.00..1010.00 rows=1000 width=8)
   Output: warehouse_id, qty
   Local server startup cost: 10
   Remote query: SELECT `warehouse_id`, `qty` FROM `edb`.`sales_records` WHERE ((`qty` > 80)) ORDER BY `warehouse_id` IS NULL,
`warehouse_id` ASC
(4 rows)
```

```
-- ORDER BY DESC
edb=#EXPLAIN VERBOSE SELECT * FROM sales_records ORDER BY warehouse_id DESC;
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

output

```
                                                
```

15 Example: LIMIT OFFSET pushdown

This example shows LIMIT OFFSET pushdown on foreign table: `sales_records`.

Table on MySQL server:

```
CREATE TABLE
sales_records(
warehouse_id      INT PRIMARY KEY,
qty           INT);
```

Table on Postgres server:

```
-- 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 public SERVER mysql_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE
sales_records(
warehouse_id      INT,
qty           INT)
SERVER mysql_server OPTIONS (dbname 'edb', table_name 'sales_records');

-- insert into
table
INSERT INTO sales_records values (1,
100);
INSERT INTO sales_records values (2,
75);
INSERT INTO sales_records values (3,
200);
```

The output:

```
-- LIMIT only
edb@91975=#EXPLAIN VERBOSE SELECT * FROM sales_records WHERE qty > 80 ORDER BY warehouse_id LIMIT
5;
```

```
output
                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
----------------------
 Foreign Scan on public.sales_records  (cost=1.00..2.00 rows=1 width=8)
   Output: warehouse_id, qty
   Local server startup cost: 10
   Remote query: SELECT `warehouse_id`, `qty` FROM `edb`.`sales_records` WHERE ((`qty` > 80)) ORDER BY `warehouse_id` IS NULL,
`warehouse_id` ASC LIMIT 5
(4 rows)
```

```
-- LIMIT and
OFFSET
edb@91975=#EXPLAIN VERBOSE SELECT * FROM sales_records WHERE qty > 80 ORDER BY warehouse_id LIMIT 5 OFFSET
5;
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

output

```
                                                
```

16 Example: Using a connection file

This example shows how to use the connection file.

```
-- Prepare the default file with the connection
details:
cat /tmp/my.cnf
[client]
host=localhost
port=3306
user=edb
password=edb
secure_auth=true
```

```
-- create a server object with mysql_default_file
option
CREATE SERVER mysql_server FOREIGN DATA WRAPPER mysql_fdw
  OPTIONS (mysql_default_file
'/tmp/my.cnf');
```

```
-- Create a user mapping without providing username and password
options
CREATE USER MAPPING FOR public SERVER
mysql_server;
```

```
-- Create a foreign
table:
CREATE FOREIGN TABLE f_mysql_file_test(a int, b int) SERVER mysql_server OPTIONS (dbname 'mysql_fdw_regress', table_name
'mysql_test');
```

```
-- Query the data from the foreign table to check if the connection details are read from the default
file:
SELECT * FROM f_mysql_file_test ORDER BY 1, 2;
```

output

```
 a | b
---+---
 1 | 1
(1 row)
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

17 Identifying the 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>
```

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.

18 Authentication plugin ‘caching_sha2_password’ error

If you encounter the following 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 is:

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

> **Note**
>
> See the MySQL 8 documentation for more details on this error.

Copyright © 2009 - 2025 EnterpriseDB Corporation. All rights reserved.