MongoDB Foreign Data Wrapper

Version 5

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

Built at 2025-12-12T14:51:10

1 MongoDB Foreign Data Wrapper 3
2 Release notes 4
2.1 MongoDB Foreign Data Wrapper 5.5.3 release notes 5
2.2 MongoDB Foreign Data Wrapper 5.5.2 release notes 6
2.3 MongoDB Foreign Data Wrapper 5.5.1 release notes 7
2.4 MongoDB Foreign Data Wrapper 5.5.0 release notes 8
2.5 MongoDB Foreign Data Wrapper 5.4.0 release notes 9
2.6 MongoDB Foreign Data Wrapper 5.3.0 release notes 10
2.7 MongoDB Foreign Data Wrapper 5.2.9 release notes 11
2.8 MongoDB Foreign Data Wrapper 5.2.8 release notes 12
2.9 MongoDB Foreign Data Wrapper 5.2.6 release notes 13
2.10 MongoDB Foreign Data Wrapper 5.2.3 release notes 14
3 Supported platforms 15
4 Limitations 16
5 Architecture overview 17
6 Key features 18
7 Installing MongoDB Foreign Data Wrapper on Linux 22
7.1 Installing MongoDB Foreign Data Wrapper on Linux x86 (amd64) 24
7.1.1 Installing MongoDB Foreign Data Wrapper on RHEL 9 or OL 9 x86_64 25
7.1.2 Installing MongoDB Foreign Data Wrapper on RHEL 8 or OL 8 x86_64 26
7.1.3 Installing MongoDB Foreign Data Wrapper on AlmaLinux 9 or Rocky Linux 9 x86_64 27
7.1.4 Installing MongoDB Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64 29
7.1.5 Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64 31
7.1.6 Installing MongoDB Foreign Data Wrapper on Ubuntu 24.04 x86_64 33
7.1.7 Installing MongoDB Foreign Data Wrapper on Ubuntu 22.04 x86_64 34
7.1.8 Installing MongoDB Foreign Data Wrapper on Debian 12 x86_64 35
7.1.9 Installing MongoDB Foreign Data Wrapper on Debian 11 x86_64 36
7.2 Installing MongoDB Foreign Data Wrapper on Linux AArch64 (ARM64) 37
7.2.1 Installing MongoDB Foreign Data Wrapper on RHEL 9 or OL 9 arm64 38
7.2.2 Installing MongoDB Foreign Data Wrapper on Debian 12 arm64 39
7.3 Installing MongoDB Foreign Data Wrapper on Linux IBM Power (ppc64le) 40
7.3.1 Installing MongoDB Foreign Data Wrapper on RHEL 9 ppc64le 41
7.3.2 Installing MongoDB Foreign Data Wrapper on RHEL 8 ppc64le 43
7.3.3 Installing MongoDB Foreign Data Wrapper on SLES 15 ppc64le 45
8 Initial Configuration 47
9 Upgrading 57
10 Uninstalling 58
11 Example: End-to-end 59
12 Example: Join pushdown 62
13 Example: Aggregate function pushdown 64
14 Example: LIMIT OFFSET pushdown 67
15 Example: ORDER BY pushdown 69
16 Example: WHERE clause pushdown 71
17 Identifying the version 72
18 Example: Remote query logging 73

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

1 MongoDB Foreign Data Wrapper

The MongoDB Foreign Data Wrapper (`mongo_fdw`) is a Postgres extension that lets you access data that resides on a MongoDB 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 MongoDB 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 Mongo Foreign Data Wrapper documentation describes the latest version of Mongo 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 indicates the release that introduced the feature.

Version	Release date
5.5.3	13 Oct 2025
5.5.2	18 Jul 2024
5.5.1	20 Jul 2023
5.5.0	06 Jan 2023
5.4.0	26 May 2022
5.3.0	02 Dec 2021
5.2.9	24 Jun 2021
5.2.8	23 Nov 2020
5.2.6	27 Sep 2019
5.2.3	01 Nov 2018

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

2.1 MongoDB Foreign Data Wrapper 5.5.3 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 MongoDB Foreign Data Wrapper 5.5.3 include:

Type	Description
Enhancement	Added support for PostgreSQL, EDB Postgres Advanced Server*[, and EDB Postgres Extended Server]* 18.
Enhancement	Added the ability to build the `mongo_fdw` extension directly from PostgreSQL's `contrib` directory.
Feature	Added `log_remote_query` GUC to control logging of remote MongoDB queries.
Enhancement	Updated `mongoc-driver(1.30.2)` and `json-c(0.18-20240915)` libraries to support MongoDB 8.
Bug Fix	Fixed an issue where the FDW pushed down joins with full document or whole row references, which could lead to incorrect output. The FDW now avoids this specific pushdown to ensure data correctness.
Bug Fix	Fixed an issue where `ORDER BY` pushdown failed or produced incorrect results when sorting by columns from a table participating on the inner side of a join.
Bug Fix	Fixed incorrect appending of `WHERE` clause in `mongo_query_document` to prevent `invalid array detected` issues.
Bug Fix	Fixed an issue where incorrect variable names were generated within the MongoDB query pipeline when querying nested fields (sub-columns).

[*] 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 MongoDB Foreign Data Wrapper 5.5.2 release notes

Released: 18 Jul 2024

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.5.2 include:

Type	Description
Enhancement	Added support for PostgreSQL 17 and upcoming EDB Postgres Advanced Server 17.
Deprecation	Removed support for legacy drivers.
Deprecation	Removed support for EDB Postgres Advanced Server 11 and PostgreSQL 11.

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

2.3 MongoDB Foreign Data Wrapper 5.5.1 release notes

Released: 20 Jul 2023

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.5.0 include:

Type	Description
Enhancement	Added support for PostgreSQL 16.
Feature	Added `enable_join_pushdown` and `enable_aggregate_pushdown` GUCs to support JOIN and AGGREGATE push downs, respectively.
Feature	Added table and server level option to `enable_order_by_pushdown` GUC.
Bugfix	Fixed boolean expression handling in building the query document.

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

2.4 MongoDB Foreign Data Wrapper 5.5.0 release notes

Released: 06 Jan 2023

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.5.0 include:

Type	Description
Feature	When possible, we push the `ORDER BY` clause to the remote MongoDB server. This approach provides the ordered result set from the foreign server, which can help to have an efficient merge join.
Feature	When possible, perform `LIMIT` and `OFFSET` operations on the remote server. This reduces network traffic between local Postgres and remote MongoDB servers.
Enhancement	Improved the WHERE clause pushdown to remote MongoDB servers. Now supports recursive operator expressions, Boolean expressions, relabel types, and vars on both sides of an operator.
Bug Fix	For nested join queries, save the status of the `enable_aggregate_pushdown` option to access the aggregation path later.
Bug Fix	Fix server crash due to missing `Param` node handling.
Bug Fix	Fix typos in `autogen.sh` and `README.md` files.

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

2.5 MongoDB Foreign Data Wrapper 5.4.0 release notes

Released: 26 May 2022

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.4.0 include:

Type	Description
Feature	Push down aggregates to remote MongoDB servers: Push aggregates to the remote MongoDB server instead of fetching all of the rows and aggregating them locally. This gives a very good performance boost for the cases where aggregates can be pushed down.
Enhancement	Add a new server option use_remote_estimate to estimate the rows.
Bug Fix	Force type modifiers for NUMERIC type.
Bug Fix	Fix data type incompatibility between NUMERIC types.
Bug Fix	Fix server crash when updating the document with text array element.
Bug Fix	Fix server crash while updating/deleting with null value for _id column.

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

2.6 MongoDB Foreign Data Wrapper 5.3.0 release notes

Released: 02 Dec 2021

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.3.0 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 14.
Enhancement	Join pushdown: If a query has a join between two foreign tables from the same remote server, you can now push that join down to the remote server instead of fetching all the rows for both the tables and performing a join locally.
Bug fix	Improve API performance.
Bug fix	Need support for the whole-row reference.

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

2.7 MongoDB Foreign Data Wrapper 5.2.9 release notes

Released: 24 Jun 2021

New features, enhancements, bug fixes, and other changes in Mongo Foreign Data Wrapper 5.2.9 include:

Type	Description
Enhancement	Updated mongo-c-driver to 1.17.3.
Enhancement	Updated json-c to 0.15.
Enhancement	Updated LICENSE file.
Bug fix	Fixed crash with the queries involving LEFT JOIN LATERAL.
Bug fix	Restrict fetching PostgreSQL-specific system attributes from the remote relation to avoid a server crash.
Bug fix	Improved WHERE pushdown so that more conditions can be sent to the remote server.

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

2.8 MongoDB Foreign Data Wrapper 5.2.8 release notes

Released: 23 Nov 2020

New features, enhancements, bug fixes, and other changes in Mongo Foreign Data Wrapper 5.2.8 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 13.
Enhancement	Support for Ubuntu 20.04 LTS platform.
Enhancement	Updated LICENSE file.
Bug fix	Fixed crash with COPY FROM and/or foreign partition routing operations. The crash was caused by Mongo Foreign Data Wrapper not supporting routable foreign-table partitions and/or executing COPY FROM on foreign tables. Instead of crashing, Mongo Foreign Data Wrapper now throws an error.
Bug fix	Fixed issue where casting target list produces 'NULL'. Correct results are returned not only for have an explicit casts, but also for function calls or operators in the target list.
Bug fix	Fixed ReScanForeignScan API to make the parameterized query work correctly. Sub-select or correlated queries now use a parameterized plan.
Bug fix	Changed the server port option's type from int32 to int16 to resolve compilation warnings. Meta driver APIs expect port value in unsigned short type, which resulted in a compilation warning on some gcc versions.

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

2.9 MongoDB Foreign Data Wrapper 5.2.6 release notes

Released: 27 Sep 2019

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.2.6 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 12.

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

2.10 MongoDB Foreign Data Wrapper 5.2.3 release notes

Released: 01 Nov 2018

Enhancements, bug fixes, and other changes in MongoDB Foreign Data Wrapper 5.2.3 include:

Type	Description
Enhancement	Support for EDB Postgres Advanced Server 11.

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

3 Supported platforms

MongoDB Foreign Data Wrapper is supported on the same platforms as EDB Postgres Advanced Server. To determine the platform support for the MongoDB Foreign Data Wrapper, see the Platform Compatibility page on the EDB website or Installing MongoDB Foreign Data Wrapper.

Supported database versions

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

MongoDB Foreign Data Wrapper	EPAS, PGE, 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
5.5.3	Y	Y	Y	Y	Y	Y
5.5.2	N	Y	Y	Y	Y	Y
5.5.1	N	N	Y	Y	Y	Y
5.5.0	N	N	N	Y	Y	Y
5.4.0	N	N	N	Y	Y	Y
5.3.0	N	N	N	N	Y	Y
5.2.10	N	N	N	N	N	Y
5.2.9	N	N	N	N	N	Y
5.2.8	N	N	N	N	N	Y
5.2.7	N	N	N	N	N	Y

> **Note**
>
> While the source code for EDB's MongoDB 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 5.5.2 and continues for later versions.

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

4 Limitations

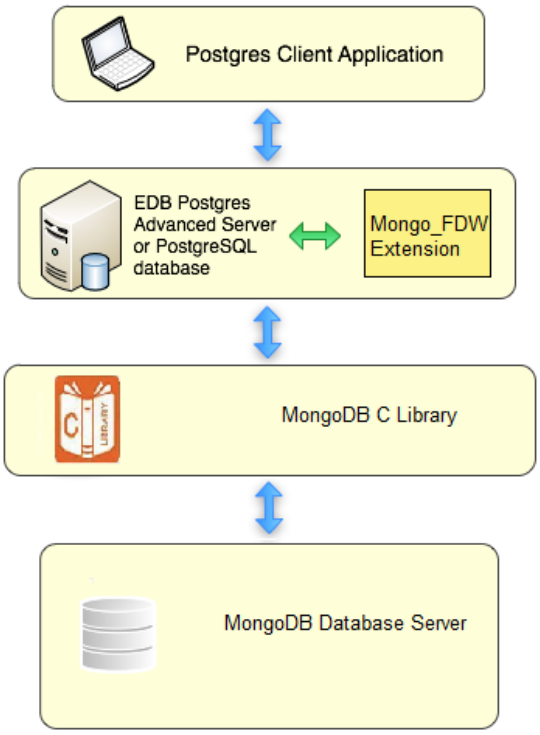
The following limitations apply to MongoDB Foreign Data Wrapper:

- If the BSON document key contains uppercase letters or occurs in a nested document, MongoDB Foreign Data Wrapper requires the corresponding column names to be declared in double quotes.
- PostgreSQL limits column names to 63 characters by default. You can increase the `NAMEDATALEN` constant in `src/include/pg_config_manual.h`, compile, and reinstall when column names exceed 63 characters.
- MongoDB Foreign Data Wrapper returns an error on BSON field that isn't listed in the known types (for example, byte, arrays). It returns this error: `Cannot convert BSON type to column type`.

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

5 Architecture overview

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



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

6 Key features

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

Writable FDW

The MongoDB Foreign Data Wrapper lets you modify data on a MongoDB server. You can insert, update, and delete data in the remote MongoDB collections by inserting, updating, and deleting data locally in foreign tables.

For more information, see:

- Example: Using the MongoDB Foreign Data Wrapper
- Data type mappings

WHERE clause pushdown

MongoDB Foreign Data Wrapper allows the pushdown of the `WHERE` clause only when clauses include the comparison expressions that have a column and a constant as arguments. `WHERE` clause pushdown isn't supported where the constant is an array. From version 5.5.0 and later, MongoDB Foreign Data Wrapper supports recursive operator expressions, Boolean expressions, relabel types, and vars on both sides of an operator.

For more information, see Example: WHERE clause pushdown.

Join pushdown

MongoDB Foreign Data Wrapper supports pushdown for inner joins, left joins, and right joins. Currently, joins involving only relational and arithmetic operators in join clauses are pushed down to avoid any potential join failures.

For more information, see Example: Join pushdown.

Aggregate pushdown

MongoDB Foreign Data Wrapper supports aggregate pushdown. It pushes the aggregates to the remote MongoDB server instead of fetching all of the rows and aggregating them locally. This gives a very good performance boost for the cases where aggregates can be pushed down. The pushdown is currently limited to aggregate functions min, max, sum, avg, and count, to avoid pushing down the functions that are not present on the MongoDB server. The aggregate filters, orders, variadic and distinct are not pushed down.

For more information, see Example: Aggregate pushdown.

ORDER BY pushdown

MongoDB Foreign Data Wrapper supports `ORDER BY` pushdown. If possible, push the `ORDER BY` clause to the remote server. This approach provides the ordered result set from the foreign server, which can help to have an efficient merge join. NULLs behavior is opposite on the MongoDB server. To get an equivalent result, push down `ORDER BY` with either `ASC NULLS FIRST` or `DESC NULLS LAST`. As MongoDB sorts only on fields, only column names in `ORDER BY` expressions are pushed down.

For more information, see Example: ORDER BY pushdown.

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

LIMIT OFFSET pushdown

MongoDB Foreign Data Wrapper supports `LIMIT` / `OFFSET` pushdown. Wherever possible, perform `LIMIT` and `OFFSET` operations on the remote server. This reduces network traffic between local Postgres and remote MongoDB servers.

For more information, see Example: LIMIT OFFSET pushdown.

Connection pooling

The MongoDB 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.

Automated cleanup

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

```
DROP EXTENSION mongo_fdw CASCADE;
```

For more information, see DROP EXTENSION.

Full-document retrieval

This feature lets you retrieve documents along with all their fields from collection without any knowledge of the fields in the BSON document available in MongoDB's collection. Those retrieved documents are in JSON format.

You can retrieve all available fields in a collection residing in MongoDB Foreign Data Wrapper as explained in the following example.

Example

```
> db.warehouse.find();
{ "_id" : ObjectId("58a1ebbaf543ec0b90545859"), "warehouse_id" : 1, "warehouse_name" : "UPS",
"warehouse_created" : ISODate("2014-12-12T07:12:10Z") }
{ "_id" : ObjectId("58a1ebbaf543ec0b9054585a"), "warehouse_id" : 2, "warehouse_name" : "Laptop",
"warehouse_created" : ISODate("2015-11-11T08:13:10Z") }
```

To retrieve the document:

1. Create a foreign table with a column name `__doc` . The type of the column can be json, jsonb, text, or varchar.

   ```
   CREATE FOREIGN TABLE test_json(__doc json) SERVER mongo_server OPTIONS (database 'testdb', collection
   'warehouse');
   ```

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

2. Retrieve the document.

```
SELECT * FROM test_json ORDER BY __doc::text COLLATE
"C";
```

The following is the output:

```
edb=#SELECT * FROM test_json ORDER BY __doc::text COLLATE
"C";
```

output

```
                                                                    __doc
--------------------------------------------------------------------------------------------------------------
----------------------------------------------
{ "_id" : { "$oid" : "58a1ebbaf543ec0b90545859" }, "warehouse_id" : 1, "warehouse_name" : "UPS",
"warehouse_created" : { "$date" : 1418368330000 } }
{ "_id" : { "$oid" : "58a1ebbaf543ec0b9054585a" }, "warehouse_id" : 2, "warehouse_name" : "Laptop",
"warehouse_created" : { "$date" : 1447229590000 } }
(2 rows)
```

Accessing nested fields

MongoDB Foreign Data Wrapper allows you to access individual fields in nested JSON documents by mapping the nested structure to columns in a foreign table. This works by mapping the nested structure of the MongoDB document to relational columns in the foreign table definition, using dot notation (key2.subkey21) to reference nested fields. You can retrieve these fields from a collection as shown in the following example.

Example

```
db1> db.test_sub_json.find()
[
  {
    _id: ObjectId('658040214890799d6e0173d0'),
    key1: 'hello',
    key2: {
      subkey21: 'hello-sub1',
      subkey22: 'hello-sub2',
      subtstmp: ISODate('2022-12-16T19:16:17.801Z')
    }
  }
]
```

To retrieve subfields from the document:

1. Create a foreign table. To access a subfield, use the dot (".") in the column name:

```
CREATE FOREIGN TABLE ft_nested_json_test(
    _id
NAME,
    key1 varchar,
    "key2.subkey21"   varchar,
    "key2.subkey22"   varchar,
    "key2.subtstmp"   timestamp
)SERVER mongo_server
OPTIONS (database 'db1', collection
'test_sub_json');
```

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

2. Retrieve the document with subfields:

```
SELECT * FROM ft_nested_json_test;
```

```
                                                output
          _id            | key1  | key2.subkey21 | key2.subkey22 |     key2.subtstmp
-------------------------+-------+---------------+---------------+-----------------------
 658040214890799d6e0173d0 | hello | hello-sub1    | hello-sub2    | 16-DEC-22 19:16:17.801
```

3. Retrieve an individual field:

```
SELECT "key2.subkey21" FROM ft_nested_json_test;
```

```
                output
 key2.subkey21
---------------
 hello-sub1
```

Remote query logging

Starting from version 5.5.3, MongoDB Foreign Data Wrapper has a new `log_remote_query` GUC parameter, which, when enabled, logs the remote query executed on the MongoDB server directly to the PostgreSQL server logs. This feature is designed exclusively for debugging purposes, aiding in troubleshooting and development by providing visibility into the exact queries sent to MongoDB at a session level. By default, this parameter is set to false.

For more information, see Example: remote query logging.

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

7 Installing MongoDB 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

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

Debian and derivatives

- Debian 12

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

7.1 Installing MongoDB 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

After you complete the installation, see Initial configuration.

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

7.1.1 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.1.2 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.1.3 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

7.1.4 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

7.1.5 Installing MongoDB 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.
- 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>-mongo_fdw

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

# For PostgreSQL:
sudo zypper -n install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

7.1.6 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo apt-get -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.1.7 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo apt-get -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.1.8 Installing MongoDB 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.

Install the package

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

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

# For PostgreSQL:
sudo apt-get -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.1.9 Installing MongoDB 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.

Install the package

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

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

# For PostgreSQL:
sudo apt-get -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.2 Installing MongoDB 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

After you complete the installation, see Initial configuration.

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

7.2.1 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.2.2 Installing MongoDB 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.

Install the package

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

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

# For PostgreSQL:
sudo apt-get -y install postgresql-<X>-mongo-fdw
```

Where `<X>` is the version of your Postgres.

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

7.3 Installing MongoDB 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

After you complete the installation, see Initial configuration.

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

7.3.1 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

7.3.2 Installing MongoDB 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>-mongo_fdw

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

# For PostgreSQL:
sudo dnf -y install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

7.3.3 Installing MongoDB 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.
- 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>-mongo_fdw

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

# For PostgreSQL:
sudo zypper -n install postgresql-<X>-mongo-fdw
```

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

Where `<X>` is the version of your Postgres.

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

8 Initial Configuration

Before using the MongoDB Foreign Data Wrapper:

1. Use the CREATE EXTENSION command to create the MongoDB Foreign Data Wrapper extension on the Postgres host.
2. Use the CREATE SERVER command to define a connection to the MongoDB 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 table in the Postgres database that corresponds to a database that resides on the MongoDB cluster.

CREATE EXTENSION

Use the `CREATE EXTENSION` command to create the `mongo_fdw` extension. To invoke the command, use your client of choice (for example, psql) to connect to the Postgres database from which you want to query the MongoDB server, and invoke the command:

```
CREATE EXTENSION [IF NOT EXISTS] mongo_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 MongoDB foreign data wrapper:

```
CREATE EXTENSION mongo_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
mongo_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.

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

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 `mongo_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 object. You can include these options.

Option	Description
address	The address or host name of the Mongo server. The default value is `127.0.0.1`.
port	The port number of the Mongo server. Valid range is 0 to 65535. The default value is `27017`.
authentication_database	The database against which the user is authenticated. This option is valid only with password-based authentication.
ssl	Requests an authenticated, encrypted SSL connection. By default, the value is set to `false`. Set the value to `true` to enable SSL. See mongoc_ssl_opt_t to understand the options.
pem_file	SSL option.
pem_pwd	SSL option.
ca_file	SSL option.
ca_dir	SSL option.
crl_file	SSL option.
weak_cert_validation	SSL option. The default value is `false`.
enable_aggregate_pushdown	Similar to the table-level option but configured at the server level. If `true`, pushes the aggregate operations to the foreign server instead of fetching rows from the foreign server and performing the operations locally. You can also set this option for an individual table. The table-level value of the option takes precedence over the server-level option value. Default is `true`.
enable_join_pushdown	Similar to the table-level option but configured at the server level. If `true`, pushes the join between two foreign tables from the same foreign server instead of fetching all the rows for both the tables and performing a join locally. You can also set this option for an individual table. In this case, if any of the tables involved in the join has the option set to `false`, then the join isn't pushed down. The table-level value of the option takes precedence over the server-level option value. Default is `true`.
enable_order_by_pushdown	Similar to the table-level option but configured at the server level. If `true`, pushes the order-by operation to the foreign server instead of fetching rows from the foreign server and performing the sort locally. You can also set this option for an individual table. The table-level value of the option takes precedence over the server-level option value. Default is `true`.

Example

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

```
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (host '127.0.0.1', port
'27017');
```

The foreign server uses the default port (`27017`) for the connection to the client on the MongoDB cluster.

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

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

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 MongoDB cluster.

`OPTIONS`

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

`username` is the name of the user on the MongoDB 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 `mongo_server`.

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

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

```
CREATE USER MAPPING FOR enterprisedb SERVER mongo_server OPTIONS (username 'mongo_user', password
'mongo_pass');
```

The command creates a user mapping for a role named `enterprisedb` that is associated with a server named `mongo_server`. When connecting to the MongoDB server, the server authenticates as `mongo_user` and provides a password of `mongo_pass`.

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

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

CREATE FOREIGN TABLE

A foreign table is a pointer to a table that resides on the MongoDB host. Before creating a foreign table definition on the Postgres server, connect to the MongoDB server and create a collection. 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 collection that resides on the MongoDB 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`

Specify the name of the foreign table. Include a schema name to specify the schema in which the foreign table resides.

`IF NOT EXISTS`

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`

Specify the name of a column in the new table. Each column must correspond to a column described on the MongoDB server.

`data_type`

Specify the data type of the column. When possible, specify the same data type for each column on the Postgres server and the MongoDB 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 MongoDB 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.

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

`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 must reference 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 can't 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 MongoDB file system, include the `SERVER` clause and specify `server_name` for the foreign server that uses the MongoDB data adapter.

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

Option	Value
database	The name of the database to query. The default value is `test`.
collection	The name of the collection to query. The default value is the foreign table name.
enable_aggregate_pushdown	Similar to the server-level option but configured at the table level. If `true`, pushes the aggregate operations to the foreign server instead of fetching rows from the foreign server and performing the operations locally. You can also set this option for an individual table. The table-level value of the option takes precedence over the server-level option value. Default is `true`.
enable_join_pushdown	Similar to the server-level option but configured at the table level. If `true`, pushes the join between two foreign tables from the same foreign server instead of fetching all the rows for both the tables and performing a join locally. You can also set this option for an individual table. In this case, if any of the tables involved in the join has set the option to `false`, then the join isn't pushed down. The table-level value of the option takes precedence over the server-level option value. Default is `true`.
enable_order_by_pushdown	Similar to the server-level option but configured at the table level. If `true`, pushes the order-by operation to the foreign server instead of fetching rows from the foreign server and performing the sort locally. You can also set this option for an individual table. The table-level value of the option takes precedence over the server-level option value. Default is `true`.

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

Example

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

```
db.warehouse.find
(

{
                "warehouse_id" :
1

}
).pretty()
{
        "_id" :
ObjectId("53720b1904864dc1f5a571a0"),
        "warehouse_id" :
1,
        "warehouse_name" :
"UPS",
        "warehouse_created" : ISODate("2014-12-
12T07:12:10Z")
}
```

Execute a command on the Postgres server that creates a comparable table on the Postgres server:

```
CREATE FOREIGN TABLE warehouse
(
 _id
NAME,
 warehouse_id      INT,
 warehouse_name
TEXT,
 warehouse_created TIMESTAMPZ
)
SERVER mongo_server
OPTIONS (database 'db', collection
'warehouse');
```

The first column of the table must be `_id` of the type `name`.

Include the `SERVER` clause to specify the name of the database stored on the MongoDB 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**
>
> MongoDB 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 MongoDB server. The MongoDB data wrapper converts the following MongoDB data types to the target Postgres type:

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

MongoDB (BSON Type)	Postgres
ARRAY	JSON
BOOL	BOOL
BINARY	BYTEA
DATE_TIME	DATE/TIMESTAMP/TIMESTAMPTZ
DOCUMENT	JSON
DOUBLE	FLOAT/FLOAT4/FLOAT8/DOUBLE PRECISION/NUMERIC
INT32	SMALLINT/INT2/INT/INTEGER/INT4
INT64	BIGINT/INT8
OID	NAME
UTF8	BPCHAR/VARCHAR/CHARACTER VARYING/TEXT

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 MongoDB 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.

Example

The following command removes the extension from the existing database:

```
DROP EXTENSION mongo_fdw;
```

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

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

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 exist.

`name`

Optionally, specify the name of the installed server.

`CASCADE`

Drop objects that depend on the server. It drops 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 `mongo_server` :

```
DROP SERVER mongo_server;
```

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

DROP USER MAPPING

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;
```

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

Parameters

`IF EXISTS`

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

`user_name`

Specify the user name of the mapping.

`server_name`

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

Example

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

```
DROP USER MAPPING FOR enterprisedb SERVER mongo_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 MongoDB 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. It drops all the other dependent objects too.

`RESTRICT`

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

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

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.

9 Upgrading

If you have an existing installation of MongoDB Foreign Data Wrapper that you installed using the EDB repository, you can update your repository configuration file and then upgrade MongoDB 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 upgrade edb-as<xx>-mongo_fdw`
SLES	`sudo zypper upgrade edb-as<xx>-mongo_fdw`
Debian and Ubuntu	`sudo apt-get --only-upgrade install edb-as<xx>-mongo-fdw edb-libmongoc`

Where

- `<package-manager>` is the package manager used with your operating system.
- `<xx>` is the EDB Postgres Advanced Server version number.

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

10 Uninstalling

You can use the `remove` command to uninstall MongoDB 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. `xx` is the EDB Postgres Advanced Server version number.

- On RHEL or Rocky Linux or AlmaLinux 8:

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

- On SLES:

  ```
  zypper remove edb-as<xx>-mongo_fdw
  ```

- On Debian or Ubuntu

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

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

11 Example: End-to-end

Before using the MongoDB foreign data wrapper, you must connect to your database with a client application. The following example uses the wrapper with the psql client. After connecting to psql, you can follow the steps in the example:

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

-- create server
object
CREATE SERVER
mongo_server
          FOREIGN DATA WRAPPER mongo_fdw
          OPTIONS (address '127.0.0.1', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR enterprisedb
          SERVER
mongo_server
          OPTIONS (username 'mongo_user', password 'mongo_pass');

-- create foreign
table
CREATE FOREIGN TABLE warehouse

(
            _id
name,
            warehouse_id int,
            warehouse_name
text,
            warehouse_created
timestamptz

)
          SERVER
mongo_server
          OPTIONS (database 'db', collection
'warehouse');

-- Note: first column of the table must be "_id" of type
"name".

-- select from
table
SELECT * FROM warehouse WHERE warehouse_id = 1;
           _id            | warehouse_id | warehouse_name |
warehouse_created
--------------------------+--------------+----------------+---------------------------
 53720b1904864dc1f5a571a0 |            1 | UPS            | 2014-12-12
12:42:10+05:30
(1 row)

db.warehouse.find
(

{
            "warehouse_id" :
1

}
).pretty()
{
```

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

```
        "_id" :
ObjectId("53720b1904864dc1f5a571a0"),
        "warehouse_id" :
1,
        "warehouse_name" :
"UPS",
        "warehouse_created" : ISODate("2014-12-
12T07:12:10Z")
}

-- insert row in
table

INSERT INTO warehouse VALUES (0, 2, 'Laptop', '2015-11-
11T08:13:10Z');

db.warehouse.insert
(

{
            "warehouse_id" :
NumberInt(2),
            "warehouse_name" :
"Laptop",
            "warehouse_created" : ISODate("2015-11-
11T08:13:10Z")

}
)
-- Note: The given value for "_id" column will be ignored and allow MongoDB to insert
-- the unique value for the "_id"
column.

-- delete row from
table
DELETE FROM warehouse WHERE warehouse_id = 2;

db.warehouse.remove
(

{
            "warehouse_id" :
2

}
)

-- update a row of
table
UPDATE warehouse SET warehouse_name = 'UPS_NEW' WHERE warehouse_id =
1;

db.warehouse.update
(

{
            "warehouse_id" :
1
        },

{
            "warehouse_id" :
1,
            "warehouse_name" :
"UPS_NEW",
            "warehouse_created" : ISODate("2014-12-
12T07:12:10Z")
```

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

```
}
)

-- explain a
table
EXPLAIN SELECT * FROM warehouse WHERE warehouse_id = 1;
                            QUERY PLAN
------------------------------------------------------------------
 Foreign Scan on warehouse  (cost=0.00..0.00 rows=1000
width=84)
   Filter: (warehouse_id = 1)
   Foreign Namespace: db.warehouse
(3 rows)

-- collect data distribution statistics
ANALYZE warehouse;

-- drop foreign table
DROP FOREIGN TABLE warehouse;

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

-- drop
server
DROP SERVER
mongo_server;
```

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

12 Example: Join pushdown

MongoDB Foreign Data Wrapper supports pushdown for inner joins, left joins, and right joins. For example:

Postgres data set:

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

-- create server
object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (address 'localhost', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR public SERVER mongo_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE emp (_id NAME, eid INTEGER, ename TEXT, deptid INTEGER) SERVER mongo_server OPTIONS
(database 'edb', collection 'emp');

-- insert into
table
INSERT INTO emp VALUES (0, 100, 'John',
10);
INSERT INTO emp VALUES (0, 110, 'Mark',
10);
INSERT INTO emp VALUES (0, 120, 'Smith',
20);
INSERT INTO emp VALUES (0, 130, 'Ed',
30);

-- create foreign
table
CREATE FOREIGN TABLE dept (_id NAME, deptid INTEGER, dname TEXT) SERVER mongo_server OPTIONS (database
'edb', collection 'dept');

-- insert into
table
INSERT INTO dept VALUES (0, 10,
'SALES');
INSERT INTO dept VALUES (0, 20,
'ENGG');
INSERT INTO dept VALUES (0, 30,
'IT');
```

Enable/disable GUC for JOIN pushdown queries at the session level, table level, or server level:

```
-- Session level
edb=# SET mongo_fdw.enable_join_pushdown to true;
SET
-- Table level
edb=# ALTER FOREIGN TABLE emp OPTIONS (ADD enable_join_pushdown
'true');
Table altered
-- Server
level
edb=# ALTER SERVER mongo_server OPTIONS (ADD enable_join_pushdown
'true');
altered
```

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

Query with JOIN pushdown:

```
--inner join
edb=# EXPLAIN VERBOSE SELECT e.ename, d.dname FROM emp e INNER JOIN dept d ON (e.deptid =
d.deptid);
```

output

```
                        QUERY PLAN
-----------------------------------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=64)
   Output: e.ename, d.dname
   Foreign Namespace: (edb.emp e) INNER JOIN (edb.dept d)
(3 rows)
```

```
--left join
edb=# EXPLAIN VERBOSE SELECT e.ename, d.dname FROM emp e LEFT JOIN dept d ON (e.deptid =
d.deptid);
```

output

```
                       QUERY PLAN
-----------------------------------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=64)
   Output: e.ename, d.dname
   Foreign Namespace: (edb.emp e) LEFT JOIN (edb.dept d)
(3 rows)
```

```
--right join
edb=# EXPLAIN VERBOSE SELECT e.ename, d.dname FROM emp e RIGHT JOIN dept d ON (e.deptid =
d.deptid);
```

output

```
                       QUERY PLAN
-----------------------------------------------------------
 Foreign Scan  (cost=15.00..35.00 rows=5000 width=64)
   Output: e.ename, d.dname
   Foreign Namespace: (edb.dept d) LEFT JOIN (edb.emp e)
(3 rows)
```

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

13 Example: Aggregate function pushdown

MongoDB 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.

Postgres data set:

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

-- create server
object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (address 'localhost', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR public SERVER mongo_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE emp (_id NAME, eid INTEGER, deptid INTEGER) SERVER mongo_server OPTIONS (database
'edb', collection 'emp');

-- insert into
table
INSERT INTO emp VALUES (0, 100,
10);
INSERT INTO emp VALUES (0, 110,
10);
INSERT INTO emp VALUES (0, 120,
20);
INSERT INTO emp VALUES (0, 130,
30);
```

Enable/disable GUC for aggregate pushdown queries at the session level, table level, or server level:

```
-- Session level
edb=# SET mongo_fdw.enable_aggregate_pushdown to true;
SET
-- Table level
edb=# ALTER FOREIGN TABLE emp OPTIONS (ADD enable_aggregate_pushdown
'true');
Table altered
-- Server
level
edb=# ALTER SERVER mongo_server OPTIONS (ADD enable_aggregate_pushdown
'true');
altered
```

Query with aggregate pushdown:

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

```
-- COUNT
function
edb# EXPLAIN VERBOSE SELECT COUNT(*) FROM emp;
```

output

```
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=8)
   Output: (count(*))
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

```
-- SUM
function
edb# EXPLAIN VERBOSE SELECT SUM(deptid) FROM emp;
```

output

```
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=8)
   Output: (sum(deptid))
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

```
-- AVG
function
edb# EXPLAIN VERBOSE SELECT AVG(deptid) FROM emp;
```

output

```
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=32)
   Output: (avg(deptid))
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

```
-- MAX
function
edb# EXPLAIN VERBOSE SELECT MAX(eid) FROM emp;
```

output

```
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=4)
   Output: (max(eid))
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

```
-- MIN
function
edb# EXPLAIN VERBOSE SELECT MIN(eid) FROM emp;
```

output

```
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=1 width=4)
   Output: (min(eid))
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

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

```
-- MIN and SUM functions with
GROUPBY
edb# EXPLAIN VERBOSE SELECT MIN(deptid), SUM(eid) FROM emp GROUP BY deptid HAVING MAX(eid) >
120;
```

output

```
                     QUERY PLAN
-----------------------------------------------------
 Foreign Scan  (cost=15.00..25.00 rows=200 width=16)
   Output: (min(deptid)), (sum(eid)), deptid
   Foreign Namespace: Aggregate on (db1.emp)
(3 rows)
```

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

14 Example: LIMIT OFFSET pushdown

This example shows LIMIT OFFSET pushdown on the EMP table.

Postgres data set:

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

-- create server
object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (address 'localhost', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR public SERVER mongo_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE emp (_id NAME, eid INTEGER, ename TEXT, deptid INTEGER) SERVER mongo_server OPTIONS
(database 'edb', collection 'emp');

-- insert into
table
INSERT INTO emp VALUES (0, 100, 'John',
10);
INSERT INTO emp VALUES (0, 110, 'Mark',
10);
INSERT INTO emp VALUES (0, 120, 'Smith',
20);
INSERT INTO emp VALUES (0, 130, 'Ed',
30);
```

```
-- LIMIT and
OFFSET
edb=# SELECT min(eid), eid FROM emp GROUP BY eid ORDER BY eid ASC NULLS FIRST LIMIT 0 OFFSET
0;
 min | eid
-----+-----
(0 rows)

-- LIMIT and
OFFSET
edb=# EXPLAIN (VERBOSE, COSTS OFF)
edb-# SELECT min(eid), eid FROM emp GROUP BY eid ORDER BY eid ASC NULLS FIRST LIMIT NULL OFFSET
2;
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan
   Output: (min(eid)),
eid
   Foreign Namespace: Aggregate on
("FDW_134".emp)
(3 rows)

-- LIMIT and
OFFSET
edb=# SELECT min(eid), eid FROM emp GROUP BY eid ORDER BY eid ASC NULLS FIRST LIMIT NULL OFFSET
2;
 min | eid
-----+-----
```

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

```
 120 |
120
 130 |
130
 140 |
140
(3 rows)

-- LIMIT and
OFFSET
edb=# EXPLAIN (VERBOSE, COSTS OFF)
edb-# SELECT min(eid), eid FROM emp GROUP BY eid ORDER BY eid ASC NULLS FIRST LIMIT ALL OFFSET
2;
                    QUERY PLAN
--------------------------------------------------
 Foreign Scan
   Output: (min(eid)),
eid
   Foreign Namespace: Aggregate on
("FDW_134".emp)
(3 rows)

-- LIMIT only
edb=# EXPLAIN (VERBOSE, COSTS FALSE)
edb-# SELECT eid, max(eid) FROM emp GROUP BY eid ORDER BY 1 ASC NULLS FIRST LIMIT
-1;
                  QUERY PLAN
-----------------------------------------------
 Limit
   Output: eid,
(max(eid))
   ->
GroupAggregate
         Output: eid, max(eid)
         Group Key: emp.eid
         ->  Foreign Scan on public.emp
               Output: _id, eid,
deptid
               Foreign Namespace:
FDW_134.emp
(8 rows)

-- OFFSET
only
edb=# EXPLAIN (VERBOSE, COSTS FALSE)
edb-# SELECT eid, max(eid) FROM emp GROUP BY eid ORDER BY 1 ASC NULLS FIRST OFFSET
-2;
                      QUERY PLAN
-------------------------------------------------------
 Limit
   Output: eid,
(max(eid))
   ->  Foreign Scan
         Output: eid,
(max(eid))
         Foreign Namespace: Aggregate on
("FDW_134".emp)
(5 rows)

edb=#
```

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

15 Example: ORDER BY pushdown

This example shows ORDER BY pushdown on the EMP table.

Postgres data set:

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

-- create server
object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (address 'localhost', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR public SERVER mongo_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE emp (_id NAME, eid INTEGER, ename TEXT, deptid INTEGER) SERVER mongo_server OPTIONS
(database 'edb', collection 'emp');

-- insert into
table
INSERT INTO emp VALUES (0, 100, 'John',
10);
INSERT INTO emp VALUES (0, 110, 'Mark',
10);
INSERT INTO emp VALUES (0, 120, 'Smith',
20);
INSERT INTO emp VALUES (0, 130, 'Ed',
30);
```

```
edb=# SELECT eid, sum(eid), count(*) FROM emp GROUP BY eid HAVING min(eid) > 100
ORDER
edb-# BY eid ASC NULLS
FIRST;
 eid | sum | count
-----+-----+-------
 110 | 110 |
1
 120 | 120 |
1
 130 | 130 |
1
 140 | 140 |
1
(4 rows)
```

Enable/disable GUC for ORDER BY pushdown queries at the session level, table level, or server level:

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

```
-- Session level
edb=# SET mongo_fdw.enable_order_by_pushdown to
true;
SET
-- Table level
edb=# ALTER FOREIGN TABLE emp OPTIONS (ADD enable_order_by_pushdown
'true');
Table altered
-- Server
level
edb=# ALTER SERVER mongo_server OPTIONS (ADD enable_order_by_pushdown
'true');
altered
```

Query with ORDER BY pushdown:

```
edb=# EXPLAIN (VERBOSE, COSTS FALSE)
edb-# SELECT eid, sum(eid), count(*) FROM emp GROUP BY eid HAVING min(eid) > 100
ORDER
edb-# BY eid ASC NULLS
FIRST;
                     QUERY PLAN
-----------------------------------------------------
 Foreign Scan
   Output: eid, (sum(eid)),
(count(*))
   Foreign Namespace: Aggregate on
("FDW_134".emp)
(3 rows)

edb=#
edb=# SELECT deptid, min(eid) FROM emp WHERE deptid > 20 GROUP BY deptid HAVING min(deptid) =
edb-# 30 ORDER BY deptid ASC NULLS
FIRST;
 deptid | min
--------+-----
     30 |
120
(1 row)

edb=#
edb=# EXPLAIN (VERBOSE, COSTS FALSE)
edb-# SELECT deptid, min(eid) FROM emp WHERE deptid > 20 GROUP BY deptid HAVING min(deptid) =
edb-# 30 ORDER BY deptid ASC NULLS
FIRST;
                     QUERY PLAN
-----------------------------------------------------
 Foreign Scan
   Output: deptid,
(min(eid))
   Foreign Namespace: Aggregate on
("FDW_134".emp)
(3 rows)
```

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

16 Example: WHERE clause pushdown

MongoDB Foreign Data Wrapper supports pushdown for the WHERE clause. For example:

Postgres data set:

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

-- create server
object
CREATE SERVER mongo_server FOREIGN DATA WRAPPER mongo_fdw OPTIONS (address 'localhost', port
'27017');

-- create user
mapping
CREATE USER MAPPING FOR public SERVER mongo_server OPTIONS (username 'edb', password
'edb');

-- create foreign
table
CREATE FOREIGN TABLE emp (_id NAME, eid INTEGER, ename TEXT, deptid INTEGER) SERVER mongo_server OPTIONS
(database 'edb', collection 'emp');

-- insert into
table
INSERT INTO emp VALUES (0, 100, 'John',
10);
INSERT INTO emp VALUES (0, 110, 'Mark',
10);
INSERT INTO emp VALUES (0, 120, 'Smith',
20);
INSERT INTO emp VALUES (0, 130, 'Ed',
30);
```

The output:

```
edb=# EXPLAIN (VERBOSE, COSTS FALSE) select eid from emp where deptid>20 order by
eid;
             QUERY PLAN
------------------------------------
 Sort
   Output:
eid
   Sort Key: emp.eid
   ->  Foreign Scan on public.emp
         Output:
eid
         Foreign Namespace: edb.emp
(6 rows)

edb=#
edb=# select eid from emp where deptid>20 order by
eid;
 eid
-----
 130
(1 row)
```

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

17 Identifying the version

The MongoDB 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 mongo_fdw_version();
```

The function returns the version number:

```
mongo_fdw_version
-----------------
<xxxxx>
```

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

18 Example: Remote query logging

MongoDB Foreign Data Wrapper supports storing logs for queries executed on the MongoDB server directly to the PostgreSQL server logs. For example:

```
SET mongo_fdw.log_remote_query TO
true;
SET client_min_messages TO
log;
SELECT c1, c2, c6 FROM f_test_tbl1
e
  WHERE c6 > 3000
  ORDER BY c1 ASC NULLS FIRST;
LOG:  remote query: db.test_tbl1.aggregate( [ { "$match" : { "$expr" : { "$and" : [ { "$gt" : [ "$c6",
3000.0 ] }, { "$ne" : [ "$c6", null ] } ] } } }, { "$sort" : { "c1" : 1 } } ] )
 c1  |  c2  |  c6
-----+------+------
 900 | EMP9 |
5000
(1 row)
```

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