



EDB

MongoDB Foreign Data Wrapper Guide

Release 5.2.8

MongoDB Foreign Data Wrapper User's Guide

Feb 13, 2021

CONTENTS

1	What's New	3
2	Requirements Overview	5
2.1	Supported Versions	5
2.2	Supported Platforms	5
3	Architecture Overview	7
4	Installing the MongoDB Foreign Data Wrapper	9
4.1	Installing the MongoDB Foreign Data Wrapper using an RPM Package	9
4.1.1	On RHEL 7	9
4.1.2	On RHEL 8	11
4.1.3	On CentOS 7	13
4.1.4	On CentOS 8	15
4.2	Installing the MongoDB Foreign Data Wrapper on a Debian or Ubuntu Host	17
5	Features of the MongoDB Foreign Data Wrapper	19
5.1	Writable FDW	19
5.2	Connection Pooling	19
5.3	Automated Cleanup	19
6	Configuring the MongoDB Foreign Data Wrapper	21
6.1	CREATE EXTENSION	21
6.2	CREATE SERVER	22
6.3	CREATE USER MAPPING	24
6.4	CREATE FOREIGN TABLE	25
6.4.1	Data Type Mappings	27
7	Example: Using the MongoDB Foreign Data Wrapper	29
8	Identifying the MongoDB Foreign Data Wrapper Version	31
9	Conclusion	33
	Index	35

The MongoDB Foreign Data Wrapper (`mongo_fdw`) is a Postgres extension that allows you to access data that resides on a MongoDB database from EDB Postgres Advanced Server. It is a writable foreign data wrapper that you can use with Postgres functions and utilities, or in conjunction with other data that resides on a Postgres host.

The MongoDB Foreign Data Wrapper can be installed with an RPM package. You can download an installer from the [EDB website](#).

This guide uses the term `Postgres` to refer to an instance of EDB Postgres Advanced Server.

WHAT'S NEW

The following features are added to create MongoDB Foreign Data Wrapper 5 . 2 . 8:

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

REQUIREMENTS OVERVIEW

2.1 Supported Versions

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

2.2 Supported Platforms

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

Linux x86-64

- RHEL 8.x/7.x
- CentOS 8.x/7.x
- OEL 8.x/7.x
- Ubuntu 20.04/18.04 LTS
- Debian 10.x/9.x

Linux on IBM Power8/9 (LE)

- RHEL 7.x

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 is understood by the MongoDB database.

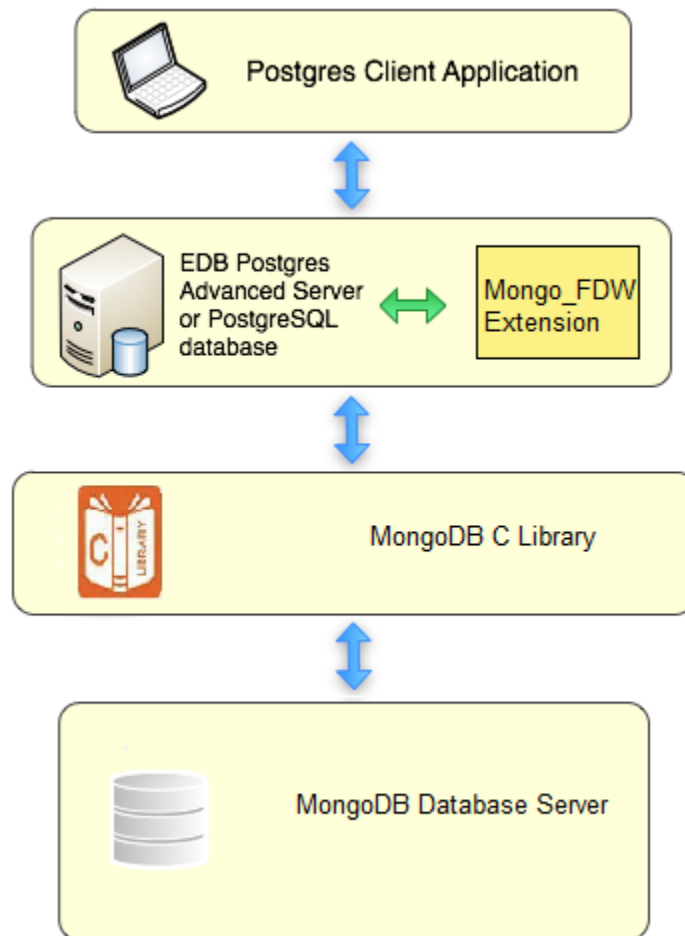


Fig. 1: *Using MongoDB FDW with Postgres*

INSTALLING THE MONGODB FOREIGN DATA WRAPPER

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

4.1 Installing the MongoDB Foreign Data Wrapper using an RPM Package

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

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

4.1.1 On RHEL 7

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

Install the `epel-release` package:

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

Enable the optional, extras, and HA repositories:

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

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

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

After receiving your repository credentials:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-as<xx>-mongo_fdw`.

Creating a Repository Configuration File

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

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

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

Modifying the file to provide your user name and password

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

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

Installing the MongoDB Foreign Data Wrapper

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

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

where `xx` is the server version number.

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

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

4.1.2 On RHEL 8

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

Install the `epel-release` package:

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

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

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

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

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

After receiving your repository credentials:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-as<xx>-mongo_fdw`.

Creating a Repository Configuration File

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

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

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

Modifying the file to provide your user name and password

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

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

Installing the MongoDB Foreign Data Wrapper

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

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

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

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

4.1.3 On CentOS 7

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

Install the `epel-release` package:

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

Note: You may need to enable the `[extras]` repository definition in the `CentOS-Base.repo` file (located in `/etc/yum.repos.d`).

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

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

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-as<xx>-mongo_fdw`.

Creating a Repository Configuration File

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

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

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

Modifying the file to provide your user name and password

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

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

Installing the MongoDB Foreign Data Wrapper

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

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

where `xx` is the server version number.

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

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

4.1.4 On CentOS 8

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

Install the `epel-release` package:

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

Enable the PowerTools repository:

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

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

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

After receiving your repository credentials:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install `edb-as<xx>-mongo_fdw`.

Creating a Repository Configuration File

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

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

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

Modifying the file to provide your user name and password

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

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

Installing the MongoDB Foreign Data Wrapper

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

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

where `xx` is the server version number.

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

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

4.2 Installing the MongoDB Foreign Data Wrapper on a Debian or Ubuntu Host

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

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

1. Assume superuser privileges:

```
sudo su -
```

2. Configure the EDB repository:

On Debian 9 and Ubuntu:

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

On Debian 10:

- a. Set up the EDB repository:

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

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

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

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

```
apt-get install apt-transport-https
```

4. Add the EDB signing key:

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

5. Update the repository metadata:

```
apt-get update
```

6. Install the Debian package:

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

where `xx` is the server version number.

FEATURES OF THE MONGODB FOREIGN DATA WRAPPER

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

5.1 Writable FDW

The MongoDB Foreign Data Wrapper allows you to modify data on a MongoDB server. Users can `INSERT`, `UPDATE` and `DELETE` data in the remote MongoDB collections by inserting, updating and deleting data locally in foreign tables. See also:

Example: Using the MongoDB Foreign Data Wrapper

Data Type Mappings

5.2 Connection Pooling

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

5.3 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 has been created for a temporary purpose, as in the case of data migration. The syntax of a `DROP EXTENSION` command is:

```
DROP EXTENSION mongo_fdw CASCADE;
```

For more information, see [DROP EXTENSION](#).

CONFIGURING THE MONGODB FOREIGN DATA WRAPPER

Before using the MongoDB Foreign Data Wrapper, you must:

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.

6.1 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 will be querying 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 throwing an error if an extension with the same name already exists.

`schema_name`

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

Example

The following command installs the MongoDB foreign data wrapper:

```
CREATE EXTENSION mongo_fdw;
```

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

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

6.2 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.

Parameters

`server_name`

Use `server_name` to specify a name for the foreign server. The server name must be unique within the database.

`FOREIGN_DATA_WRAPPER`

Include the `FOREIGN_DATA_WRAPPER` clause to specify that the server should use the `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:

Option	Description
<code>address</code>	The address or hostname of the Mongo server. The default value is <code>127.0.0.1</code> .
<code>port</code>	The port number of the Mongo Server. Valid range is 0 to 65535. The default value is <code>27017</code> .
<code>authentication_database</code>	The database against which user will be authenticated. This option is only valid with password based authentication.
<code>replica_set</code>	The replica set the server is member of. If it is set, the driver will auto-connect to correct primary in the replica set when writing.
<code>read_preference</code>	The order of read preference. Options available are: <code>primary</code> [default], <code>secondary</code> , <code>primaryPreferred</code> , <code>secondaryPreferred</code> , and <code>nearest</code> .
<code>ssl</code>	Requests an authenticated, encrypted SSL connection. By default, the value is set to <code>false</code> . Set the value to <code>true</code> to enable ssl. See http://mongoc.org/libmongoc/current/mongoc_ssl_opt_t.html to understand the options.
<code>pem_file</code>	SSL option
<code>pem_pwd</code>	SSL option.
<code>ca_file</code>	SSL option
<code>ca_dir</code>	SSL option
<code>crl_file</code>	SSL option
<code>weak_cert_validation</code>	SSL option

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:

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

6.3 CREATE USER MAPPING

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

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

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

Parameters

`role_name`

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

`server_name`

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

`OPTIONS`

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

`username`: the name of the user on the MongoDB server.

`password`: the password associated with the username.

Example

The following command creates a user mapping for a role named `enterprisedb`; the mapping is associated with a server named `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 will authenticate as `mongo_user`, and provide a password of `mongo_pass`.

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

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

6.4 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 will map to columns in a table on the Postgres server. Then, use the `CREATE FOREIGN TABLE` command to define a table on the Postgres server with columns that correspond to the 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' [, ... ] ) ]
```

where `column_constraint` is:

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

and `table_constraint` is:

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

Parameters

`table_name`

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

`IF NOT EXISTS`

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

`column_name`

Specifies the name of a column in the new table; each column should correspond to a column described on the MongoDB 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 MongoDB server. If a data type with the same name is not available, the Postgres server will attempt to cast the data type to a type compatible with the MongoDB server. If the server cannot identify a compatible data type, it will return an error.

`COLLATE collation`

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

`INHERITS (parent_table [, ... ])`

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

`CONSTRAINT constraint_name`

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

NOT NULL

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

NULL

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

CHECK (expr) [NO INHERIT]

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

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

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

DEFAULT default_expr

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

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

To create a foreign table that will allow you to query a table that resides on a MongoDB file system, include the SERVER clause and specify the server_name of 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.

Example

To use data that is 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")
}
```

You should 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:

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

Note: MongoDB foreign data wrapper supports the write capability feature.

6.4.1 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 will automatically convert the following MongoDB data types to the target Postgres type:

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

EXAMPLE: USING THE MONGODB FOREIGN DATA WRAPPER

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

```
-- 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 postgres
    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 timestampz
)
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()
{
```

(continues on next page)

(continued from previous page)

```

        "_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")
    }
)

-- 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")
    }
)

-- 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;

```

IDENTIFYING THE MONGODB FOREIGN DATA WRAPPER 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>
```


CONCLUSION

MongoDB Foreign Data Wrapper Guide

Copyright © 2020 - 2020 EnterpriseDB Corporation.

All rights reserved.

EnterpriseDB® Corporation

34 Crosby Drive, Suite 201, Bedford, MA 01730, USA

T +1 781 357 3390 F +1 978 467 1307 E

info@enterprisedb.com

www.enterprisedb.com

- EnterpriseDB and Postgres Enterprise Manager are registered trademarks of EnterpriseDB Corporation. EDB and EDB Postgres are trademarks of EnterpriseDB Corporation. Oracle is a registered trademark of Oracle, Inc. Other trademarks may be trademarks of their respective owners.
- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB Postgres product where present. EDB follows the same approach for additional input components, however the nature of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from entering incorrect or otherwise dangerous inputs.
- EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous inputs from authenticated, trusted users in the future, but will ensure all such features are designed and tested to ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.
- EDB does not warrant that we can or will anticipate all potential threats and therefore our process cannot fully guarantee that all potential vulnerabilities have been addressed or considered.

A

Architecture Overview, [7](#)

C

centos7, [13](#)

centos8, [15](#)

Conclusion, [33](#)

Configuring the MongoDB Foreign Data
Wrapper, [21](#)

E

Example: Using the MongoDB Foreign
Data Wrapper, [29](#)

F

Features of the MongoDB Foreign Data
Wrapper, [19](#)

I

Identifying the MongoDB Foreign Data
Wrapper Version, [31](#)

Installing MongoDB Data Adapter on
Debian, [17](#)

Installing MongoDB Data Adapter on
Ubuntu, [17](#)

Installing the MongoDB Foreign Data
Wrapper, [9](#)

R

Requirements Overview, [5](#)

rhel8, [11](#)

S

supported platforms, [5](#)

supported server versions, [5](#)

W

What's New, [3](#)