



EDB

MongoDB Foreign Data Wrapper

リリース 5.2.3

EnterpriseDB Corporation

2023 年 02 月 02 日

目次

第 1 章	Release notes	3
1.1	Version 5.5.0	3
1.2	Version 5.4.0	4
1.3	Version 5.3.0	4
1.4	Version 5.2.9	4
1.5	Version 5.2.8	5
1.6	Version 5.2.6	5
1.7	Version 5.2.3	6
第 2 章	Supported database versions	7
第 3 章	Limitations	9
第 4 章	Architecture overview	11
第 5 章	Key features	13
5.1	書き込み可能な FDW	13
5.2	WHERE 句プッシュダウン	13
5.3	参加プッシュダウン	13
5.4	アグリゲートプッシュダウン	14
5.5	ORDER BY プッシュダウン	14
5.6	LIMIT OFFSET プッシュダウン	14
5.7	接続プーリング	14
5.8	自動クリーンアップ	15
5.9	全文書取得	15
5.9.1	例	15
第 6 章	Installing	17
6.1	Red Hat Enterprise Linux (RHEL) およびその派生物	17
6.2	SUSE Linux Enterprise (SLES) とその派生	17
6.3	Debian とその派生物	17
6.4	Red Hat Enterprise Linux (RHEL)	18
6.5	SUSE Linux Enterprise (SLES)	18
6.5.1	Installing MongoDB Foreign Data Wrapper on Linux x86 (amd64)	18
6.5.2	Installing MongoDB Foreign Data Wrapper on IBM Power (ppc64le)	25
第 7 章	Initial Configuration	29
7.1	拡張機能を作成する	29
7.1.1	パラメーター	29
7.1.2	例	30
7.2	サーバーを作成する	30

7.2.1	パラメーター	30
7.2.2	例	31
7.3	ユーザーマッピングを作成する	31
7.3.1	パラメーター	32
7.3.2	例	32
7.4	外部テーブルを作成する	32
7.4.1	パラメーター	33
7.4.2	例	34
7.4.3	データ型のマッピング	35
7.5	ドロップエクステンション	36
7.5.1	パラメーター	36
7.5.2	例	37
7.6	ドロップサーバー	37
7.6.1	パラメーター	37
7.6.2	例	37
7.7	DROP ユーザーマッピング	38
7.7.1	パラメーター	38
7.7.2	例	38
7.8	外部テーブルをドロップ	38
7.8.1	パラメーター	39
7.8.2	例	39
第 8 章	Upgrading	41
8.1	RHEL、Rocky Linux、AlmaLinux または OL 8 の場合	41
8.2	SLES について	41
8.3	Debian または Ubuntu の場合	42
8.4	PPCLE の RHEL または CentOS 7 の場合	42
第 9 章	Uninstalling	43
第 10 章	End-to-end	45
第 11 章	Join pushdown	49
第 12 章	Aggregate function pushdown	51
第 13 章	LIMIT OFFSET pushdown	55
第 14 章	ORDER BY pushdown	59
第 15 章	WHERE clause pushdown	61
第 16 章	Identifying the version	63

MongoDB Foreign Data Wrapper (`mongo_fdw`) は、EDB Postgres Advanced Server から MongoDB データベースにあるデータにアクセスできるようにする Postgres の拡張機能です。これは、Postgres の関数とユーティリティ、または Postgres ホストにある他のデータで利用できる書き込み可能な外部データラッパーです。

RPM パッケージを使用して MongoDB Foreign Data Wrapper をインストールできます。

[EDB website](#) からインストーラーをダウンロードできます。

第1章 Release notes

Mongo Foreign Data Wrapper のドキュメントには、マイナーリリースとパッチを含む Mongo Foreign Data Wrapper 5 の最新バージョンが記載されています。リリースノートには、各リリースの新機能に関する情報が記載されています。マイナーリリースまたはパッチリリースで導入された新機能の場合、コンテンツにはその機能が導入されたリリースも示されます。

Version	Release date
<i>Version 5.5.0</i>	2023 Jan 06
<i>Version 5.4.0</i>	2022 May 26
<i>Version 5.3.0</i>	2021 Dec 02
<i>Version 5.2.9</i>	2021 Jun 24
<i>Version 5.2.8</i>	2020 Nov 23
<i>Version 5.2.6</i>	2019 Sep 27
<i>Version 5.2.3</i>	2018 Nov 01

1.1 Version 5.5.0

MongoDB Foreign Data Wrapper 5.5.0 の機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description
Feature	可能な場合、 <i>ORDER BY</i> 句をリモートの MongoDB サーバーにプッシュします。このアプローチは、外部サーバーからの順序付けられた結果セットを提供し、効率的なマージ結合に役立ちます。
Feature	可能な場合、リモートサーバーで <code>`LIMIT`</code> および <code>`OFFSET`</code> 操作を実行します。これにより、ローカルの Postgres とリモートの MongoDB サーバー間のネットワークトラフィックが削減されます。
Enhancement	リモート MongoDB サーバーへの <i>WHERE</i> 句のプッシュダウンを改善しました。演算子の両側で再帰演算子式、ブール式、リラベル タイプ、および <i>vars</i> をサポートするようになりました。
Bug Fix	ネストされた結合クエリの場合、 <code>`enable_aggregate_pushdown`</code> オプションのステータスを保存して、後で集約パスにアクセスします。
Bug Fix	<code>`Param`</code> ノードの処理がないためにサーバーがクラッシュする問題を修正しました。
Bug Fix	<code>`autogen.sh`</code> および <code>`README.md`</code> ファイルのタイプミスを修正しました。

1.2 Version 5.4.0

MongoDB Foreign Data Wrapper 5.4.0 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Feature	集計をリモートの MongoDB サーバーにプッシュダウンする: すべての行を取得してローカルに集計するのではなく、集計をリモートの MongoDB サーバーにプッシュします。これにより、アグリゲートをプッシュダウンできる場合に非常に優れたパフォーマンスが向上します。
Enhancement	行を推定する新しいサーバーオプション <code>use_remote_estimate</code> を追加します。
Bug Fix	NUMERIC 型の強制型修飾子。
Bug Fix	NUMERIC 型間のデータ型の非互換性を修正しました。
Bug Fix	テキスト配列要素でドキュメントを更新するときにサーバーがクラッシュする問題を修正しました。
Bug Fix	<code>_id</code> 列に <code>null</code> 値を指定して更新/削除中にサーバーがクラッシュする問題を修正しました。

1.3 Version 5.3.0

MongoDB Foreign Data Wrapper 5.3.0 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	EDB Postgres Advanced Server 14 のサポート。
Enhancement	結合プッシュダウン: クエリに同じリモートサーバーの 2 つの外部テーブル間の結合がある場合、両方のテーブルのすべての行をフェッチしてローカルで結合を実行する代わりに、その結合をリモートサーバーにプッシュできるようになりました。
Bug fix	API のパフォーマンスを向上させます。
Bug fix	全行参照のサポートが必要です。

1.4 Version 5.2.9

Mongo Foreign Data Wrapper 5.2.9 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description
Enhancement	mongo-c-driver を 1.17.3 に更新しました。
Enhancement	json-c を 0.15 に更新しました。
Enhancement	LICENSE ファイルを更新しました。
Bug fix	LEFT JOIN LATERAL を含むクエリで発生するクラッシュを修正しました。

次のページに続く

表 5 – 前のページからの続き

Type	Description
Bug fix	サーバーのクラッシュを回避するために、リモートリレーションからの PostgreSQL 固有のシステム属性の取得を制限します。
Bug fix	より多くの条件をリモートサーバーに送信できるように、WHERE プッシュダウンを改善しました。

1.5 Version 5.2.8

Mongo Foreign Data Wrapper 5.2.8 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

Type	Description
Enhancement	EDB Postgres Advanced Server 13 のサポート。
Enhancement	Ubuntu 20.04 LTS プラットフォームのサポート。
Enhancement	LICENSE ファイルを更新しました。
Bug fix	COPY FROM および/または外部パーティションのルーティング操作で発生するクラッシュを修正しました。クラッシュは、Mongo Foreign Data Wrapper がルーティング可能な外部テーブルパーティションをサポートしていないか、外部テーブルで COPY FROM を実行したことが原因でした。Mongo Foreign Data Wrapper は、クラッシュする代わりにエラーをスローするようになりました。
Bug fix	キャストターゲットリストが「NULL」を生成する問題を修正しました。明示的なキャストだけでなく、ターゲットリスト内の関数呼び出しまたは演算子に対しても正しい結果が返されます。
Bug fix	パラメーター化されたクエリが正しく動作するように ReScanForeignScan API を修正しました。副選択または相関クエリでパラメーター化されたプランが使用されるようになりました。
Bug fix	コンパイル警告を解決するために、サーバーポートオプションのタイプが int32 から int16 に変更されました。メタ ドライバー API は unsigned short 型のポート値を想定しているため、一部の gcc バージョンでコンパイル警告が発生しました。

1.6 Version 5.2.6

MongoDB Foreign Data Wrapper 5.2.6 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	EDB Postgres Advanced Server 12 のサポート。

1.7 Version 5.2.3

MongoDB Foreign Data Wrapper 5.2.3 の機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	EDB Postgres Advanced Server 11 のサポート。

第2章 Supported database versions

この表は、MongoDB Foreign Data Wrapper の最新バージョンと、サポートされている対応する EDB Postgres Advanced Server (EPAS) バージョンをリストしています。MongoDB Foreign Data Wrapper は、EDB Postgres Advanced Server と同じプラットフォームでサポートされています。詳細は [Product Compatibility](#) を参照してください。

MongoDB Foreign Data Wrapper	EPAS 14	EPAS 13	EPAS 12	EPAS 11
5.5.0	Y	Y	Y	Y
5.4.0	Y	Y	Y	Y
5.3.0	Y	Y	Y	Y
5.2.9	N	Y	Y	Y
5.2.8	N	Y	N	N
5.2.6	N	N	Y	N
5.2.3	N	Y	N	Y

第3章 Limitations

MongoDB Foreign Data Wrapper には次の制限が適用されます。

- BSON ドキュメントキーに大文字が含まれている場合、またはネストされたドキュメント内にある場合、MongoDB Foreign Data Wrapper は対応する列名を二重引用符で宣言する必要があります。
- PostgreSQL はデフォルトで列名を 63 文字に制限しています。列名が 63 文字を超える場合は、`src/include/pg_config_manual.h` の `NAMEDATALEN` 定数を増やし、コンパイル、および再インストールできます。
- MongoDB Foreign Data Wrapper は、既知のタイプ（バイト、配列など）にリストされていない BSON フィールドでエラーを返します。このエラーが返されます: `Cannot convert BSON type to column type`。

第4章 Architecture overview

MongoDB データラッパーは、MongoDB サーバーと Postgres データベース間のインターフェイスを提供します。Postgres ステートメント（SELECT /INSERT /DELETE /UPDATE ）を MongoDB データベースが理解するクエリに変換します。

! Using MongoDB FDW with Postgres

第5章 Key features

これらは、MongoDB Foreign Data Wrapper の主な機能です。

5.1 書き込み可能な FDW

MongoDB Foreign Data Wrapper を使用すると、MongoDB サーバー上のデータを変更できます。外部テーブルでローカルにデータを挿入、更新、および削除することにより、リモートの MongoDB コレクションのデータを挿入、更新、および削除できます。

詳細については、次を参照してください。

- [Example: Using the MongoDB Foreign Data Wrapper](#)
- [データ型のマッピング](#)

5.2 WHERE 句プッシュダウン

MongoDB Foreign Data Wrapper は、列と定数を引数として持つ比較式が句に含まれる場合にのみ、WHERE 句のプッシュダウンを許可します。定数が配列の場合、WHERE 句のプッシュダウンはサポートされていません。バージョン 5.5.0 以降、MongoDB Foreign Data Wrapper は、演算子の両側で再帰演算子式、ブール式、リラベルタイプ、および var をサポートします。

詳細については、[Example: WHERE clause pushdown](#) を参照してください。

5.3 参加プッシュダウン

MongoDB Foreign Data Wrapper は、内部結合、左結合、および右結合のプッシュダウンをサポートしています。現在、結合の失敗の可能性を回避するために、結合句に関係演算子と算術演算子のみが含まれる結合はプッシュダウンされます。

詳細については、[Example: Join pushdown](#) を参照してください。

5.4 アグリゲートプッシュダウン

MongoDB Foreign Data Wrapper は、集計プッシュダウンをサポートしています。すべての行を取得してローカルに集計する代わりに、集計をリモートの MongoDB サーバーにプッシュします。これにより、アグリゲートをプッシュダウンできる場合に非常に優れたパフォーマンスが向上します。プッシュダウンは現在、MongoDB サーバーに存在しない関数をプッシュダウンしないように、集計関数の `min`、`max`、`sum`、`avg`、および `count` に制限されています。集計フィルター、`orders`、`variadic`、`distinct` はプッシュダウンされません。

詳細については、[Example: Aggregate pushdown](#) を参照してください。

5.5 ORDER BY プッシュダウン

MongoDB Foreign Data Wrapper は `ORDER BY` プッシュダウンをサポートしています。可能であれば、`ORDER BY` 句をリモートサーバーにプッシュします。このアプローチは、外部サーバーからの順序付けられた結果セットを提供し、効率的なマージ結合に役立ちます。MongoDB サーバーでは `NULL` の動作は逆です。同等の結果を得るには、`ASC NULLS FIRST` または `DESC NULLS LAST` で `ORDER BY` を押し下げます。MongoDB はフィールドでのみソートするため、`ORDER BY` 式の列名のみがプッシュダウンされます。

詳細については、[Example: ORDER BY pushdown](#) を参照してください。

5.6 LIMIT OFFSET プッシュダウン

MongoDB Foreign Data Wrapper は `LIMIT /OFFSET` プッシュダウンをサポートしています。可能な限り、リモートサーバーで `LIMIT` および `OFFSET` 操作を実行します。これにより、ローカルの Postgres とリモートの MongoDB サーバー間のネットワークトラフィックが削減されます。

詳細については、[Example: LIMIT OFFSET pushdown](#) を参照してください。

5.7 接続プーリング

MongoDB Foreign Data Wrapper は、外部サーバーに関連付けられた外部テーブルを使用する最初のクエリ中に、外部サーバーへの接続を確立します。この接続は保持され、同じセッションの後続のクエリで再利用されます。

5.8 自動クリーンアップ

MongoDB Foreign Data Wrapper を使用すると、`DROP EXTENSION` コマンドを使用して 1 回のオペレーションで外部テーブルをクリーンアップできます。この機能は、外部テーブルが一時的な目的で作成された場合に特に役立ちます。`DROP EXTENSION` コマンドの構文は次のとおりです。

```
DROP EXTENSION mongo_fdw CASCADE;
```

詳細については、[ドロップエクステンション](#) を参照してください。

5.9 全文書取得

この機能を使用すると、MongoDB のコレクションで使用可能な BSON ドキュメントのフィールドを知らなくても、コレクションからドキュメントとそのすべてのフィールドを取得できます。これらの取得されたドキュメントは JSON 形式です。

次の例で説明するように、MongoDB Foreign Data Wrapper にあるコレクションで使用可能なすべてのフィールドを取得できます。

5.9.1 例

```
> db.warehouse.find();
{ "_id" : ObjectId("58a1ebbf543ec0b90545859"), "warehouse_id" : 1, "warehouse_name" ↵
↵: "UPS", "warehouse_created" : ISODate("2014-12-12T07:12:10Z") }
{ "_id" : ObjectId("58a1ebbf543ec0b9054585a"), "warehouse_id" : 2, "warehouse_name" ↵
↵: "Laptop", "warehouse_created" : ISODate("2015-11-11T08:13:10Z") }
```

ドキュメントを取得する手順：

1. 列名が `__doc` の外部テーブルを作成します。列のタイプは、`json`、`jsonb`、`text`、または `varchar` です。

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

2. ドキュメントを取得します。

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

出力：

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

```

                                     __doc
-----
↵-----
↵-----
```

(次のページに続く)

(前のページからの続き)

```
{ "_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)
```

第6章 Installing

リンクを選択して、該当するインストール手順にアクセスします。

Linux x86-64 (amd64)

6.1 Red Hat Enterprise Linux (RHEL) およびその派生物

- *Installing MongoDB Foreign Data Wrapper on RHEL 8 or OL 8 x86_64*
- *Installing MongoDB Foreign Data Wrapper on RHEL 7 or OL 7 x86_64*
- Oracle Linux (OL) 8
- Oracle Linux (OL) 7
- *Installing MongoDB Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64*
- *Installing MongoDB Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64*
- *Installing MongoDB Foreign Data Wrapper on CentOS 7 x86_64*

6.2 SUSE Linux Enterprise (SLES) とその派生

- *Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 12 x86_64*

6.3 Debian とその派生物

- *Installing MongoDB Foreign Data Wrapper on Ubuntu 20.04 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Ubuntu 18.04 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Debian 11 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Debian 10 x86_64*

Linux on IBM Power (ppc64le)

6.4 Red Hat Enterprise Linux (RHEL)

- *Installing MongoDB Foreign Data Wrapper on RHEL 8 or OL 8 x86_64*

6.5 SUSE Linux Enterprise (SLES)

- *Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 12 x86_64*

6.5.1 Installing MongoDB Foreign Data Wrapper on Linux x86 (amd64)

オペレーティングシステム固有のインストール手順については、次を参照してください。

- RHEL 8/OL 8
- Rocky Linux 8/AlmaLinux 8
- RHEL 7/OL 7
- *Installing MongoDB Foreign Data Wrapper on CentOS 7 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 12 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Ubuntu 20.04 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Ubuntu 18.04 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Debian 11 x86_64*
- *Installing MongoDB Foreign Data Wrapper on Debian 10 x86_64*

インストールが完了したら、 *Initial Configuration* を参照してください。

Installing MongoDB Foreign Data Wrapper on RHEL 8 or OL 8 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

EDB repositories に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Install the EPEL repository:
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.
↳noarch.rpm

# Disable the built-in PostgreSQL module:
sudo dnf -qy module disable postgresql
```

パッケージをインストールする

```
sudo dnf -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on RHEL 7 or OL 7 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに EDB Postgres Advanced Server をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

EDB repositories に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Install the EPEL repository:
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
↳noarch.rpm

# Enable additional repositories to resolve dependencies:
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-
↳rpms" --enable "rhel-ha-for-rhel-*-server-rpms"
```

パッケージをインストールする

```
sudo yum -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on AlmaLinux 8 or Rocky Linux 8 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。

- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Install the EPEL repository:
sudo dnf -y install epel-release
# Enable additional repositories to resolve dependencies:
sudo dnf config-manager --set-enabled PowerTools
# Disable the built-in PostgreSQL module:
sudo dnf -qy module disable postgresql
```

パッケージをインストールする

```
sudo dnf -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on CentOS 7 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

EDB repositories に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Install the EPEL repository:  
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.  
↪noarch.rpm
```

パッケージをインストールする

```
sudo yum -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

EDB repositories に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/15.3/x86_64

# Refresh the metadata
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on SLES 12 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/12.5/x86_64
sudo SUSEConnect -p sle-sdk/12.5/x86_64

# Refresh the metadata
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on Ubuntu 20.04 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on Ubuntu 18.04 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on Debian 11 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on Debian 10 x86_64

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

パッケージをインストールする

```
sudo apt-get -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

6.5.2 Installing MongoDB Foreign Data Wrapper on IBM Power (ppc64le)

オペレーティングシステム固有のインストール手順については、次を参照してください。

- *Installing MongoDB Foreign Data Wrapper on RHEL 8 or OL 8 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 15 x86_64*
- *Installing MongoDB Foreign Data Wrapper on SLES 12 x86_64*

インストールが完了したら、*Initial Configuration* を参照してください。

Installing MongoDB Foreign Data Wrapper on RHEL 8 ppc64le

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

EDB repositories に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# 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

# Disable the built-in PostgreSQL module:
sudo dnf -qy module disable postgresql
```

パッケージをインストールする

```
sudo dnf -y install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on SLES 15 ppc64le

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。

- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/15.3/ppc64le

# Refresh the metadata
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

Installing MongoDB Foreign Data Wrapper on SLES 12 ppc64le

前提条件

インストールプロセスを開始する前に：

- 同じホストに **EDB Postgres Advanced Server** をインストールします。
Installing EDB Postgres Advanced Server を参照してください。
- リポジトリをセットアップする

リポジトリの設定は 1 回限りのタスクです。リポジトリを既に設定している場合、この手順を実行する必要はありません。

リポジトリをセットアップするには、

[EDB repositories](#) に移動し、そこに表示される指示に従います。

- その他の前提条件に対処する

```
# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/12.5/ppc64le
sudo SUSEConnect -p sle-sdk/12.5/ppc64le

# Refresh the metadata
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-as14-mongo-fdw
```

14 は EDB Postgres Advanced Server のバージョンです。14 を、使用している EDB Postgres Advanced Server のバージョンに置き換えます。

第7章 Initial Configuration

MongoDB Foreign Data Wrapper を使用する前に：

1. **拡張機能を作成する** コマンドを使用して、Postgres ホストで MongoDB Foreign Data

Wrapper 拡張機能を作成します。 2.

サーバーを作成する コマンドを使用して、MongoDB サーバーへの接続を定義します。

3. **ユーザーマッピングを作成する** コマンドを使用して、Postgres ロールをサーバーに関連付けるマッピングを定義します。

4. **外部テーブルを作成する** コマンドを使用して、MongoDB クラスターにあるデータベースに対応する Postgres データベースにテーブルを定義します。

7.1 拡張機能を作成する

CREATE EXTENSION コマンドを使用して、`mongo_fdw` 拡張機能を作成します。コマンドを呼び出すには、選択したクライアント（たとえば、psql）を使用して、MongoDB サーバーを照会する Postgres データベースに接続し、コマンドを呼び出します。

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

7.1.1 パラメーター

IF NOT EXISTS

IF NOT EXISTS 句を含めて、同じ名前の拡張機能が既に存在する場合にエラーを返す代わりに通知を発行するようにサーバーに指示します。

schema_name

オプションで、拡張機能のオブジェクトをインストールするスキーマの名前を指定します。

7.1.2 例

次のコマンドは、MongoDB 外部データラッパーをインストールします。

```
CREATE EXTENSION mongo_fdw;
```

外部データラッパー **CREATE EXTENSION** コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

7.2 サーバーを作成する

CREATE SERVER コマンドを使用して、外部サーバーへの接続を定義します。構文は次のとおりです。

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

サーバーを定義するロールは、サーバーの所有者です。 **ALTER SERVER** コマンドを使用して、外部サーバーの所有権を再割り当てします。外部サーバーを作成するには、**CREATE SERVER** コマンドで指定された外部データラッパーに対する **USAGE** 特権が必要です。

7.2.1 パラメーター

server_name

server_name を使用して、外部サーバーの名前を指定します。サーバー名はデータベース内で一意である必要があります。

FOREIGN_DATA_WRAPPER

FOREIGN_DATA_WRAPPER 句を含めて、サーバーがクラスターに接続するときに **mongo_fdw** 外部データラッパーを使用するように指定します。

OPTIONS

CREATE SERVER コマンドの **OPTIONS** 句を使用して、外部サーバーオブジェクトの接続情報を指定します。これらのオプションを含めることができます。

Option	Description
address	Mongo サーバーのアドレスまたはホスト名。デフォルト値は `127.0.0.1` です。
port	Mongo サーバーのポート番号。有効な範囲は 0～65535 です。デフォルト値は `27017` です。
authentication_database	ユーザーが認証されるデータベース。このオプションは、パスワードベースの認証でのみ有効です。
ssl	認証された暗号化された SSL 接続を要求します。デフォルトでは、値は `false` に設定されます。値を <i>true</i> に設定して、SSL を有効にします。オプションを理解するには、 mongoc_ssl_opt_t を参照してください。

次のページに続く

表 1 – 前のページからの続き

Option	Description
pem_file	SSL オプション。
pem_pwd	SSL オプション。
ca_file	SSL オプション。
ca_dir	SSL オプション。
crl_file	SSL オプション。
weak_cert_validation	SSL オプション。デフォルト値は `false` です。
enable_join_pushdown	テーブルレベルのオプションに似ていますが、サーバーレベルで構成されます。`true` の場合、両方のテーブルのすべての行をフェッチしてローカルで結合を実行する代わりに、同じ外部サーバーから 2 つの外部テーブル間の結合をプッシュします。このオプションは個々のテーブルに設定することもできます。この場合、結合に関係するテーブルのいずれかでオプションが `false` に設定されていると、結合はプッシュダウンされません。オプションのテーブルレベルの値は、サーバーレベルのオプション値よりも優先されます。デフォルトは `true` です。

7.2.2 例

次のコマンドは、`mongo_fdw` 外部データラッパーを使用して `127.0.0.1` の IP アドレスを持つホストに接続する `mongo_server` という名前の外部サーバーを作成します。

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

外部サーバーは、MongoDB クラスター上のクライアントへの接続にデフォルトのポート（`27017`）を使用します。

`CREATE SERVER` コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

7.3 ユーザーマッピングを作成する

`CREATE USER MAPPING` コマンドを使用して、Postgres ロールを外部サーバーに関連付けるマッピングを定義します。

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

そのサーバーのユーザーマッピングを作成するには、そのサーバーの所有者である必要があります。

7.3.1 パラメーター

`role_name`

`role_name` を使用して、外部サーバーに関連付けるロールを指定します。

`server_name`

`server_name` を使用して、MongoDB クラスターへの接続を定義するサーバーの名前を指定します。

`OPTIONS`

`OPTIONS` 句を使用して、外部サーバーの接続情報を指定します。

`username` は、MongoDB サーバーのユーザーの名前です。

`password` は、ユーザー名に関連付けられたパスワードです。

7.3.2 例

次のコマンドは、`enterprisedb` という名前のロールのユーザーマッピングを作成します。マッピングは、`mongo_server` という名前のサーバーに関連付けられています。

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

データベースホストが安全な認証を使用する場合、ユーザーマッピングを作成するときに接続資格情報を提供します。

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

このコマンドは、`mongo_server` という名前のサーバーに関連付けられている `enterprisedb` という名前のロールのユーザーマッピングを作成します。MongoDB サーバーに接続すると、サーバーは `mongo_user` として認証し、パスワード `mongo_pass` を提供します。

`CREATE USER MAPPING` コマンドの詳細については、[PostgreSQL documentation](#) を参照してください。

7.4 外部テーブルを作成する

外部テーブルは、MongoDB ホストにあるテーブルへのポインターです。Postgres サーバーで外部テーブル定義を作成する前に、MongoDB サーバーに接続してコレクションを作成します。テーブルの列は、Postgres サーバー上のテーブルの列にマップされます。次に、`CREATE FOREIGN TABLE` コマンドを使用して、MongoDB ホストにあるコレクションに対応する列を持つテーブルを Postgres サーバーに定義します。構文は次のとおりです。

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

(次のページに続く)

(前のページからの続き)

```
[, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
SERVER server_name [ OPTIONS ( option value [, ... ] ) ]
```

column_constraint は次のとおりです。

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

table_constraint は次のとおりです。

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

7.4.1 パラメーター

table_name

外部テーブルの名前を指定します。スキーマ名を含めて、外部テーブルが存在するスキーマを指定します。

IF NOT EXISTS

IF NOT EXISTS 句を含めて、同じ名前のテーブルが既に存在する場合にエラーを返さないようにサーバーに指示します。同名のテーブルが存在する場合、サーバーは通知を発行します。

column_name

新しいテーブルの列の名前を指定します。各列は、MongoDB サーバーで説明されている列に対応している必要があります。

data_type

列のデータ型を指定します。可能な場合、Postgres サーバーと MongoDB サーバーの各列に同じデータ型を指定します。同じ名前のデータ型が利用できない場合、Postgres サーバーはデータ型を MongoDB サーバーと互換性のある型にキャストしようとします。サーバーが互換性のあるデータ型を識別できない場合、エラーを返します。

COLLATE collation

COLLATE 句を含めて、列に照合を割り当てます。指定しない場合、列のデータ型のデフォルトの照合が使用されます。

INHERITS (parent_table [, ...])

INHERITS 句を含めて、新しい外部テーブルがすべての列を継承するテーブルのリストを指定します。親テーブルは、プレーンテーブルまたは外部テーブルです。

CONSTRAINT constraint_name

列またはテーブルの制約のオプションの名前を指定します。指定しない場合、サーバーは制約名を生成します。

NOT NULL

NOT NULL キーワードを含めて、列に null 値を含めることができないことを示します。

NULL

NULL キーワードを含めて、列に NULL 値を含めることが許可されていることを示します。これがデフォルトです。

CHECK (expr) [NO INHERIT]

CHECK 句を使用して、テーブルの各行が満たす必要があるブール結果を生成する式を指定します。列制約として指定されたチェック制約はその列の値のみを参照する必要がありますが、テーブル制約に表示される式は複数の列を参照できます。

CHECK 式には、サブクエリを含めたり、現在の行の列以外の変数を参照したりできません。

NO INHERIT キーワードを含めて、制約が子テーブルに伝播できないことを指定します。

DEFAULT default_expr

DEFAULT 句を含めて、列定義が表示される列のデフォルトのデータ値を指定します。デフォルトの式のデータ型は、列のデータ型と一致する必要があります。

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

MongoDB ファイルシステムにあるテーブルを照会できる外部テーブルを作成するには、SERVER 句を含め、MongoDB データアダプターを使用する外部サーバーに server_name を指定します。

OPTIONS 句を使用して、次のオプションとそれに対応する値を指定します。

Option	Value
database	The name of the database to query. The default value is <i>test</i> .
collection	The name of the collection to query. The default value is the foreign table name.
enable_join_pushdown	Similar to the server-level option but configured at table level. If <i>true</i> , 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 <i>false</i> , then the join isn't pushed down. The table-level value of the option takes precedence over the server-level option value. Default is <i>true</i> .

7.4.2 例

MongoDB サーバーに保存されているデータを使用するには、MongoDB コレクションの列を Postgres テーブルの列にマップするテーブルを作成する必要があります。たとえば、次の定義を持つ MongoDB コレクションの場合。

```
db.warehouse.find
(
  {
```

(次のページに続く)

(前のページからの続き)

```

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

```

Postgres サーバーで同等のテーブルを作成するコマンドを実行します。

```

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

```

テーブルの最初の列は、タイプ `name` の `_id` でなければなりません。

`SERVER` 句を含めて、MongoDB サーバーに保存されているデータベースの名前と Postgres サーバー上のテーブルに対応するテーブル (`warehouse`) の名前を指定します。

`CREATE FOREIGN TABLE` コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

注釈: MongoDB Foreign Data Wrapper は書き込み機能機能をサポートしています。

7.4.3 データ型のマッピング

外部データラッパーを使用する場合、MongoDB サーバーにあるテーブルをミラーリングするテーブルを Postgres サーバーに作成する必要があります。MongoDB データラッパーは、次の MongoDB データ型をターゲットの Postgres 型に変換します。

MongoDB (BSON Type)	Postgres
ARRAY	JSON
BOOL	BOOL
BINARY	BYTEA

次のページに続く

表 3 – 前のページからの続き

MongoDB (BSON Type)	Postgres
DATE_TIME	DATE/TIMESTAMP/TIMESTAMPZ
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

7.5 ドロップエクステンション

DROP EXTENSION コマンドを使用して、拡張機能を削除します。コマンドを呼び出すには、選択したクライアント（たとえば、psql）を使用して、MongoDB サーバーを削除する Postgres データベースに接続し、コマンドを実行します。

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

7.5.1 パラメーター

IF EXISTS

IF EXISTS 句を含めて、指定された名前の拡張機能が存在しない場合にエラーを返す代わりに通知を発行するようにサーバーに指示します。

name

必要に応じて、インストールされている拡張機能の名前を指定します。

CASCADE

拡張機能に依存するオブジェクトを削除します。他のすべての依存オブジェクトも削除します。

RESTRICT

同じ DROP コマンドにリストされているメンバーオブジェクトと拡張機能以外のオブジェクトが依存している場合、拡張機能のドロップを許可しないでください。

7.5.2 例

次のコマンドは、既存のデータベースから拡張機能を削除します。

```
DROP EXTENSION mongo_fdw;
```

外部データラッパー **DROP EXTENSION** コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

7.6 ドロップサーバー

DROP SERVER コマンドを使用して、外部サーバーへの接続を削除します。構文は次のとおりです。

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

サーバーを削除するロールはサーバーの所有者です。 **ALTER SERVER** コマンドを使用して、外部サーバーの所有権を再割り当てします。外部サーバーを削除するには、**DROP SERVER** コマンドで指定された外部データラッパーに対する **USAGE** 権限が必要です。

7.6.1 パラメーター

IF EXISTS

IF EXISTS 句を含めて、指定された名前のサーバーが存在しない場合にエラーを返す代わりに通知を発行するようにサーバーに指示します。

name

オプションで、インストールされたサーバーの名前を指定します。

CASCADE

サーバーに依存するオブジェクトを削除します。他のすべての依存オブジェクトも削除します。

RESTRICT

サーバーに依存しているオブジェクトがある場合は、サーバーを削除しないでください。

7.6.2 例

次のコマンドは、**mongo_server** という名前の外部サーバーを削除します。

```
DROP SERVER mongo_server;
```

DROP SERVER コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

7.7 DROP ユーザーマッピング

DROP USER MAPPING コマンドを使用して、Postgres ロールを外部サーバーに関連付けるマッピングを削除します。そのサーバーのユーザーマッピングを削除するには、そのサーバーの所有者である必要があります。

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

7.7.1 パラメーター

IF EXISTS

ユーザーマッピングが存在しない場合にエラーを返す代わりに通知を発行するようにサーバーに指示するには、IF EXISTS 句を含めます。

user_name

マッピングのユーザー名を指定します。

server_name

MongoDB クラスターへの接続を定義するサーバーの名前を指定します。

7.7.2 例

次のコマンドは、enterprisedb という名前のロールのユーザーマッピングを削除します。マッピングは、mongo_server という名前のサーバーに関連付けられています。

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

DROP USER MAPPING コマンドの詳細については、[PostgreSQL documentation](#) を参照してください。

7.8 外部テーブルをドロップ

外部テーブルは、MongoDB ホストにあるテーブルへのポインターです。DROP FOREIGN TABLE コマンドを使用して、外部テーブルを削除します。外部テーブルの所有者のみが削除できます。

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

7.8.1 パラメーター

IF EXISTS

IF EXISTS 句を含めて、指定された名前の外部テーブルが存在しない場合にエラーを返す代わりに通知を発行するようにサーバーに指示します。

name

外部テーブルの名前を指定します。

CASCADE

外部テーブルに依存するオブジェクトを削除します。他のすべての依存オブジェクトも削除します。

RESTRICT

オブジェクトが依存している場合、外部テーブルを削除しないでください。

7.8.2 例

```
DROP FOREIGN TABLE warehouse;
```

DROP FOREIGN TABLE コマンドの使用の詳細については、[PostgreSQL documentation](#) を参照してください。

第8章 Upgrading

EDB リポジトリを使用してインストールした MongoDB Foreign Data Wrapper の既存のインストールがある場合、**upgrade** コマンドを使用してリポジトリ構成ファイルを更新し、より新しい製品バージョンにアップグレードできます。プロセスを開始するには、ターミナルウィンドウを開き、スーパーユーザ権限を引き受けて、インストールに使用するオペレーティングシステムとパッケージマネージャーに適用可能なコマンドを入力します。

8.1 RHEL、Rocky Linux、AlmaLinux または OL 8 の場合

edb_notranlate_0 ## RHEL または CentOS または OL 7 の場合

```
# Update your edb.repo file to access the current EDB repository

yum upgrade edb-repo

# Upgrade to the latest version product version

yum upgrade edb-as<xx>-mongo_fdw edb-libmongoc-libs
# where <xx> is the EDB Postgres Advanced Server version number
```

8.2 SLES について

```
# Update your edb.repo file to access the current EDB repository

zypper upgrade edb-repo

# Upgrade to the latest version product

zypper upgrade edb-as<xx>-mongo_fdw
# where <xx> is the EDB Postgres Advanced Server version number
```

8.3 Debian または Ubuntu の場合

```
# Update your edb.repo file to access the current EDB repository

zypper upgrade edb-repo

# Upgrade to the latest version product

apt-get --only-upgrade install edb-as<xx>-mongo-fdw edb-libmongoc
# where <xx> is the EDB Postgres Advanced Server version number
```

8.4 PPCLE の RHEL または CentOS 7 の場合

```
# Update your edb.repo file to access the current EDB repository

yum upgrade edb-repo

# Upgrade to the latest version product version

yum upgrade edb-as<xx>-mongo_fdw edb-libmongoc-at<yy>-libs

# where:

# <xx> is the EDB Postgres Advanced Server version number

# <yy> is Advance Toolchain major version number. For EDB Postgres

# Advanced Server versions 10 to 11, <yy> must be 10 and for

# EDB Postgres Advanced Server version 12 and later, <yy> must be 11.
```

第9章 Uninstalling

`remove` コマンドを使用して、MongoDB Foreign Data Wrapper パッケージをアンインストールできます。アンインストールするには、ターミナルウィンドウを開き、スーパーユーザー権限を引き受けて、インストールに使用するオペレーティングシステムとパッケージマネージャーに適用するコマンドを入力します。`xx` は EDB Postgres Advanced Server のバージョン番号です。

- RHEL または CentOS 7 の場合：

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

- RHEL または Rocky Linux または AlmaLinux 8 の場合：

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

- SLES の場合：

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

- Debian または Ubuntu の場合

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


第10章 End-to-end

MongoDB 外部データラッパーを使用する前に、クライアントアプリケーションでデータベースに接続する必要があります。次の例では、psql クライアントでラッパーを使用しています。psql に接続した後、例の手順に従うことができます。

```
- - 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 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)
```

(次のページに続く)

(前のページからの続き)

```
db.warehouse.find
(
    {
        "warehouse_id" : 1
    }
).pretty()
{
    "_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
```

(次のページに続く)

(前のページからの続き)

```
    },
    {
      "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;
```

```
- - drop foreign table
```

```
DROP FOREIGN TABLE warehouse;
```

```
- - drop user mapping
```

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

```
- - drop server
```

```
DROP SERVER mongo_server;
```


第11章 Join pushdown

MongoDB Foreign Data Wrapper は、内部結合、左結合、および右結合のプッシュダウンをサポートしています。例：

Postgres データセット：

```
- - 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);
```

出力：

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

(次のページに続く)

(前のページからの続き)

__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)
```

第12章 Aggregate function pushdown

MongoDB Foreign Data Wrapper は、次の集計関数のプッシュダウンをサポートしています。

- AVG - 一連の値の平均を計算します。
- COUNT-指定されたテーブルまたはビューの行をカウントします。
- MIN - 値のセットの最小値を取得します。
- MAX - 値のセットの最大値を取得します。
- SUM - 値の合計を計算します。

Postgres データセット：

```
- - 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);
```

出力：

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

          QUERY PLAN
```

(次のページに続く)

(前のページからの続き)

```
-----
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)
```

(次のページに続く)

(前のページからの続き)

(3 rows)

```
- - 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)

第13章 LIMIT OFFSET pushdown

この例は、EMP 表の LIMIT OFFSET プッシュダウンを示しています。

Postgres データセット：

```
- - 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
-----
```

(次のページに続く)

```

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
-----+-----
 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

```

(前のページからの続き)

```
-----  
Limit  
  Output: eid, (max(eid))  
-> Foreign Scan  
    Output: eid, (max(eid))  
    Foreign Namespace: Aggregate on ("FDW_134".emp)  
(5 rows)  
  
edb=#
```


第14章 ORDER BY pushdown

この例は、EMP 表での ORDER BY プッシュダウンを示しています。

Postgres データセット：

```
- - 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)
```

```
edb=#
edb=# EXPLAIN (VERBOSE, COSTS FALSE)
edb=# SELECT eid, sum(eid), count(*) FROM emp GROUP BY eid HAVING min(eid) > 100 ORDER
```

(次のページに続く)

```

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)

```

第15章 WHERE clause pushdown

MongoDB Foreign Data Wrapper は、WHERE 句のプッシュダウンをサポートしています。例：

Postgres データセット：

```
- - 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);
```

出力：

```
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=#
```

(次のページに続く)

(前のページからの続き)

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

第16章 Identifying the version

MongoDB Foreign Data Wrapper には、データラッパー用に現在インストールされている .so ファイルのバージョンを識別するために使用できる関数が含まれています。この機能を使用するには、Postgres サーバーに接続し、次のように入力します。

```
SELECT mongo_fdw_version();
```

この関数はバージョン番号を返します。

```
mongo_fdw_version  
--  
<xxxxxx>
```