



MySQL Foreign Data Wrapper

リリース 2.9.0

EnterpriseDB Corporation

2023 年 01 月 26 日

目次

第 1 章	Release notes	3
1.1	Version 2.9.0	3
1.2	Version 2.8.0	4
1.3	Version 2.7.0	4
1.4	Version 2.6.0	5
1.5	Version 2.5.5	5
1.6	Version 2.5.3	5
1.7	Version 2.5.1	5
第 2 章	Supported database and versions	7
2.1	サポートされているデータベースバージョン	7
2.2	サポートされている MySQL のバージョン	7
第 3 章	Architecture overview	9
第 4 章	Key features	11
4.1	書き込み可能な外部データラッパー	11
4.2	接続プーリング	11
4.3	WHERE 句プッシュダウン	11
4.4	列プッシュダウン	12
4.5	参加プッシュダウン	12
4.6	アグリゲートプッシュダウン	12
4.7	ORDER BY プッシュダウン	12
4.8	LIMIT OFFSET プッシュダウン	13
4.9	プッシュダウンを制限する設定ファイル	13
4.10	IS [NOT] DISTINCT FROM 演算子	14
4.11	準備されたステートメント	14
4.12	外部スキーマのインポート	14
4.13	自動クリーンアップ	14
4.14	テーブルを切り捨てる	14
第 5 章	Installing	15
5.1	Red Hat Enterprise Linux (RHEL) およびその派生物	15
5.2	SUSE Linux Enterprise (SLES) とその派生	15
5.3	Debian とその派生物	15
5.4	Red Hat Enterprise Linux (RHEL)	16
5.5	SUSE Linux Enterprise (SLES)	16
5.5.1	Installing MySQL Foreign Data Wrapper on Linux x86 (amd64)	16
5.5.2	Installing MySQL Foreign Data Wrapper on IBM Power (ppc64le)	26
第 6 章	Initial configuration	29

6.1	拡張機能を作成する	29
6.1.1	パラメーター	29
6.1.2	例	30
6.2	サーバーを作成する	30
6.2.1	パラメーター	30
6.2.2	例	31
6.3	ユーザーマッピングを作成する	32
6.3.1	パラメーター	32
6.3.2	例	32
6.4	外国のテーブルを作成する	33
6.4.1	パラメーター	33
6.4.2	例	35
6.4.3	データ型のマッピング	35
6.5	外国人スキーマのインポート	36
6.5.1	パラメーター	36
6.5.2	例	37
6.6	ドロップエクステンション	38
6.6.1	パラメーター	38
6.6.2	例	39
6.7	ドロップサーバー	39
6.7.1	パラメーター	39
6.7.2	例	39
6.7.3	パラメーター	40
6.7.4	例	40
6.8	外国のテーブルをドロップ	40
6.8.1	パラメーター	40
6.8.2	例	41
第 7 章	Upgrading	43
7.1	RPM インストールのアップグレード	43
7.2	Debian または Ubuntu ホストでの MySQL Foreign Data Wrapper のアップグレード	43
第 8 章	Uninstalling	45
第 9 章	End-to-end	47
第 10 章	Import foreign schema	49
第 11 章	Join pushdown	51
第 12 章	Aggregate pushdown	55
第 13 章	ORDER BY pushdown	59
第 14 章	LIMIT OFFSET pushdown	61
第 15 章	Identifying the version	63
第 16 章	Authentication plugin ‘caching_sha2_password’ error	65

MySQL Foreign Data Wrapper (`mysql_fdw`) は、MySQL データベースにあるデータに EDB Postgres Advanced Server からアクセスできるようにする Postgres 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 MySQL 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。

何

第1章 Release notes

MySQL 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 MySQL 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、機能を導入したリリースに関するインジケータも含まれています。

Version	Release Date
<i>Version 2.9.0</i>	2023 Jan 06
<i>Version 2.8.0</i>	2022 May 26
<i>Version 2.7.0</i>	2021 Dec 02
<i>Version 2.6.0</i>	2021 May 18
<i>Version 2.5.5</i>	2020 Nov 23
<i>Version 2.5.3</i>	2019 Dec 10
<i>Version 2.5.1</i>	2018 Nov 28

1.1 Version 2.9.0

MySQL Foreign Data Wrapper 2.9.0 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	プッシュダウンを制御する構成ファイルのアプローチを追加します。これで、リモートサーバーに安全にプッシュするオペレーターと関数のリストを提供できます。
Feature	<code>`IMPORT FOREIGN SCHEMA`</code> に <code>`import_generated`</code> オプションのサポートを追加して、外部サーバーからインポートされた外部テーブルの定義に式から生成された列を含めます。
Feature	外部テーブルを切り詰める <code>`TRUNCATE`</code> コマンドのサポートを追加します。
Feature	<code>`INSERT`</code> に <code>`ON CONFLICT DO NOTHING`</code> のサポートを追加。 <code>`INSERT IGNORE`</code> へのマッピングです。
Feature	<code>`IS [ NOT ] DISTINCT FROM`</code> 演算子のサポートを追加します。
Bug Fix	プッシュダウン <code>`ORDER BY`</code> 句の評価の見落としを修正しました。 <code>`query_pathkeys`</code> の下にプッシュするのが安全でない場合はプッシュしないでください。
Bug Fix	<code>`SQL/select`</code> テストの不安定な順序付けを修正しました。
Bug Fix	<code>`OFFSET`</code> が <code>`NULL`</code> の場合は <code>`LIMIT n`</code> 句をプッシュダウンします。

1.2 Version 2.8.0

MySQL Foreign Data Wrapper 2.8.0 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	ORDER BY をリモート MySQL サーバーにプッシュダウン：可能であれば、リモート SQL に ORDER BY 句を追加して、外部サーバーから順序付けられた結果セットを取得するパスを検討します。これは、効率的なマージ結合に役立ちます。
Feature	LIMIT/OFFSET をリモート MySQL サーバーにプッシュダウン：該当する場合、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 PostgreSQL とリモートの MySQL 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	character_set オプションを追加。ユーザーは MySQL データベースごとに適切な値に設定できるようになりました。
Enhancement	sql_mode オプションを追加します。MySQL サーバーはさまざまな SQL モードで動作できます。このオプションにより、それに合わせます。
Bug Fix	import_enum_as_text IMPORT FOREIGN SCHEMA オプションを導入します。true の場合、mysql_fdw は ENUM カラムを TEXT 型にマッピングして、テーブルを失敗せずにインポートします。
Bug Fix	MySQL 値を PostgreSQL に変換するときに正しい型修飾子を強制します。
Bug Fix	非常に長いデータを読み取るときのテキスト列データの切り捨てを修正します。
Bug Fix	MySQL は VARIADIC 配列をサポートしていないため、集計をプッシュしないでください。

1.3 Version 2.7.0

MySQL Foreign Data Wrapper 2.7.0 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
Enhancement	EDB Postgres Advanced Server 14 のサポート。
Enhancement	MySQL Foreign Data Wrapper は、min、max、sum、avg、および count の集計関数のプッシュダウンをサポートするようになりました。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 MySQL サーバーにプッシュできるようになりました。これにより、アグリゲートをプッシュダウンできる場合のパフォーマンスが向上します。集計フィルターとオーダーは MySQL でサポートされていないため、プッシュダウンされません。
Bug fix	MySQL サーバーに間違ったクエリを送信する暗黙的/明示的な強制を解析する関数式。
Bug fix	SET タイプのため、外部スキーマのインポートに失敗しました。

1.4 Version 2.6.0

MySQL Foreign Data Wrapper 2.6.0 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	リモート MySQL サーバーへのプッシュダウン参加のサポート。
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。

1.5 Version 2.5.5

MySQL Foreign Data Wrapper 2.5.5 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 13 のサポート。
Enhancement	Ubuntu 20.04 LTS プラットフォームのサポート。

1.6 Version 2.5.3

MySQL Foreign Data Wrapper 2.5.3 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 12 のサポート。

1.7 Version 2.5.1

MySQL Foreign Data Wrapper 2.5.1 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 and versions

2.1 サポートされているデータベースバージョン

この表は、MySQL 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。MySQL 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。

MySQL Foreign Data Wrapper	EPAS 14	EPAS 13	EPAS 12	EPAS 11
2.9.0	Y	Y	Y	Y
2.8.0	Y	Y	Y	Y
2.7.0	Y	Y	Y	Y
2.6.0	N	Y	Y	Y
2.5.5	N	Y	N	N
2.5.3	N	N	Y	N
2.5.1	N	N	N	Y

2.2 サポートされている MySQL のバージョン

MySQL Foreign Data Wrapper は、次のプラットフォームの MySQL 8 でサポートされています。

Linuxx86-64

- RHEL 8/OL 8 - Rocky Linux 8/AlmaLinux 8 - RHEL 7/OL 7 - CentOS 7 - SLES 15 - SLES 12 - Ubuntu 20.04/18.04 LTS - Debian 11/10.x

IBMPower 上の Linux (ppc64le)

- RHEL 8
- SLES 15
- SLES 12

MySQL Foreign Data Wrapper は、次のプラットフォームの MySQL 5 でサポートされています。

Linuxx86-64

- RHEL 7/OL 7 - CentOS 7 - SLES 12 - Ubuntu 18.04 LTS

第3章 Architecture overview

MySQL データラッパーは、MySQL 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 MySQL データベースが理解するクエリに変換します。

! Using mysql_fdw with Postgres

第4章 Key features

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

4.1 書き込み可能な外部データラッパー

MySQL Foreign Data Wrapper は書き込み機能を提供します。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 MySQL テーブルのデータを挿入、更新、および削除できます。MySQL 外部データラッパーは、Postgres 型キャストメカニズムを使用して、MySQL と Postgres データ型の間で反対の型キャストを提供します。

注釈: DML が機能するには、MySQL テーブルの最初の列に一意/主キーが必要です。

次も参照してください。

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

4.2 接続プーリング

MySQL 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。

4.3 WHERE 句プッシュダウン

MySQL Foreign Data Wrapper を使用すると、実行のために **WHERE** 句を外部サーバーにプッシュダウンできます。この機能はリモートクエリを最適化して、外部サーバーから転送される行数を減らします。

4.4 列プッシュダウン

MySQL Foreign Data Wrapper はカラムプッシュダウンをサポートしています。その結果、クエリは選択ターゲットリストの一部である列のみを戻します。

4.5 参加プッシュダウン

MySQL Foreign Data Wrapper は参加プッシュダウンをサポートしています。同じリモート MySQL サーバーの外部テーブル間の結合をそのリモート MySQL サーバーにプッシュし、パフォーマンスを向上させます。

注釈:

- 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ます。
- INNER および LEFT/RIGHT OUTER ジョインのみがサポートされています。
-

次も参照してください。

- [Example: Join pushdown](#)
- [Blog: Join Pushdown](#) -パフォーマンスの向上とパーティション単位の結合プッシュダウンをカバー

4.6 アグリゲートプッシュダウン

MySQL Foreign Data Wrapper は、min、max、sum、avg、および count 集計関数の集計プッシュダウンをサポートしているため、すべての行をフェッチしてローカルに集計する代わりに、リモート MySQL サーバーに集計をプッシュできます。集計フィルターと集計オーダーは、MySQL でサポートされていないためプッシュダウンされません。

次も参照してください。

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

また、[Blog: Aggregate Pushdown](#) を参照してください - パフォーマンスの向上、結合と集計のプッシュダウンを一緒に使用し、集計をパーティションテーブルにプッシュダウン

4.7 ORDER BY プッシュダウン

MySQL 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て、外部サーバー自身自身から順序付けられた結果セットを取得します。効率的なマージ結合に役立つ場合があります。NULL の動作は、MySQL サーバーでは反対です。したがって、同等の結果を得るには、各 ORDER BY 式の先頭に「expression IS NULL」句を追加します。

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

4.8 LIMIT OFFSET プッシュダウン

MySQL Foreign Data Wrapper は、制限オフセットの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 PostgreSQL とリモートの MySQL 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。ALL/NULL オプションは MySQL サーバーでサポートされていないため、プッシュダウンされません。また、LIMIT のない OFFSET は MySQL サーバーでサポートされていないため、そのコンストラクトを持つクエリはプッシュされません。

詳細については、Example: LIMIT OFFSET pushdown を参照してください

4.9 プッシュダウンを制限する設定ファイル

MySQL 2.9.0 以降は、プッシュダウンを制限するための `mysql_fdw_pushdown.config` 構成ファイルを提供します。このファイルで、リモートサーバーにプッシュダウンできるファンクションと演算子のリストを定義できます。要件に応じてリストを簡単に追加または変更できます。

このファイルには、リモートサーバーへのプッシュダウンが許可されている集計、ファンクション、およびオペレーターとしてオブジェクトがリストされています。各エントリは 1 行にする必要があります。各エントリには 2 つの列が必要です。

- ROUTINE（ファンクション、集計、およびプロシージャ）または OPERATOR にできるオブジェクトタイプ
- 2 列目は、スキーマ修飾オブジェクト名とその引数です

2 列目の正確な形式は、次のクエリを使用してフォーマットできます。

ルーチンの場合：

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

オペレーターの場合：

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

`mysql_fdw_pushdown.config` ファイルの例：

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

4.10 IS [NOT] DISTINCT FROM 演算子

MySQL 2.9.0 以降は、IS [NOT] DISTINCT FROM 演算子をサポートしています。MySQL は、IS NOT DISTINCT FROM 演算子に対応する `<=>` 演算子と、IS DISTINCT FROM 演算子に対応する `NOT <=>` 演算子を使用します。

4.11 準備されたステートメント

MySQL Foreign Data Wrapper は Prepared Statement をサポートしています。選択クエリは、単純なクエリプロトコルの代わりに準備されたステートメントを使用します。

4.12 外部スキーマのインポート

MySQL Foreign Data Wrapper はインポート外部スキーマをサポートしています。これにより、ローカルホストは MySQL サーバーから EDB Postgres Advanced Server にテーブル定義をインポートできます。新しい外部テーブルは、対応する列の種類と既存のローカルスキーマのリモートテーブルと同じ名前で作成されます。バージョン 2.9.0 以降、外部サーバーからインポートされた外部テーブルの定義に式から生成された列を含める `import_generated` オプションをサポートしています。

詳細については、例については [Example: Import foreign schema](#) を参照してください。

4.13 自動クリーンアップ

MySQL Foreign Data Wrapper では、`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設定されている場合に役立ちます。構文は次のとおりです。

```
DROP EXTENSION mysql_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。

4.14 テーブルを切り捨てる

バージョン 2.9.0 以降、MySQL Foreign Data Wrapper は外部テーブルで `TRUNCATE TABLE` コマンドをサポートします。ただし、`CASCADE` オプションは、`TRUNCATE TABLE` コマンドではサポートされていません。

第5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)

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

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

5.2 SUSE Linux Enterprise (SLES) とその派生

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

5.3 Debian とその派生物

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

Linux on IBM Power (ppc64le)

5.4 Red Hat Enterprise Linux (RHEL)

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

5.5 SUSE Linux Enterprise (SLES)

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

5.5.1 Installing MySQL 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 MySQL Foreign Data Wrapper on CentOS 7 x86_64*
- *Installing MySQL Foreign Data Wrapper on SLES 15 x86_64*
- *Installing MySQL Foreign Data Wrapper on SLES 12 x86_64*
- *Installing MySQL Foreign Data Wrapper on Ubuntu 20.04 x86_64*
- *Installing MySQL Foreign Data Wrapper on Ubuntu 18.04 x86_64*
- *Installing MySQL Foreign Data Wrapper on Debian 11 x86_64*
- *Installing MySQL Foreign Data Wrapper on Debian 10 x86_64*

Installing MySQL 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
```

パッケージをインストールする

```
sudo dnf -y install edb-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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
```

パッケージをインストールする

```
sudo dnf -y install edb-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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"

# Download and install the MYSQL repo:
sudo yum -y install https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.
↳rpm

# Enable the MYSQL repo:
# For MySQL 8:
sudo yum -y install --enablerepo=mysql80-community --disablerepo=mysql57-community_
↳edb-as<xx>-mysql8_fdw

# For MySQL 5:
sudo yum -y install --enablerepo=mysql57-community --disablerepo=mysql80-community_
↳edb-as<xx>-mysql5_fdw
```

パッケージをインストールする

```
sudo yum -y install edb-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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

# Download and install the MySQL repo:
sudo yum -y install https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.
↳rpm

# Enable the MySQL repo:
# For MySQL 8:
sudo yum -y install --enablerepo=mysql80-community --disablerepo=mysql57-community
↳edb-as<xx>-mysql8_fdw

# For MySQL 5:
sudo yum -y install --enablerepo=mysql57-community --disablerepo=mysql80-community
↳edb-as<xx>-mysql5_fdw
```

パッケージをインストールする

```
sudo yum -y install edb-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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

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

# Enable the MySQL8 repository and disable the MySQL 5 repository
sudo zypper modifyrepo -e mysql80-community
sudo zypper modifyrepo -d mysql57-community

# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/15.3/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-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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

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

# Enable the MySQL8 repository and disable the MySQL 5 repository
sudo zypper modifyrepo -e mysql80-community
sudo zypper modifyrepo -d mysql57-community

# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/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-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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。

- その他の前提条件に対処する

```
# Download the GPG key to your APT keyring directly using the apt-key utility:
```

```
sudo apt-key adv --keyserver pgp.mit.edu --recv-keys 3A79BD29
```

```
# Install and configure the MySQL repo:
```

(次のページに続く)

(前のページからの続き)

```
# For MySQL 8:
sudo echo "deb http://repo.mysql.com/apt/debian/bionic mysql-8.0" | sudo tee /etc/
→apt/sources.list.d/mysql.list
# For MySQL 5:
sudo echo "deb http://repo.mysql.com/apt/debian/bionic mysql-5.7" | sudo tee /etc/
→apt/sources.list.d/mysql.list

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

パッケージをインストールする

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

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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。

- その他の前提条件に対処する

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

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

(次のページに続く)

(前のページからの続き)

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

パッケージをインストールする

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

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は `edb-as13-mysql8-fdw` です。

Installing MySQL 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。

- その他の前提条件に対処する

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

# Install and configure the MySQL repo:
For MySQL 8:
sudo echo "deb http://repo.mysql.com/apt/debian/buster mysql-8.0" | sudo tee /etc/
→apt/sources.list.d/mysql.list
For MySQL 5:
sudo echo "deb http://repo.mysql.com/apt/debian/buster mysql-5.7" | sudo tee /etc/
→apt/sources.list.d/mysql.list

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

パッケージをインストールする

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

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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
```

パッケージをインストールする

```
sudo dnf -y install edb-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

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

Installing MySQL 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

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

# Enable the MySQL8 repository and disable the MySQL 5 repository
sudo zypper modifyrepo -e mysql80-community
sudo zypper modifyrepo -d mysql57-community

# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/15.3/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-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は edb-as13-mysql8-fdw です。

Installing MySQL 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

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

# Enable the MySQL8 repository and disable the MySQL 5 repository
sudo zypper modifyrepo -e mysql80-community
sudo zypper modifyrepo -d mysql57-community

# Activate the required SUSE module
sudo SUSEConnect -p PackageHub/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-as<xx>-mysql<y>-fdw
```

<xx> は EDB Postgres Advanced サーバーのバージョンであり、<y> はインストールする MySQL のバージョンです。たとえば、EDB Postgres のバージョンが 13 で MySQL のバージョンが 8 の場合、パッケージ名は `edb-as13-mysql8-fdw` です。

第6章 Initial configuration

MySQL 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 ホストに MySQL 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、MySQL 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. **外国のテーブルを作成する** コマンドを使用して、MySQL サーバーにあるテーブルに対応する単一のテーブルを定義するか、**外部スキーマのインポート** コマンドを使用してローカルスキーマに複数のリモートテーブルをインポートします。

6.1 拡張機能を作成する

CREATE EXTENSION コマンドを使用して、mysql_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、MySQL 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] mysql_fdw [WITH] [SCHEMA schema_name];
```

6.1.1 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。

6.1.2 例

次のコマンドは、MySQL 外部データラッパーをインストールします。

```
CREATE EXTENSION mysql_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。

6.2 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 mysql_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** 権限が必要です。

6.2.1 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 **mysql_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
character_set	MySQL 接続に使用する文字セット。デフォルト値は <code>'auto'</code> で、オペレーティングシステム設定に基づいた自動検出を意味します。 character_set オプションが導入される前は、文字セットは <i>PostgreSQL</i> データベースのエンコーディングと同様に設定されていました。この古い動作を取得するには、 character_set を特別な値 <code>'PGDatabaseEncoding'</code> に設定します。

次のページに続く

表 1 – 前のページからの続き

Option	Description
fetch_size	このオプションは、mysql_fdw が各フェッチ操作で取得する行数を指定します。外部テーブルまたは外部サーバーに指定できます。テーブルに指定されたオプションは、サーバーに指定されたオプションをオーバーライドします。デフォルト値は`100`です。
host	MySQL 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。
init_command	MySQL サーバーへの接続時に実行する SQL ステートメント。
port	MySQL サーバーのポート番号。デフォルトは`3306`です。
reconnect	既存の接続が失われた場合の MySQL サーバーへの自動再接続を有効または無効にします。デフォルト値は`false`です。
secure_auth	安全な認証を有効または無効にするために使用します。デフォルトは`true`です。
sql_mode	確立された接続に MySQL `sql_mode`を設定します。デフォルトは`ANSI_QUOTES`です。
ssl_key	クライアントの秘密鍵ファイルのパス名。
ssl_cert	クライアント公開鍵証明書ファイルのパス名。
ssl_ca	認証局 (CA) 証明書ファイルのパス名。このオプションを使用する場合、サーバーが使用するのと同じ証明書を指定する必要があります。
ssl_capath	信頼できる SSL CA 証明書ファイルを含むディレクトリのパス名。
ssl_cipher	SSL 暗号化で許可される暗号のリスト。
use_remote_estimate	`use_remote_estimate`を含めて、処理コストを見積もるときにリモートサーバーで`EXPLAIN`コマンドを使用するようにサーバーに指示します。デフォルトでは、`use_remote_estimate`は`false`です。

6.2.2 例

次のコマンドは、mysql_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 mysql_server という名前の外部サーバーを作成します。

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

外部サーバーは、MySQL 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（3306）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。

6.3 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。

6.3.1 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、MySQL 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、MySQL サーバー上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。

6.3.2 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、mysql_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 mysql_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 public SERVER mysql_server OPTIONS (username foo, password_
→bar);
```

このコマンドは、mysql_server という名前のサーバーに関連付けられている public 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。MySQL サーバーに接続すると、サーバーは foo として認証され、bar のパスワードを提供します。

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。

6.4 外国のテーブルを作成する

外部テーブルは、MySQL ホストにあるテーブルへのポインタです。Postgres サーバーで外部テーブル定義を作成する前に、MySQL サーバーに接続してテーブルを作成します。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、MySQL ホストにあるテーブル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 }
    [, ... ]
] )
[ 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 ]
```

6.4.1 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、MySQL 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 MySQL 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 MySQL 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しますが、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搬しないように指定します。

`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`' [, ...])]

MySQL 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、MySQL 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
<code>dbname</code>	The name of the database on the MySQL server. The database name is required.
<code>fetch_size</code>	This option specifies the number of rows <code>mysql_fdw</code> should get in each fetch operation. It can be specified for a foreign table or a foreign server. The option specified on a table overrides an option specified for the server. The default value is <code>100</code> .
<code>table_name</code>	The name of the table on the MySQL server. The default is the name of the foreign table.
<code>max_blob_size</code>	The maximum BLOB size to read without truncation.

6.4.2 例

MySQL サーバーに保存されているデータを使用するには、MySQL テーブルの列を 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 MySQL テーブルの場合。

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

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
(
  warehouse_id      INT,
  warehouse_name    TEXT,
  warehouse_created TIMESTAMP
)
SERVER mysql_server
  OPTIONS (dbname db, table_name warehouse);
```

SERVER 句を含めて、MySQL 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: MySQL 外部データラッパーは書き込み機能機能をサポートしています。

6.4.3 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、MySQL 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。MySQL データラッパーは、次の MySQL データ型をターゲットの Postgres 型に変換します。

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

次のページに続く

表 3 – 前のページからの続き

MySQL	Postgres
INT/INTEGER	INT/INTEGER/INT4
LONGTEXT	TEXT
SMALLINT	SMALLINT/INT2
TIMESTAMP	TIMESTAMP
VARCHAR()	VARCHAR()/CHARACTER VARYING()

注釈: *ENUM* データ型の場合:

MySQL は、文字列形式の *enum* 値を受け入れます。MySQL サーバーに存在するのとまったく同じ *enum* リストを EDB Postgres Advanced Server に作成する必要があります。ローカルサーバーで不明な値を持つ行をフェッチしているときに、何らかの不整合が発生するとエラーが発生します。

また、指定された *enum* 値が MySQL 側に存在しないが、EDB Postgres Advanced Server 側に存在する場合、空の文字列 ('') が *enum* 列の MySQL 側で値として挿入されます。 *enum* 値を '' として持つテーブルから選択するには、Postgres 側ですべての有効な値と '' を使用して *enum* タイプを作成します。

6.5 外国人スキーマのインポート

`IMPORT FOREIGN SCHEMA` コマンドを使用して、MySQL サーバーから Postgres サーバーにテーブル定義をインポートします。新しい外部テーブルは、既存のローカススキーマのリモートテーブルと同じ列定義で作成されます。構文は次のとおりです。

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

6.5.1 パラメーター

`remote_schema`

インポートするリモートスキーマ (MySQL データベース) を指定します。

`LIMIT TO ( table_name [, ...] )`

デフォルトでは、MySQL ホスト上の特定のデータベースに存在するすべてのビューとテーブルがインポートされます。このオプションを使用すると、テーブルのリストを指定したサブセットに制限できます。

`EXCEPT ( table_name [, ...] )`

デフォルトでは、MySQL ホスト上の特定のデータベースに存在するすべてのビューとテーブルがインポートされます。このオプションを使用すると、指定した外部テーブルをインポートから除外できます。

SERVER *server_name*

外部テーブルをインポートするサーバーの名前を指定します。

local_schema

インポートされた外部テーブルを作成するローカスキーマの名前を指定します。

OPTIONS

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	Description
import_default	外部サーバーからインポートされた外部テーブルの定義に列 `DEFAULT` 式を含めるかどうかを制御します。デフォルトは `false` です。
import_not_null	外部サーバーからインポートされた外部テーブルの定義に列 `NOT NULL` 制約を含めるかどうかを制御します。デフォルトは `true` です。
import_enum_as_text	外部テーブルの定義で MySQL `ENUM` タイプを `TEXT` タイプにマップするために使用できます。それ以外の場合、タイプが作成されるという警告を出力します。デフォルトは `false` です。

6.5.2 例

次の定義で edb データベースに作成された MySQL テーブルの場合：

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

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

Postgres サーバー上の同等のテーブルをインポートするコマンドを実行します。

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

SELECT * FROM color;

cid | cname
-----+-----
  1 | Red
  2 | Green
  3 | Orange
(3 rows)
```

(次のページに続く)

(前のページからの続き)

```
SELECT * FROM fruit;
```

```

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

このコマンドは、サーバー `mysql_server` 上のリモートスキーマ `edb` からテーブル定義をインポートし、ローカルスキーマ `public` に外部テーブルを作成します。

IMPORT FOREIGN SCHEMA 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。

6.6 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、MySQL 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 ];
```

6.6.1 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、拡張機能を削除しないでください。

6.6.2 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 mysql_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。

6.7 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。

6.7.1 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。

6.7.2 例

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

```
DROP SERVER mysql_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。

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;
```

6.7.3 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スローする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

MySQL 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。

6.7.4 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、mysql_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 mysql_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。

6.8 外国のテーブルをドロップ

外部テーブルは、MySQL ホストにあるテーブルへのポインタです。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 ]
```

6.8.1 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。

6.8.2 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7章 Upgrading

7.1 RPM インストールのアップグレード

MySQL Foreign Data Wrapper の既存の RPM インストールがある場合、yum または dnf を使用してリポジトリ構成ファイルをアップグレードし、より新しい製品バージョンにアップグレードできます。yum または dnf は edb.repo ファイルを更新して、edb.repo ファイルで指定された資格情報で接続するように構成された現在の EDB リポジトリへのアクセスを有効にします。次に、yum または dnf を使用して、インストールされているパッケージをアップグレードできます。

- MySQL 8 の場合：
- RHEL または CentOS 7 の場合：

edb_notranlate_0 - RHEL または Rocky Linux または AlmaLinux 8 の場合：

```
sudo dnf -y upgrade edb-as<xx>-mysql8_fdw* mysql-community-devel
```

- RHEL または CentOS 7 の MySQL 5 の場合：

```
sudo yum -y upgrade edb-as<xx>-mysql5_fdw* mysql-community-devel
```

7.2 Debian または Ubuntu ホストでの MySQL Foreign Data Wrapper のアップグレード

以前にリリースされたバージョンの MySQL FDW では、MySQL FDW のパッケージ名は edb-as*-mysql-fdw でした。現在のリリースでは、MySQL 5 および MySQL 8 で 2 つの個別のパッケージ、つまり edb-as*-mysql5-fdw および edb-as*-mysql8-fdw が利用可能になりました。

MySQL 5 の MySQL FDW 2.5.5 から 2.6.0 にアップグレードする場合、バージョン 2.6.0 の edb-as*-mysql5-fdw パッケージをインストールすると、以前にインストールしたパッケージ edb-as*-mysql-fdw が自動的に削除されます。

MySQL 8 の MySQL FDW 2.5.5 から 2.6.0 にアップグレードする場合、バージョン 2.6.0 の edb-as*-mysql8-fdw パッケージをインストールする前に、次のパッケージを手動で削除する必要があります。

- default-mysqclient-dev
- edb-as*-mysql-fdw

次のコマンドを使用して、FDW インストールをアップグレードします。

- MySQL 8 の場合：

```
apt-get upgrade edb-as<xx>-mysql8-fdw*
```

- MySQL 5 の場合：

edb_notranlate_4 ここで、**xx** はサーバーバージョン番号です。

第8章 Uninstalling

`remove` コマンドを使用して、MySQL 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した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。

- RHEL または CentOS 7 の場合：

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

ここで： - <xx> は EDB Postgres Advanced Server のバージョンです。 - <x> は、MySQL のサポートされているリリースバージョン番号です。

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

```
dnf remove edb-as<xx>-mysql8_fdw <xx> は EDB Postgres Advanced Server のバージョンです。
```

- SLES の場合：

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

ここで： - <xx> は EDB Postgres Advanced Server のバージョンです。 - <x> は、MySQL のサポートされているリリースバージョン番号です。

- Debian または Ubuntu の場合

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

<xx> は EDB Postgres Advanced Server のバージョンです。

第9章 End-to-end

EDB Postgres Advanced Server からデータにアクセスし、psql に接続します。psql に接続したら、次の手順に従います。

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

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

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

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

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

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

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

(次のページに続く)

```

- - delete row from table
DELETE FROM warehouse where warehouse_id = 3;

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

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

```

QUERY PLAN

```

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

- - drop foreign table
DROP FOREIGN TABLE warehouse;

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

- - drop server
DROP SERVER mysql_server;

```

第10章 Import foreign schema

EDB Postgres Advanced Server からデータにアクセスし、psql に接続します。psql に接続したら、次の手順に従います。

edb_notranlate_0 詳細については、[外部スキーマのインポート](#) を参照してください。

第11章 Join pushdown

この例は、2つの外部テーブル warehouse と sales_records 間の結合プッシュダウンを示しています。

MySQL サーバー上のテーブル：

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

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

Postgres サーバー上のテーブル：

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

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

CREATE FOREIGN TABLE sales_records
(
warehouse_id      INT,
```

(次のページに続く)

(前のページからの続き)

```

qty          INT
)
SERVER mysql_server OPTIONS (dbname edb, table_name sales_records);
INSERT INTO sales_records values (1, 100);
INSERT INTO sales_records values (2, 75);
INSERT INTO sales_records values (3, 200);

```

出力：

```

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

→PLAN
-----
→-----
Foreign Scan (cost=15.00..35.00 rows=5000 width=36)
  Output: t1.warehouse_name, t2.qty
  Relations: (edb.warehouse t1) INNER JOIN (edb.sales_records t2)
  Local server startup cost: 10
  Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`warehouse` r1_
→INNER JOIN `edb`.`sales_records` r2 ON (((r1.`warehouse_id` = r2.`warehouse_id`))))
(5 rows)

```

```

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

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

```

```

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

```

(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=36)
  Output: t1.warehouse_name, t2.qty
  Relations: (edb.sales_records t2) LEFT JOIN (edb.warehouse t1)
  Local server startup cost: 10
  Remote query: SELECT r1.`warehouse_name`, r2.`qty` FROM (`edb`.`sales_records` r2
→LEFT JOIN `edb`.`warehouse` r1 ON (((r1.`warehouse_id` = r2.`warehouse_id`)))
(5 rows)

```

- -cross join

```

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

```

__OUTPUT__

QUERY PLAN

```

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

```


第12章 Aggregate pushdown

MySQL 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。

MySQL サーバー上のテーブル：

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

Postgres サーバー上のテーブル：

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

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

出力：

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

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

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

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

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

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

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

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

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

```
__OUTPUT__
```

QUERY PLAN

```
-----
```

```
Foreign Scan (cost=15.00..25.00 rows=1 width=8)
```

```
Output: (sum(qty))
```

```
Relations: Aggregate on (edb.sales_records)
```

```
Local server startup cost: 10
```

```
Remote query: SELECT sum(`qty`) FROM `edb`.`sales_records`
```

```
(5 rows)
```

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

```
__OUTPUT__
```

QUERY PLAN

```
-----
```

```
→-----
```

```
Foreign Scan (cost=15.00..25.00 rows=200 width=12)
```

```
Output: (sum(qty)), warehouse_id
```

```
Relations: Aggregate on (edb.sales_records)
```

```
Local server startup cost: 10
```

```
Remote query: SELECT sum(`qty`), `warehouse_id` FROM `edb`.`sales_records` GROUP_
→BY 2 HAVING ((sum(`qty`) = 75))
```

```
(5 rows)
```


第13章 ORDER BY pushdown

この例は、外部テーブル sales_records での ORDER BY プッシュダウンを示しています。

MySQL サーバー上のテーブル：

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

Postgres サーバー上のテーブル：

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

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

- - create user mapping
CREATE USER MAPPING FOR public SERVER mysql_server OPTIONS (username edb, password_
→edb);

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

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

出力：

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

QUERY PLAN

→

(次のページに続く)

(前のページからの続き)

```

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

```

```

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

                                QUERY PLAN

-----
→-----
Foreign Scan on public.sales_records (cost=10.00..1010.00 rows=1000 width=8)
  Output: warehouse_id, qty
  Local server startup cost: 10
  Remote query: SELECT `warehouse_id`, `qty` FROM `edb`.`sales_records` ORDER BY
→`warehouse_id` IS NOT NULL, `warehouse_id` DESC
(4 rows)

```

```

- - ORDER BY with AGGREGATES
edb@91975=#EXPLAIN VERBOSE SELECT count(warehouse_id) FROM sales_records WHERE qty >
→80 group by warehouse_id ORDER BY warehouse_id;
__OUTPUT__

                                QUERY PLAN

-----
→-----
→-----
Foreign Scan (cost=15.00..25.00 rows=10 width=12)
  Output: (count(warehouse_id)), warehouse_id
  Relations: Aggregate on (edb.sales_records)
  Local server startup cost: 10
  Remote query: SELECT count(`warehouse_id`), `warehouse_id` FROM `edb`.`sales_
→records` WHERE ((`qty` > 80)) GROUP BY 2 ORDER BY `warehouse_id` IS NULL,
→`warehouse_id` ASC
(5 rows)

```

第14章 LIMIT OFFSET pushdown

この例は、外部テーブル sales_records での LIMIT OFFSET プッシュダウンを示しています。

MySQL サーバー上のテーブル：

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

Postgres サーバー上のテーブル：

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

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

- - create user mapping
CREATE USER MAPPING FOR public SERVER mysql_server OPTIONS (username edb, password_
↳edb);

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

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

出力：

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

QUERY PLAN

(次のページに続く)

(前のページからの続き)

```

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

```

```

- - LIMIT and OFFSET
edb@91975=#EXPLAIN VERBOSE SELECT * FROM sales_records WHERE qty > 80 ORDER BY
→warehouse_id LIMIT 5 OFFSET 5;
__OUTPUT__
QUERY
→PLAN
-----
→
Foreign Scan on public.sales_records (cost=1.00..2.00 rows=1 width=8)
  Output: warehouse_id, qty
  Local server startup cost: 10
  Remote query: SELECT `warehouse_id`, `qty` FROM `edb`.`sales_records` WHERE
→((`qty` > 80)) ORDER BY `warehouse_id` IS NULL, `warehouse_id` ASC LIMIT 5 OFFSET 5
(4 rows)

```

第15章 Identifying the version

MySQL 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 mysql_fdw_version();
```

この関数はバージョン番号を返します。

```
mysql_fdw_version  
-- -----  
<xxxxxx>
```


第16章 Authentication plugin ‘caching_sha2_password’ error

次のエラーが発生した場合：

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

認証プラグインを `mysql_native_password` として指定し、クリアテキストのパスワード値を設定します。構文は次のとおりです。

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

注釈： このエラーの詳細については、 [MySQL 8 documentation](#) を参照してください。
