



Hadoop Foreign Data Wrapper

リリース 2.3.0

EnterpriseDB Corporation

2023 年 01 月 19 日

目次

第 1 章	Release Notes	3
1.1	Version 2.3.0	3
1.2	Version 2.2.0	4
1.3	Version 2.1.0	4
1.4	Version 2.0.8	4
1.5	Version 2.0.7	5
1.6	Version 2.0.5	5
1.7	Version 2.0.4	5
第 2 章	Supported database versions	7
第 3 章	Architecture overview	9
第 4 章	Key features	11
4.1	WHERE 句プッシュダウン	11
4.2	列プッシュダウン	11
4.3	参加プッシュダウン	11
4.4	アグリゲートプッシュダウン	11
4.5	ORDER BY プッシュダウン	12
4.6	LIMIT OFFSET プッシュダウン	12
4.7	自動クリーンアップ	12
第 5 章	Supported authentication methods	13
5.1	LDAP 認証の使用	13
5.2	NOSASL 認証の使用	14
第 6 章	Installing Hadoop Foreign Data Wrapper	15
6.1	Red Hat Enterprise Linux (RHEL) およびその派生物	15
6.2	SUSE Linux Enterprise (SLES) とその派生	15
6.3	Debian とその派生物	15
6.4	Red Hat Enterprise Linux (RHEL)	16
6.5	SUSE Linux Enterprise (SLES)	16
6.5.1	Installing Hadoop Foreign Data Wrapper on Linux x86 (amd64)	16
6.5.2	Installing Hadoop Foreign Data Wrapper on IBM Power (ppc64le)	23
第 7 章	Initial configuration	27
7.1	拡張機能を作成する	28
7.1.1	パラメーター	28
7.1.2	例	28
7.2	サーバーを作成する	28
7.2.1	パラメーター	29

7.2.2	例	30
7.3	ユーザーマッピングを作成する	30
7.3.1	パラメーター	31
7.3.2	例	31
7.4	外国のテーブルを作成する	31
7.4.1	パラメーター	32
7.4.2	例	34
7.5	データ型のマッピング	35
7.6	ドロップエクステンション	35
7.6.1	パラメーター	36
7.6.2	例	36
7.7	ドロップサーバー	36
7.7.1	パラメーター	37
7.7.2	例	37
7.8	DROP ユーザーマッピング	37
7.8.1	パラメーター	37
7.8.2	例	38
7.9	外国のテーブルをドロップ	38
7.9.1	パラメーター	38
7.9.2	例	38
第 8 章	Upgrading	39
8.1	RHEL、Rocky Linux、AlmaLinux または OL 8 の場合	39
8.2	SLES について	39
8.3	Debian または Ubuntu の場合	40
8.4	PPCLE の RHEL または CentOS 7 の場合	40
第 9 章	Uninstalling	41
第 10 章	Using Apache Hive or Apache Spark	43
10.1	Hadoop 上で Apache Hive で HDFS FDW を使用する	43
10.2	Hadoop 上で Apache Spark で HDFS FDW を使用する	46
第 11 章	Join pushdown	51
第 12 章	Aggregate pushdown	57
第 13 章	ORDER BY pushdown	59
第 14 章	LIMIT OFFSET pushdown	61
第 15 章	Identifying the version	63

Hadoop Foreign Data Wrapper (hdfs_fdw) は、Hadoop ファイルシステムにあるデータに EDB Postgres Advanced Server からアクセスできるようにする Postgres の拡張機能です。外部データラッパーにより、Hadoop ファイルシステムは、Postgres の機能とユーティリティ、または Postgres ホストにある他のデータで利用できる読み取り専用データソースになります。

第1章 Release Notes

Hadoop 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最新バージョンが記載されています。このセクションの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、コンテンツ内には、その機能が導入されたリリースに関するインジケータもあります。

Version	Release Date
<i>Version 2.3.0</i>	2023 Jan 06
<i>Version 2.2.0</i>	2022 May 26
<i>Version 2.1.0</i>	2021 Dec 02
<i>Version 2.0.8</i>	2021 Jun 24
<i>Version 2.0.7</i>	2020 Nov 23
<i>Version 2.0.5</i>	2019 Dec 10
<i>Version 2.0.4</i>	2018 Nov 28

1.1 Version 2.3.0

Hadoop Foreign Data Wrapper 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
Feature	可能な場合、 <i>ORDER BY</i> 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を有効にします。
Feature	<i>LIMIT</i> / <i>OFFSET</i> をリモート HDFS サーバーにプッシュダウンします。可能な場合、リモートサーバーで <i>LIMIT</i> および <i>OFFSET</i> 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 HDFS / Hive 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。

1.2 Version 2.2.0

Hadoop Foreign Data Wrapper 2.2 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	アグリゲートをリモート HDFS 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、アグリゲートをリモート HDFS 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のパフォーマンスが向上します。
Enhancement	NUMERIC データ型のサポートを追加します。
Bug fix	非組み込みデータ型として評価される式をプッシュしないでください。

1.3 Version 2.1.0

Hadoop Foreign Data Wrapper 2.1.0 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。

1.4 Version 2.0.8

Hadoop Foreign Data Wrapper 2.0.8 の新機能、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	Hadoop バージョン 3.2.x のサポート
Enhancement	Hive バージョン 3.1.x のサポート
Enhancement	Spark バージョン 3.0.x のサポート
Bug Fix	エラーを回避するために、行全体を参照する SELECT クエリのビルドを修正しました。
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。
Bug Fix	特殊文字を含むテーブルまたは列の名前には、適切なハイブ SQL 引用符を使用します。

1.5 Version 2.0.7

Hadoop Foreign Data Wrapper 2.0.7 の新機能、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。
Enhancement	LICENSE ファイルを更新しました。

1.6 Version 2.0.5

Hadoop Foreign Data Wrapper 2.0.5 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 12 のサポート。

1.7 Version 2.0.4

Hadoop Foreign Data Wrapper 2.0.4 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、Hadoop 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。Hadoop 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。

Hadoop Foreign Data Wrapper	EPAS 14	EPAS 13	EPAS 12	EPAS 11
2.3.0	Y	Y	Y	Y
2.2.0	Y	Y	Y	Y
2.1.0	Y	Y	Y	Y
2.0.8	N	Y	Y	Y
2.0.7	N	Y	Y	N
2.0.5	N	N	Y	N
2.0.4	N	N	N	Y

第3章 Architecture overview

Hadoop は、大規模なデータセットを分散ファイルシステムに保存できるフレームワークです。

Hadoop データラッパーは、Hadoop ファイルシステムと Postgres データベース間のインターフェイスを提供します。Hadoop データラッパーは、Postgres SELECT ステートメントを HiveQL または Spark SQL インターフェイスが理解するクエリに変換します。

! Using a Hadoop distributed file system with Postgres

可能な場合、外部データラッパーは Hive または Spark サーバーに、SELECT ステートメントの WHERE 句に関連付けられたアクションを実行するように要求します。WHERE 句をプッシュダウンすると、ネットワーク上で移動するデータの量が減少するため、パフォーマンスが向上します。

第4章 Key features

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

4.1 WHERE 句プッシュダウン

Hadoop 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。

4.2 列プッシュダウン

Hadoop Foreign Data Wrapper は、列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。

4.3 参加プッシュダウン

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

バージョン 2.3.0 以降では、`enable_join_pushdown` GUC 変数を使用して、セッションおよびクエリレベルで参加プッシュダウンを有効にできます。

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

4.4 アグリゲートプッシュダウン

Hadoop Foreign Data Wrapper は、アグリゲート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のではなく、集計をリモートの Hive または Spark 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、Hadoop 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、現在、`min`、`max`、`sum`、`avg`、および `count` の集計関数に制限されています。また、集計フィルターと注文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。

4.5 ORDER BY プッシュダウン

Hadoop Foreign Data Wrapper は、オーダーバイ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を有効にします。

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

4.6 LIMIT OFFSET プッシュダウン

Hadoop 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 Postgres とリモートの HDFS / Hive 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 Hive サーバーでは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、リモートサーバーでは LIMIT のない OFFSET はサポートされていません。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

4.7 自動クリーンアップ

Hadoop 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設定されている場合に特に役立ちます。構文は次のとおりです。

```
DROP EXTENSION hdfs_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。

第5章 Supported authentication methods

Hadoop Foreign Data Wrapper は、NOSASL および LDAP 認証モードをサポートしています。NOSASL を使用するには、ユーザーマッピングの作成中に OPTIONS 値を指定しないでください。LDAP 認証モードの場合、ユーザーマッピング作成時に OPTIONS に username と password を指定します。

5.1 LDAP 認証の使用

LDAP 認証で Hadoop Foreign Data Wrapper を使用する場合、最初に Hive サーバーまたは Spark サーバーを LDAP 認証を使用するように構成します。構成されたサーバーは、LDAP サーバーの接続の詳細を含む hive-site.xml ファイルを提供する必要があります。例：

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
  <description>
    Expects one of [n SASL, none, ldap, kerberos, pam, custom].
    Client authentication types.
    NONE: no authentication check
    LDAP: LDAP/AD based authentication
    KERBEROS: Kerberos/GSSAPI authentication
    CUSTOM: Custom authentication provider
           (Use with property hive.server2.custom.authentication.class)
    PAM: Pluggable authentication module
    NOSASL: Raw transport
  </description>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>ldap://localhost</value>
  <description>LDAP connection URL</description>
</property>
<property>
  <name>hive.server2.authentication.ldap.baseDN</name>
  <value>ou=People,dc=itzgeek,dc=local</value>
  <description>LDAP base DN</description>
</property>
```

次に、Hive サーバーを起動するときに、コマンドに hive-site.xml ファイルへのパスを含めます。例：

```
./hive --config <path_to_hive-site.xml_file> --service hiveServer2
```

<path_to_hive-site.xml_file> hive - site.xml ファイルへの完全なパスを指定します。

ユーザーマッピングを作成するときは、登録済みの LDAP ユーザーの名前と対応するパスワードをオプションとして指定する必要があります。詳細については、[ユーザーマッピングを作成する](#) を参照してください。

5.2 NOSASL 認証の使用

Hadoop Foreign Data Wrapper で NOSASL 認証を使用する場合、Hive サーバーまたは Spark サーバーで承認を None に、認証方法を NOSASL に設定します。たとえば、コマンドラインで Hive サーバーを起動する場合、コマンドに `hive.server2.authentication` 構成パラメーターを含めます。

```
hive --service hiveserver2 --hiveconf hive.server2.authentication=NOSASL
```

第6章 Installing Hadoop Foreign Data Wrapper

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

6.2 SUSE Linux Enterprise (SLES) とその派生

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

6.3 Debian とその派生物

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

Linux on IBM Power (ppc64le)

6.4 Red Hat Enterprise Linux (RHEL)

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

6.5 SUSE Linux Enterprise (SLES)

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

6.5.1 Installing Hadoop 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 Hadoop Foreign Data Wrapper on CentOS 7 x86_64*
- *Installing Hadoop Foreign Data Wrapper on SLES 15 x86_64*
- *Installing Hadoop Foreign Data Wrapper on SLES 12 x86_64*
- *Installing Hadoop Foreign Data Wrapper on Ubuntu 20.04 x86_64*
- *Installing Hadoop Foreign Data Wrapper on Ubuntu 18.04 x86_64*
- *Installing Hadoop Foreign Data Wrapper on Debian 11 x86_64*
- *Installing Hadoop 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 Hadoop 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
# 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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 Hadoop Foreign Data Wrapper on RHEL 8 or OL 8 x86_64*
- *Installing Hadoop Foreign Data Wrapper on SLES 15 x86_64*
- *Installing Hadoop 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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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 Hadoop 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-hdfs-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

拡張機能と、拡張機能を使用するデータベースオブジェクトを作成する前に、Postgres ホストを変更して、サポートライブラリの場所を提供する必要があります。

Postgres をインストールした後、次の場所にある `postgresql.conf` を変更します。

`/var/lib/edb/as_version/data`

構成ファイルを変更し、`hdfs_fdw.jvm_path` パラメーターを構成ファイルの最後に追加し、値を設定して Java 仮想マシン (`libjvm.so`) の場所を指定します。`hdfs_fdw.classpath` の値を設定して、アダプターが使用する Java クラスファイルの場所を示します。各パス間の区切り文字としてコロン (:) を使用します。例：

```
hdfs_fdw.classpath=
/usr/edb/as12/lib/HiveJdbcClient-1.0.jar:/home/edb/Projects/hadoop_fdw/hadoop/share/
↪hadoop/common/hadoop-common-2.6.4.jar:/home/edb/Projects/hadoop_fdw/apache-hive-1.0.
↪1-bin/lib/hive-jdbc-1.0.1-standalone.jar
```

注釈: jar ファイル (`hive-jdbc-1.0.1-standalone.jar` および `hadoop-common-2.6.4.jar`) を、Hive および Hadoop のソースまたは Web サイトから Hadoop Foreign Data Wrapper がインストールされている PostgreSQL インスタンスにコピーします。

EDB Postgres Advanced Server を使用していて、データベースに `DATE` 列がある場合は、`postgresql.conf` ファイルに `edb_redwood_date = OFF` を設定する必要があります。

パラメーター値を設定したら、Postgres サーバーを再起動します。EDB Postgres Advanced Server ホストでのサービスの制御の詳細については、

EDB Postgres Advanced Server documentation を参照してください。

Hadoop 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、Hadoop ファイルシステム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、Hadoop クラスターにあるデータベースに対応する EDB

Postgres Advanced Server 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、hdfs_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（psql など）を使用して、Hive または Spark 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] hdfs_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、hdfs_fdw Hadoop Foreign Data Wrapper をインストールします。

```
CREATE EXTENSION hdfs_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 ドキュメント] (<https://www.postgresql.org/docs/current/static/sql-createextension.html>) 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 hdfs_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 `hdfs_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。次を含めることができます。

Option	Description
host	Hadoop クラスターのアドレスまたはホスト名。デフォルト値は <code>'localhost'</code> です。
port	Hive Thrift サーバーまたは Spark Thrift サーバーのポート番号。デフォルトは <code>'10000'</code> です。
client_type	クライアントタイプは <code>'hiveserver2'</code> または <code>'spark'</code> を指定します。Spark で <code>'ANALYZE'</code> ステートメントを使用するには、 <code>'spark'</code> の値を指定する必要があります。デフォルト値は <code>'hiveserver2'</code> です。
auth_type	クライアントの認証タイプ。 <code>'LDAP'</code> または <code>'NOSASL'</code> を指定します。 <code>'auth_type'</code> を指定しない場合、データラッパーはユーザーマッピングに基づいて <code>'auth_type'</code> 値を決定します。ユーザーマッピングにユーザー名とパスワードが含まれる場合、データラッパーは LDAP 認証を使用します。ユーザーマッピングにユーザー名とパスワードが含まれていない場合、データラッパーは NOSASL 認証を使用します。
connect_timeout	接続試行がタイムアウトするまでの時間。デフォルト値は 300 秒です
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。 <code>true</code> 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。このオプションを個々のテーブルに設定することもできます。結合に関係するテーブルのいずれかでオプションが <code>false</code> に設定されている場合、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 <code>'true'</code> です。
fetch_size	JDBC API <code>'setFetchSize'</code> へのパラメーターとして提供されます。デフォルト値は <code>'10,000'</code> です。
log_remote_sql	<code>'true'</code> の場合、ログにはリモート Hive サーバーで実行された SQL コマンドとスキャンが繰り返された回数が含まれます。デフォルトは <code>'false'</code> です。
query_timeout	<code>'query_timeout'</code> を使用して、Hive サーバーが要求を満たさない場合にタイムアウトするまでの秒数を提供します。クエリのタイムアウトは、Hive JDBC ドライバーでサポートされていません。

次のページに続く

表 1 – 前のページからの続き

Option	Description
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`であり、リモートテーブルには 1000 行があると想定されます。

7.2.2 例

次のコマンドは、hdfs_fdw 外部データラッパーを使用して 170.11.2.148 の IP アドレスを持つホストに接続する hdfs_server という名前の外部サーバーを作成します。

```
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw OPTIONS (host 170.11.2.148,
→port 10000, client_type hiveserver2, auth_type LDAP, connect_timeout 10000, query_
→timeout 10000);
```

外部サーバーは、Hadoop 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（10000）を使用します。接続は LDAP サーバーを使用します。

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。

注釈: Hadoop Foreign Data Wrapper は NOSASL および LDAP 認証をサポートしています。LDAP 認証を使用するサーバーのユーザーマッピングを作成している場合は、*OPTIONS* 句を使用して、既存の LDAP ユーザーの接続資格情報 (ユーザー名とパスワード) を指定します。サーバーが NOSASL 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 `OPTIONS` 句を省略します。

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、Hadoop 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。LDAP 認証を使用している場合は、次を提供します。

`username` — LDAP サーバー上のユーザーの名前。

`password` — ユーザー名に関連付けられたパスワード。

ユーザー名とパスワードを指定しない場合、データラッパーは NOSASL 認証を使用します。

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 `hdfs_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 hdfs_server;
```

データベースホストが LDAP 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 hdfs_server OPTIONS (username alice, ↵
↵password 1safepwd);
```

このコマンドは、`hdfs_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。LDAP サーバーに接続すると、Hive または Spark サーバーは `alice` として認証し、`1safepwd` 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。

7.4 外国のテーブルを作成する

外部テーブルは、Hadoop 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、Hive または Spark 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、Hadoop 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 ]
```

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、Hive または Spark サーバーで説明されている列に対応しています。

`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 Hive または Spark 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 Hive または Spark サーバーと互換性のある型にデータ型を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。デフォルトでは、列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自動的に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できますが、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' [, ...])]

Hadoop 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、Hadoop データアダプターを使用する外部サーバーの **server_name** 値を指定します。

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
dbname	The name of the database on the Hive server. The database name is required.
table_name	The name of the table on the Hive server. The default is the name of the foreign table.
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. 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

分散ファイルシステムに保存されているデータを使用するには、Hadoop 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 Postgres ホストに作成する必要があります。たとえば、次の定義を持つ Hadoop テーブルの場合。

```
CREATE TABLE weblogs (  
  client_ip          STRING,  
  full_request_date  STRING,  
  day                STRING,  
  month              STRING,  
  month_num          INT,  
  year               STRING,  
  hour               STRING,  
  minute             STRING,  
  second             STRING,  
  timezone           STRING,  
  http_verb          STRING,  
  uri                STRING,  
  http_status_code   STRING,  
  bytes_returned     STRING,  
  referrer           STRING,  
  user_agent         STRING)  
row format delimited  
fields terminated by \t;
```

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 weblogs  
(  
  client_ip          TEXT,  
  full_request_date  TEXT,  
  day                TEXT,  
  Month              TEXT,  
  month_num          INTEGER,  
  year               TEXT,  
  hour               TEXT,  
  minute             TEXT,  
  second             TEXT,  
  timezone           TEXT,  
  http_verb          TEXT,  
  uri                TEXT,  
  http_status_code   TEXT,
```

(次のページに続く)

(前のページからの続き)

```

bytes_returned      TEXT,
referrer            TEXT,
user_agent          TEXT
)
SERVER hdfs_server
    OPTIONS (dbname webdata, table_name weblogs);

```

SERVER 句を含めて、Hadoop ファイルシステム（webdata）の名前と Postgres サーバー上のテーブルに対応するテーブルの名前（weblogs）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。

7.5 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、Hive 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。Hadoop データラッパーは、次の Hive データ型をターゲットの Postgres 型に自動的に変換します。

Hive	Postgres
BIGINT	BIGINT/INT8
BOOLEAN	BOOL/BOOLEAN
BINARY	BYTEA
CHAR	CHAR
DATE	DATE
DOUBLE	FLOAT8
FLOAT	FLOAT/FLOAT4
INT/INTEGER	INT/INTEGER/INT4
SMALLINT	SMALLINT/INT2
STRING	TEXT
TIMESTAMP	TIMESTAMP
TINYINT	INT2
VARCHAR	VARCHAR

7.6 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、Hadoop 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 ];
```

7.6.1 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自動的にドロップします。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。

7.6.2 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 hdfs_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。

7.7 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。

7.7.1 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自動的に削除します。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。

7.7.2 例

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

```
DROP SERVER hdfs_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。

7.8 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;
```

7.8.1 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

Hadoop 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。

7.8.2 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 `hdfs_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 hdfs_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。

7.9 外国のテーブルをドロップ

外部テーブルは、Hadoop 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 ]
```

7.9.1 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自動的に削除します。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。

7.9.2 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。

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>-hdfs_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>-hdfs_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>-hdfs-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>-hdfs_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、Hadoop Foreign Data Wrapper パッケージをアンインストールします。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>-hdfs_fdw
```

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

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

- SLES の場合：

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

- Debian または Ubuntu の場合

```
apt-get remove edb-as<xx>-hdfs_fdw
```


第10章 Using Apache Hive or Apache Spark

Apache Hive または Apache Spark のいずれかを介して、Hadoop Foreign Data Wrapper を使用できます。Hive と Spark は両方とも、HiveQL を使用してデータベースとテーブルが作成される構成済みのメタストアにメタデータを保存します。

10.1 Hadoop 上で Apache Hive で HDFS FDW を使用する

Apache Hive データウェアハウスソフトウェアは、分散ストレージにある大規模なデータセットのクエリと管理に役立ちます。Hive は、このデータに構造を投影し、HiveQL と呼ばれる SQL に似た言語を使用してデータを照会するメカニズムを提供します。同時に、この言語を使用すると、従来の map/reduce プログラマーは、このロジックを HiveQL で表現するのが不便または非効率な場合に、カスタムのマッパーとリデューサーをプラグインできます。

Hive—HiveServer1 と HiveServer2—の 2 つのバージョンは、[Apache Hive website](#) からダウンロードできます。

注釈: Hadoop Foreign Data Wrapper は HiveServer2 のみをサポートしています。

Hadoop 上で Apache Hive で HDFS FDW を使用するには：

1. [weblogs_parse](#) をダウンロードし、[Wiki Pentaho website](#) の指示に従います。
1. 次のコマンドを使用して `weblog_parse.txt` ファイルをアップロードします。

```
hadoop fs -mkdir /weblogs
hadoop fs -mkdir /weblogs/parse
hadoop fs -put weblog_parse.txt /weblogs/parse/part-000000
```

1. 次のコマンドを使用して、HiveServer をまだ実行していない場合は起動します。

```
$HIVE_HOME/bin/hiveserver2
```

または

```
$HIVE_HOME/bin/hive --service hiveserver2
```

1. hive beeline クライアントを使用して HiveServer2 に接続します。例：

```
$ beeline
Beeline version 1.0.1 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000/default;auth=noSasl
```

1. Hive でテーブルを作成します。この例では、weblogs という名前のテーブルを作成します。

```
CREATE TABLE weblogs (
  client_ip          STRING,
  full_request_date  STRING,
  day                STRING,
  month              STRING,
  month_num          INT,
  year               STRING,
  hour               STRING,
  minute             STRING,
  second             STRING,
  timezone           STRING,
  http_verb          STRING,
  uri                STRING,
  http_status_code   STRING,
  bytes_returned     STRING,
  referrer           STRING,
  user_agent         STRING)
row format delimited
fields terminated by \t;
```

1. テーブルにデータをロードします。

```
hadoop fs -cp /weblogs/parse/part-000000 /user/hive/warehouse/weblogs/
```

1. Postgres からデータにアクセスします。weblog テーブルを使用できるようになりました。psql を使用して接続したら、次の手順に従います。

```
-- set the GUC variables appropriately, e.g. :
hdfs_fdw.jvmpath=/home/edb/Projects/hadoop_fdw/jdk1.8.0_111/jre/lib/amd64/server/
hdfs_fdw.classpath=/usr/local/edb/lib/postgresql/HiveJdbcClient-1.0.jar:/home/edb/
→Projects/hadoop_fdw/hadoop/share/hadoop/common/hadoop-common-2.6.4.jar:/home/edb/
→Projects/hadoop_fdw/apache-hive-1.0.1-bin/lib/hive-jdbc-1.0.1-standalone.jar

-- load extension first time after install
CREATE EXTENSION hdfs_fdw;

-- create server object
CREATE SERVER hdfs_server
  FOREIGN DATA WRAPPER hdfs_fdw
  OPTIONS (host 127.0.0.1);
```

(次のページに続く)

(前のページからの続き)

```

-- create user mapping
CREATE USER MAPPING FOR postgres
SERVER hdfs_server OPTIONS (username hive_username, password hive_password);

-- create foreign table
CREATE FOREIGN TABLE weblogs
(
  client_ip          TEXT,
  full_request_date  TEXT,
  day                TEXT,
  Month              TEXT,
  month_num          INTEGER,
  year               TEXT,
  hour               TEXT,
  minute             TEXT,
  second             TEXT,
  timezone           TEXT,
  http_verb          TEXT,
  uri                TEXT,
  http_status_code   TEXT,
  bytes_returned     TEXT,
  referrer           TEXT,
  user_agent         TEXT
)
SERVER hdfs_server
  OPTIONS (dbname default, table_name weblogs);

-- select from table
postgres=# SELECT DISTINCT client_ip IP, count(*)
          FROM weblogs GROUP BY IP HAVING count(*) > 5000 ORDER BY 1;
   ip      | count
-----+-----
13.53.52.13 |  5494
14.323.74.653 | 16194
322.6.648.325 | 13242
325.87.75.336 |   6500
325.87.75.36  |   6498
361.631.17.30 |  64979
363.652.18.65 |  10561
683.615.622.618 | 13505
(8 rows)

-- EXPLAIN output showing WHERE clause being pushed down to remote server.
EXPLAIN (VERBOSE, COSTS OFF) SELECT client_ip, full_request_date, uri FROM weblogs
→ WHERE http_status_code = 200;

```

(次のページに続く)

QUERY PLAN

```

-----
Foreign Scan on public.weblogs
  Output: client_ip, full_request_date, uri
  Remote SQL: SELECT client_ip, full_request_date, uri FROM default.weblogs WHERE
  ((http_status_code = 200))
(3 rows)

```

10.2 Hadoop 上で Apache Spark で HDFS FDW を使用する

Apache Spark は、さまざまなユースケースをサポートする汎用の分散コンピューティングフレームワークです。リアルタイムのストリーミングと、高速で使いやすい高度な分析を備えたバッチ処理を提供します。Spark は、Hadoop、HBASE、Cassandra、S3 などのサードパーティのストレージプロバイダーに依存しているため、ストレージレイヤーを提供しません。Spark は Hadoop とシームレスに統合し、既存のデータを処理できます。Spark SQL は HiveQL と 100 % の互換性があります。Spark Thrift Server を使用して、Hiveserver2 を置き換えることができます。

Hadoop 上で Apache Spark で HDFS FDW を使用するには：

1. ローカルモードで Apache Spark をダウンロードしてインストールします。

1. フォルダ \$SPARK_HOME/conf に、次の行を含むファイル spark-defaults.conf を作成します。

```
spark.sql.warehouse.dir hdfs://localhost:9000/user/hive/warehouse
```

デフォルトでは、Spark はメタデータとデータ自身（Spark ではウェアハウスと呼ばれます）の両方に derby を使用します。Spark で Hadoop をウェアハウスとして使用するには、このプロパティを追加します。

1. Spark Thrift サーバーを起動します。

```
./start-thriftserver.sh
```

1. Spark Thrift サーバーが実行されており、ログファイルに書き込んでいることを確認します。

1. 次のエントリを含むローカルファイル（names.txt）を作成します。

```

$ cat /tmp/names.txt
1,abcd
2,pqrs
3,wxzy
4,a_b_c
5,p_q_r
,

```

1. Spark beeline クライアントを使用して Spark Thrift Server2 に接続します。例：

```
$ beeline
Beeline version 1.2.1.spark2 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000/default;auth=noSasl org.apache.hive.
↪ jdbc.HiveDriver
```

1. Spark でサンプルデータを準備します。beeline コマンドラインツールで次のコマンドを実行します。

```
./beeline
Beeline version 1.2.1.spark2 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000/default;auth=noSasl org.apache.hive.
↪ jdbc.HiveDriver
Connecting to jdbc:hive2://localhost:10000/default;auth=noSasl
Enter password for jdbc:hive2://localhost:10000/default;auth=noSasl:
Connected to: Spark SQL (version 2.1.1)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000> create database my_test_db;
+-----+
| Result |
+-----+
+-----+
No rows selected (0.379 seconds)
0: jdbc:hive2://localhost:10000> use my_test_db;
+-----+
| Result |
+-----+
+-----+
No rows selected (0.03 seconds)
0: jdbc:hive2://localhost:10000> create table my_names_tab(a int, name string)
                                row format delimited fields terminated by ;
+-----+
| Result |
+-----+
+-----+
No rows selected (0.11 seconds)
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> load data local inpath /tmp/names.txt
                                into table my_names_tab;
+-----+
| Result |
+-----+
+-----+
No rows selected (0.33 seconds)
0: jdbc:hive2://localhost:10000> select * from my_names_tab;
+-----+-----+
(次のページに続く)
```

(前のページからの続き)

a	name
1	abcd
2	pqrs
3	wxyz
4	a_b_c
5	p_q_r
NULL	NULL

次のコマンドは、Hadoop 内の対応するファイルをリストします。

```
$ hadoop fs -ls /user/hive/warehouse/
Found 1 items
drwxrwxrwx - org.apache.hive.jdbc.HiveDriver supergroup 0 2020-06-12 17:03 /user/
→hive/warehouse/my_test_db.db

$ hadoop fs -ls /user/hive/warehouse/my_test_db.db/
Found 1 items
drwxrwxrwx - org.apache.hive.jdbc.HiveDriver supergroup 0 2020-06-12 17:03 /user/
→hive/warehouse/my_test_db.db/my_names_tab
```

1. psql を使用して Postgres からデータにアクセスします。

```
-- set the GUC variables appropriately, e.g. :
hdfs_fdw.jvmpath=/home/edb/Projects/hadoop_fdw/jdk1.8.0_111/jre/lib/amd64/server/
hdfs_fdw.classpath=/usr/local/edb/lib/postgresql/HiveJdbcClient-1.0.jar:/home/edb/
→Projects/hadoop_fdw/hadoop/share/hadoop/common/hadoop-common-2.6.4.jar:/home/edb/
→Projects/hadoop_fdw/apache-hive-1.0.1-bin/lib/hive-jdbc-1.0.1-standalone.jar

-- load extension first time after install
CREATE EXTENSION hdfs_fdw;

-- create server object
CREATE SERVER hdfs_server
  FOREIGN DATA WRAPPER hdfs_fdw
  OPTIONS (host 127.0.0.1, port 10000, client_type spark, auth_type NOSASL);

-- create user mapping
CREATE USER MAPPING FOR postgres
  SERVER hdfs_server OPTIONS (username spark_username, password spark_password);

-- create foreign table
CREATE FOREIGN TABLE f_names_tab( a int, name varchar(255)) SERVER hdfs_svr
  OPTIONS (dbname testdb, table_name my_names_tab);
```

(次のページに続く)

(前のページからの続き)

```
-- select the data from foreign server
select * from f_names_tab;
 a | name
---+-----
 1 | abcd
 2 | pqr
 3 | wxyz
 4 | a_b_c
 5 | p_q_r
 0 |
(6 rows)

-- EXPLAIN output showing WHERE clause being pushed down to remote server.
EXPLAIN (verbose, costs off) SELECT name FROM f_names_tab WHERE a > 3;
          QUERY PLAN
-----
Foreign Scan on public.f_names_tab
Output: name
Remote SQL: SELECT name FROM my_test_db.my_names_tab WHERE ((a > 3))
(3 rows)
```

注釈: Spark Thrift サーバーは Hive Thrift サーバーと互換性があるため、この例では外部サーバーの作成中に同じポートを使用します。Hiveserver2 を使用するアプリケーションは、`ANALYZE` コマンドの動作と `'NOSASL'` の場合の接続文字列を除き、Spark で動作します。Hive を Spark に置き換える場合、`ALTER SERVER` を使用し、`client_type` オプションを変更することをお勧めします。

第11章 Join pushdown

この例は、同じリモート HIVE/SPARK サーバーの外部テーブルとそのリモート HIVE/SPARK サーバー間の結合プッシュダウンを示しています。

HIVE/SPARK サーバー上のテーブル：

```
0: jdbc:hive2://localhost:10000> describe emp;
+-----+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+-----+
| empno    | int       | NULL    |
| ename     | string    | NULL    |
| job       | string    | NULL    |
| mgr       | int       | NULL    |
| hiredate  | date      | NULL    |
| sal       | int       | NULL    |
| comm      | int       | NULL    |
| deptno    | int       | NULL    |
+-----+-----+-----+-----+
8 rows selected (0.747 seconds)
0: jdbc:hive2://localhost:10000> describe dept;
+-----+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+-----+
| deptno   | int       | NULL    |
| dname    | string    | NULL    |
| loc      | string    | NULL    |
+-----+-----+-----+-----+
3 rows selected (0.067 seconds)
```

Postgres サーバー上のテーブル：

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

- - create server object
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw OPTIONS(host localhost, port_
→10000, client_type spark, auth_type LDAP);

- - create user mapping
```

(次のページに続く)

(前のページからの続き)

```

CREATE USER MAPPING FOR public SERVER hdfs_server OPTIONS (username user1, password_
↳pwd123);

-- create foreign table
CREATE FOREIGN TABLE dept (
    deptno          INTEGER,
    dname           VARCHAR(14),
    loc             VARCHAR(13)
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name dept);

-- create foreign table
CREATE FOREIGN TABLE emp (
    empno           INTEGER,
    ename           VARCHAR(10),
    job             VARCHAR(9),
    mgr             INTEGER,
    hiredate        DATE,
    sal             INTEGER,
    comm            INTEGER,
    deptno          INTEGER
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name emp);

```

ジョインプッシュダウンを使用したクエリ：

```

--inner join
edb=# EXPLAIN VERBOSE SELECT t1.ename, t2.dname FROM emp t1 INNER JOIN dept t2 ON (
↳t1.deptno = t2.deptno );
__OUTPUT__

```

QUERY PLAN

```

↳-----
Foreign Scan (cost=15.00..35.00 rows=5000 width=84)
  Output: t1.ename, t2.dname
  Relations: (fdw_db.emp t1) INNER JOIN (fdw_db.dept t2)
  Remote SQL: SELECT r1.`ename`, r2.`dname` FROM (`fdw_db`.`emp` r1 INNER JOIN `fdw_
↳db`.`dept` r2 ON ((r1.`deptno` = r2.`deptno`)))
(4 rows)

```

```

--left join
edb=# EXPLAIN VERBOSE SELECT t1.ename, t2.dname FROM emp t1 LEFT JOIN dept t2 ON ( t1.
↳deptno = t2.deptno );
__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..35.00 rows=5000 width=84)
  Output: t1.ename, t2.dname
  Relations: (fdw_db.emp t1) LEFT JOIN (fdw_db.dept t2)
  Remote SQL: SELECT r1.`ename`, r2.`dname` FROM (`fdw_db`.`emp` r1 LEFT JOIN `fdw_
->db`.`dept` r2 ON (((r1.`deptno` = r2.`deptno`))))
(4 rows)

```

```

- -right join
edb=# EXPLAIN VERBOSE SELECT t1.ename, t2.dname FROM emp t1 RIGHT JOIN dept t2 ON (
->t1.deptno = t2.deptno );
__OUTPUT__

```

QUERY PLAN

```

-----
-> -----
Foreign Scan (cost=15.00..35.00 rows=5000 width=84)
  Output: t1.ename, t2.dname
  Relations: (fdw_db.dept t2) LEFT JOIN (fdw_db.emp t1)
  Remote SQL: SELECT r1.`ename`, r2.`dname` FROM (`fdw_db`.`dept` r2 LEFT JOIN `fdw_
->db`.`emp` r1 ON (((r1.`deptno` = r2.`deptno`))))
(4 rows)

```

```

- -full join
edb=# EXPLAIN VERBOSE SELECT t1.ename, t2.dname FROM emp t1 FULL JOIN dept t2 ON ( t1.
->deptno = t2.deptno );
__OUTPUT__

```

QUERY PLAN

```

-----
-> -----
Foreign Scan (cost=15.00..35.00 rows=5000 width=84)
  Output: t1.ename, t2.dname
  Relations: (fdw_db.emp t1) FULL JOIN (fdw_db.dept t2)
  Remote SQL: SELECT r1.`ename`, r2.`dname` FROM (`fdw_db`.`emp` r1 FULL JOIN `fdw_
->db`.`dept` r2 ON (((r1.`deptno` = r2.`deptno`))))
(4 rows)

```

```

- -cross join
edb=# EXPLAIN VERBOSE SELECT t1.ename, t2.dname FROM emp t1 CROSS JOIN dept t2;
__OUTPUT__

```

QUERY PLAN

```

-----
-> -----
Foreign Scan (cost=15.00..35.00 rows=1000000 width=84)

```

(次のページに続く)

(前のページからの続き)

```

Output: t1.ename, t2.dname
Relations: (fdw_db.emp t1) INNER JOIN (fdw_db.dept t2)
Remote SQL: SELECT r1.`ename`, r2.`dname` FROM (`fdw_db`.`emp` r1 INNER JOIN `fdw_`
→db`.`dept` r2 ON (TRUE))
(4 rows)

```

テーブルレベルでのジョインプッシュダウンクエリの GUC を有効/無効にします。

```

- - enable join pushdown at the table level
ALTER FOREIGN TABLE emp OPTIONS (SET enable_join_pushdown true);
EXPLAIN (VERBOSE, COSTS OFF)
SELECT e.empno, e.ename, d.dname
  FROM emp e JOIN dept d ON (e.deptno = d.deptno)
  ORDER BY e.empno;
__OUTPUT__

```

QUERY PLAN

```

→-----
Sort
  Output: e.empno, e.ename, d.dname
  Sort Key: e.empno
  -> Foreign Scan
    Output: e.empno, e.ename, d.dname
    Relations: (fdw_db.emp e) INNER JOIN (fdw_db.dept d)
    Remote SQL: SELECT r1.`empno`, r1.`ename`, r2.`dname` FROM (`fdw_db`.`emp`
→r1 INNER JOIN `fdw_db`.`dept` r2 ON (((r1.`deptno` = r2.`deptno`))))
(7 rows)

```

```

- -Disable the GUC enable_join_pushdown.
SET hdfs_fdw.enable_join_pushdown to false;
- - Pushdown shouldnt happen as enable_join_pushdown is false.
EXPLAIN (VERBOSE, COSTS OFF)
SELECT e.empno, e.ename, d.dname
  FROM emp e JOIN dept d ON (e.deptno = d.deptno)
  ORDER BY e.empno;
__OUTPUT__

```

QUERY PLAN

```

→-----
Sort
  Output: e.empno, e.ename, d.dname
  Sort Key: e.empno
  -> Nested Loop
    Output: e.empno, e.ename, d.dname
    Join Filter: (e.deptno = d.deptno)

```

(次のページに続く)

(前のページからの続き)

```

-> Foreign Scan on public.emp e
      Output: e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.
->deptno
      Remote SQL: SELECT `empno`, `ename`, `deptno` FROM `fdw_db`.`emp`
-> Materialize
      Output: d.dname, d.deptno
-> Foreign Scan on public.dept d
      Output: d.dname, d.deptno
      Remote SQL: SELECT `deptno`, `dname` FROM `fdw_db`.`dept`

```

セッションレベルで参加プッシュダウンクエリの GUC を有効/無効にします。

```

SET hdfs_fdw.enable_join_pushdown to true;
EXPLAIN (VERBOSE, COSTS OFF)
SELECT e.empno, e.ename, d.dname
FROM emp e JOIN dept d ON (e.deptno = d.deptno)
ORDER BY e.empno;
__OUTPUT__

```

QUERY PLAN

```

-> -----
Sort
      Output: e.empno, e.ename, d.dname
      Sort Key: e.empno
-> Nested Loop
      Output: e.empno, e.ename, d.dname
      Join Filter: (e.deptno = d.deptno)
-> Foreign Scan on public.emp e
      Output: e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.
->deptno
      Remote SQL: SELECT `empno`, `ename`, `deptno` FROM `fdw_db`.`emp`
-> Materialize
      Output: d.dname, d.deptno
-> Foreign Scan on public.dept d
      Output: d.dname, d.deptno
      Remote SQL: SELECT `deptno`, `dname` FROM `fdw_db`.`dept`
(14 rows)

```


第12章 Aggregate pushdown

この例は、同じリモート HIVE/SPARK サーバーの外部テーブルからそのリモート HIVE/SPARK サーバーへの集計プッシュダウンを示しています。

HIVE/SPARK サーバー上のテーブル：

```
0: jdbc:hive2://localhost:10000> describe emp;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| empno    | int       | NULL    |
| ename     | string    | NULL    |
| job       | string    | NULL    |
| mgr       | int       | NULL    |
| hiredate  | date      | NULL    |
| sal       | int       | NULL    |
| comm      | int       | NULL    |
| deptno    | int       | NULL    |
+-----+-----+-----+
8 rows selected (0.747 seconds)
0: jdbc:hive2://localhost:10000> describe dept;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| deptno   | int       | NULL    |
| dname    | string    | NULL    |
| loc      | string    | NULL    |
+-----+-----+-----+
3 rows selected (0.067 seconds)
```

Postgres サーバー上のテーブル：

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

- - create server object
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw OPTIONS(host localhost, port_
→10000, client_type spark, auth_type LDAP);

- - create user mapping
```

(次のページに続く)

(前のページからの続き)

```

CREATE USER MAPPING FOR public SERVER hdfs_server OPTIONS (username user1, password_
↳pwd123);

-- create foreign table
CREATE FOREIGN TABLE dept (
    deptno          INTEGER,
    dname            VARCHAR(14),
    loc              VARCHAR(13)
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name dept);

CREATE FOREIGN TABLE emp (
    empno            INTEGER,
    ename             VARCHAR(10),
    job               VARCHAR(9),
    mgr               INTEGER,
    hiredate          DATE,
    sal               INTEGER,
    comm              INTEGER,
    deptno            INTEGER
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name emp);

```

集計プッシュダウンを使用したクエリ：

```

-- aggregate functions
EXPLAIN (VERBOSE, COSTS OFF)
SELECT deptno, COUNT(*),SUM(sal),MAX(sal),MIN(sal),AVG(sal) FROM emp
GROUP BY deptno
HAVING deptno IN (10,20)
ORDER BY deptno;
__OUTPUT__
QUERY PLAN

-- -----
↳
Sort
  Output: deptno, (count(*)), (sum(sal)), (max(sal)), (min(sal)), (avg(sal))
  Sort Key: emp.deptno
  -> Foreign Scan
    Output: deptno, (count(*)), (sum(sal)), (max(sal)), (min(sal)), (avg(sal))
    Relations: Aggregate on (k_test.emp)
    Remote SQL: SELECT `deptno`, count(*), sum(`sal`), max(`sal`), min(`sal`),
↳avg(`sal`) FROM `k_test`.`emp` WHERE (`deptno` IN (10,20)) GROUP BY `deptno`
(7 rows)

```

第13章 ORDER BY pushdown

この例は、リモート HIVE/SPARK サーバーと同じリモート HIVE/SPARK サーバーの外部テーブル間の ORDER BY プッシュダウンを示しています。

HIVE/SPARK サーバー上のテーブル：

```
0: jdbc:hive2://localhost:10000> describe emp;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| empno    | int       | NULL    |
| ename     | string    | NULL    |
| job       | string    | NULL    |
| mgr       | int       | NULL    |
| hiredate  | date      | NULL    |
| sal       | int       | NULL    |
| comm      | int       | NULL    |
| deptno    | int       | NULL    |
+-----+-----+-----+
8 rows selected (0.747 seconds)
0: jdbc:hive2://localhost:10000> describe dept;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| deptno   | int       | NULL    |
| dname     | string    | NULL    |
| loc       | string    | NULL    |
+-----+-----+-----+
3 rows selected (0.067 seconds)
```

Postgres サーバー上のテーブル：

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

- - create server object
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw OPTIONS(host localhost, port_
→10000, client_type spark, auth_type LDAP);

- - create user mapping
```

(次のページに続く)

(前のページからの続き)

```
CREATE USER MAPPING FOR public SERVER hdfs_server OPTIONS (username user1, password_
↳pwd123);

-- create foreign table
CREATE FOREIGN TABLE emp (
    empno          INTEGER,
    ename          VARCHAR(10),
    job            VARCHAR(9),
    mgr            INTEGER,
    hiredate       DATE,
    sal            INTEGER,
    comm           INTEGER,
    deptno         INTEGER
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name emp);
```

ORDER BY プッシュダウンを使用したクエリ：

```
edb=# SET hdfs_fdw.enable_order_by_pushdown TO ON;
SET
edb=# EXPLAIN (COSTS OFF) SELECT * FROM emp order by deptno;
__OUTPUT__
      QUERY PLAN
-----
Foreign Scan on emp
(1 row)

edb=# SET hdfs_fdw.enable_order_by_pushdown TO OFF;
SET
edb=# EXPLAIN (COSTS OFF) SELECT * FROM emp order by deptno;
__OUTPUT__
      QUERY PLAN
-----
Sort
Sort Key: deptno
-> Foreign Scan on emp
(3 rows)
```

第14章 LIMIT OFFSET pushdown

この例は、リモート HIVE/SPARK サーバーと同じリモート HIVE/SPARK サーバーの外部テーブル間の LIMIT OFFSET プッシュダウンを示しています。

HIVE / SPARK サーバー上のテーブル：

```
0: jdbc:hive2://localhost:10000> describe emp;
+-----+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+-----+
| empno    | int       | NULL    |
| ename    | string    | NULL    |
| job      | string    | NULL    |
| mgr      | int       | NULL    |
| hiredate | date      | NULL    |
| sal      | int       | NULL    |
| comm     | int       | NULL    |
| deptno   | int       | NULL    |
+-----+-----+-----+-----+
8 rows selected (0.747 seconds)
0: jdbc:hive2://localhost:10000> describe dept;
+-----+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+-----+
| deptno   | int       | NULL    |
| dname    | string    | NULL    |
| loc      | string    | NULL    |
+-----+-----+-----+-----+
3 rows selected (0.067 seconds)
```

Postgres サーバー上のテーブル：

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

- - create server object
CREATE SERVER hdfs_server FOREIGN DATA WRAPPER hdfs_fdw OPTIONS(host localhost, port_
→10000, client_type spark, auth_type LDAP);

- - create user mapping
```

(次のページに続く)

(前のページからの続き)

```

CREATE USER MAPPING FOR public SERVER hdfs_server OPTIONS (username user1, password_
→pwd123);

-- create foreign table
CREATE FOREIGN TABLE emp (
    empno          INTEGER,
    ename          VARCHAR(10),
    job            VARCHAR(9),
    mgr            INTEGER,
    hiredate       DATE,
    sal            INTEGER,
    comm           INTEGER,
    deptno         INTEGER
)
SERVER hdfs_server OPTIONS (dbname fdw_db, table_name emp);

```

LIMIT OFFSET プッシュダウンを使用したクエリ：

```

-- LIMIT OFFSET
EXPLAIN (VERBOSE, COSTS OFF)
SELECT empno FROM emp e ORDER BY empno LIMIT 5 OFFSET 2;
__OUTPUT__

          QUERY PLAN
-----
→ Foreign Scan on public.emp e
   Output: empno
  Remote SQL: SELECT `empno` FROM `fdw_db`.`emp` ORDER BY `empno` ASC NULLS LAST_
→LIMIT 2, 5
(3 rows)

SELECT empno FROM emp e ORDER BY empno LIMIT 5 OFFSET 2;
__OUTPUT__
 empno
-----
 7521
 7566
 7654
 7698
 7782
(5 rows)

```

第15章 Identifying the version

Hadoop 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 hdfs_fdw_version();
```

この関数はバージョン番号を返します。

```
hdfs_fdw_version
-----
<xxxxxx>
```