



Hadoop Foreign Data Wrapper Guide

リリース *2.0.5*

Hadoop Foreign Data Wrapper Guide

2020 年 08 月 25 日

目次

第 1 章	要件の概要	2
1.1	サポートされているバージョン	2
1.2	サポートされているプラットフォーム	2
第 2 章	アーキテクチャの概要	3
第 3 章	サポートされている認証方法	5
3.1	LDAP 認証を使用する	5
3.2	NOSASL 認証の使用	7
第 4 章	Hadoop 外部データラッパーのインストール	8
4.1	RPM パッケージを使用した Hadoop 外部データラッパーのインストール	8
4.2	Debian または Ubuntu ホストへの Hadoop 外部データラッパーのインストール	10
第 5 章	Hadoop 外部データラッパーの設定	11
5.1	エクステンションを作成	12
5.2	サーバーの作成	13
5.3	ユーザーマッピングの作成	16
5.4	外部テーブルの作成	18
5.4.1	データ型マッピング	21
第 6 章	Hadoop 外部データラッパーの使用	22
6.1	Hadoop 上の ApacheHive で HDFSFDW を使用する	22
6.2	Hadoop 上の ApacheSpark で HDFSFDW を使用する	26
第 7 章	Hadoop 外部データラッパーバージョンの特定	30
第 8 章	結論	31
索引		33

Hadoop 外部データラッパー (hdfs_fdw) は、AdvancedServer または PostgreSQL サーバーから Hadoop ファイルシステムにあるデータにアクセスできるようにする Postgres 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読み取り専用のデータソースになり、Postgres の機能とユーティリティで使用したり、Postgres ホストにある他のデータと組み合わせて使用したりできます。

HadoopForeignDataWrapper は、RPM パッケージとともにインストールできます。インストーラーは、EDBWeb サイト<<https://www.enterprisedb.com/software-downloads-postgres/>>からダウンロードできます。

このガイドでは、Postgres という用語を使用して、EDBPostgresAdvancedServer または PostgreSQL のインスタンスを指します。

要件の概要

1.1 サポートされているバージョン

HadoopForeignDataWrapper は、PostgreSQL および EDBPostgresAdvancedServer9.5 以上で認定されています。

1.2 サポートされているプラットフォーム

HadoopForeignDataWrapper は、次のプラットフォームでサポートされています。

Linuxx86-64

- RHEL8.x/7.x/6.x
- CentOS8.x/7.x/6.x
- OEL8.x/7.x/6.x
- Ubuntu18.04+
- Debian9.6.x

IBMPower8/9 (LE) 上の Linux

- RHEL7.x

HadoopForeignDataWrapper は、HiveServer2 インターフェイスを使用する Hadoop ファイルシステムまたは SparkThrift サーバーを使用する ApacheSpark の使用をサポートしています。

第 2

アーキテクチャの概要

Hadoop は、分散ファイルシステムに大規模なデータセットを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 の SELECT ステートメントを、HiveQL または SparkSQL インターフェイスで認識されるクエリに変換します。

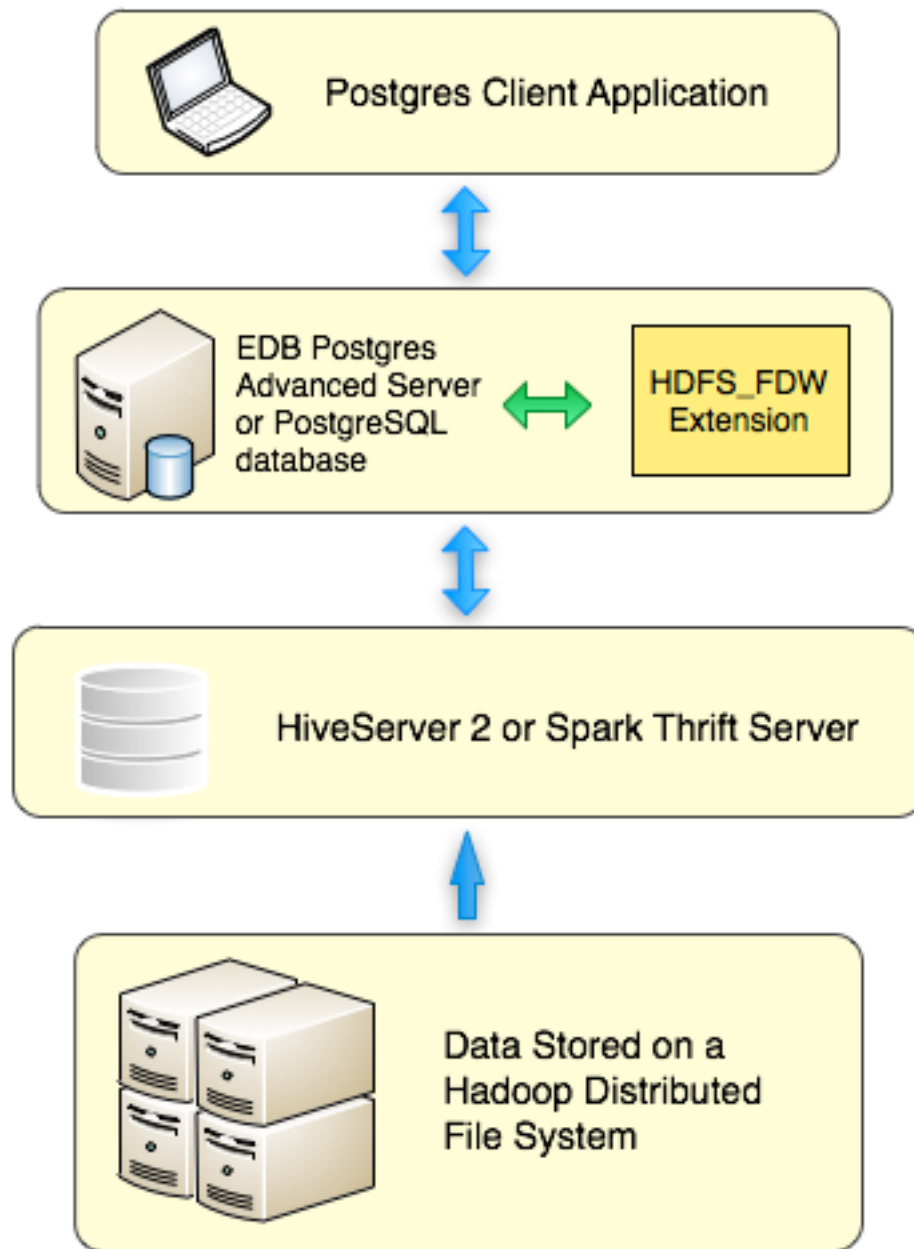


図 1 *Postgres* で *Hadoop* 分散ファイルシステムを使用する

When possible, the Foreign Data Wrapper asks the Hive or Spark server to perform the actions associated with the `WHERE` clause of a `SELECT` statement. Pushing down the `WHERE` clause improves performance by decreasing the amount of data moving across the network.

第 3

サポートされている認証方法

The Hadoop Foreign Data Wrapper supports NOSASL and LDAP authentication modes. To use NOSASL, do not specify any OPTIONS while creating user mapping. For LDAP authentication mode, specify username and password in OPTIONS while creating user mapping.

3.1 LDAP 認証を使用する

HadoopForeignDataWrapper を LDAP 認証で使用する場合、最初に Hive Server または Spark Server を設定して LDAP 認証を使用する必要があります。設定されたサーバーは、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 [nosasl, 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>
```

(次のページに続く)

(前のページからの続き)

```
</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は `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場合、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、[:ref:ユーザーマッピングの作成<create_user_mapping>](#)を参照してください。

3.2 NOSASL 認証の使用

Hadoop 外部データラッパーで NOSASL 認証を使用する場合は、Hive Server または Spark Server で認証を 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 Server 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
```

Hadoop 外部データラッパーのインストール

HadoopForeignDataWrapper は、RPM パッケージとともにインストールできます。インストールプロセス中、インストーラーはソフトウェアの前提条件を満たします。

4.1 RPM パッケージを使用した Hadoop 外部データラッパーのインストール

HadoopForeignDataWrapper の RPM インストールパッケージは、EDB リポジトリから入手できます。foreign data wrapper をインストールする前に、epel-release パッケージをインストールする必要があります：

- RHEL または CentOS7 の場合：

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

- RHEL または CentOS8 の場合：

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

CentOS-Base.repo ファイル（/etc/yum.repos.dにある）で [extras] リポジトリ定義を有効にする必要があるかもしれないことに注意してください。

EDB リポジトリへのアクセスを許可する資格情報も必要です。資格情報のリクエストについては、次をご覧ください。

<https://info.enterprisedb.com/rs/069-ALB-339/images/Repository%20Access%202004-09-2019.pdf>

リポジトリ設定ファイルの作成

リポジトリ構成ファイルを作成するには、スーパーユーザー特権を引き受けて、次のコマンドのいずれかを呼び出します。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

リポジトリ設定ファイルの名前は `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。

`edb.repo` ファイルを作成した後、エディターの選択を使用して、1 パラメーターの値が 1 であることを確認し、`username` および `password` プレースホルダーを置き換えます登録された EnterpriseDB ユーザーの名前とパスワードを含む `baseurl` 仕様。

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

設定ファイルへの変更を保存した後、`yum install` コマンドを使用して RHEL または CentOS7 に外部データラッパーをインストールできます。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

ここで、“`xx`”はサーバーのバージョン番号です。

システムで認識されないソースによって署名された RPM パッケージをインストールすると、`yum` はローカルサーバーにキーをインポートする許可を求め場合があります。プロンプトが表示され、パッケージが信頼できるソースからのものであることに満足したら、`y` を入力し、Return を押して続行します。

インストール中に、`yum` は解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある必要な依存関係のリストを提供します。

4.2 Debian または Ubuntu ホストへの Hadoop 外部データラッパーのインストール

Debian または Ubuntu ホストに HadoopForeignDataWrapper をインストールするには、EDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格情報をリクエストするには、EDBWeb サイト <<https://www.enterprisedb.com/repository-access-request/>> にアクセスしてください。

次の手順では、EDBapt リポジトリを使用して DEB パッケージをインストールする方法を説明します。コマンドを使用する場合、username と password を EDB が提供する資格情報に置き換えます。

1. スーパーユーザー特権を想定します。

```
sudo su -
```

2. EnterpriseDB リポジトリを構成します。

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

3. 安全な APT リポジトリのサポートをシステムに追加します。

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

4. EBD 署名キーを追加します。

```
wget -q -O - https://username:password
@apt.enterprisedb.com/edb-deb.gpg.key \& apt-key add -
```

5. リポジトリのメタデータを更新します。

```
apt-get update
```

6. DEB パッケージをインストールします。

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

ここで、“xx”はサーバーのバージョン番号です。

第 5

Hadoop 外部データラッパーの設定

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、サポートライブラリの場所を指定して、Postgres ホストを変更する必要があります。

1. Postgres のインストール後、次の場所にある `postgresql.conf` を変更します。

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

2. 選択したエディターで構成ファイルを変更し、構成ファイルの最後に `hdfs_fdw.jvm_path` パラメーターを追加し、値を設定して Java 仮想マシンの場所（`libjvm.so`）を指定します。アダプターが使用する Java クラスファイルの場所を示すために、`hdfs_fdw.classpath` の値を設定します。各パス間の区切り文字としてコロンの（`:`）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: EDBAdvancedServer を使用していて、データベースに DATE 列がある場合、`postgresql.conf` ファイルに `edb_redwood_date = OFF` を設定する必要があります。

1. パラメータ値を設定した後、Postgres サーバーを再起動します。AdvancedServer ホストでのサービスの制御の詳細については、次の場所にある『EDBPostgresAdvancedServer インストールガイド』を参照してください。

<https://www.enterprisedb.com/resources/product-documentation>

HadoopForeignDataWrapper を使用する前に、以下を行う必要があります。

1. `:ref:CREATEEXTENSION<create_extension>` 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. `:ref:CREATESERVER<create_server>` 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. `:ref:CREATEUSERMAPPING<create_user_mapping>` 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. `:ref:CREATEFOREIGNTABLE<create_foreign_table>` コマンドを使用して、Hadoop クラスタに存在するデータベースに対応する AdvancedServer データベースのテーブルを定義します。

5.1 エクステンションを作成

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 サーバーまたは Spark サーバーのクエリを実行する 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];
```

パラメータ

IF NOT EXISTS

同じ名前の拡張子が既に存在する場合、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 IF NOT EXISTS 句を含めます。

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。

例

次のコマンドは `hdfs_fdw` `hadoop` 外部データラッパー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、以下を参照してください。

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

5.2 サーバーの作成

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役割は、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。

パラメータ

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

クラスターに接続するときにサーバーが hdfs_fdw 外部データラッパーを使用するように指定するには、FOREIGN_DATA_WRAPPER 句を含めます。

OPTIONS

CREATE SERVER コマンドの OPTIONS 節を使用して、外部サーバーの接続情報を指定します。以下を含めることができます。

オプション	説明
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 ‘localhost’ です。
port	HiveThriftServer または SparkThriftServer のポート番号。デフォルトは ‘10000’ です。
client_type	クライアントタイプとして hiveserver2 または spark を指定します。Spark で ANALYZE ステートメントを使用するには、spark の値を指定する必要があります。client_type の値を指定しない場合、デフォルト値は hiveserver2 です。
auth_type	クライアントの認証タイプ。LDAP または NOSASL を指定します。auth_type 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 auth_type 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。
fetch_size	JDBC API setFetchSize のパラメーターとして提供されるユーザー指定の値。デフォルト値は ‘10,000’ です。
log_remote_sql	true 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る回数が含まれます。デフォルトは ‘false’ です。
query_timeout	query_timeout を使用して、リクエストが Hive サーバーによって満たされない場合にリクエストがタイムアウトするまでの秒数を指定します。クエリタイムアウトは、HiveJDBC ドライバーではサポートされていません。
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。

例

次のコマンドは、hdfs_fdw の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、以下を参照してください。

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

5.3 ユーザーマッピングの作成

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、外部サーバーの所有者である必要があります。

注意:HadoopForeignDataWrapper 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する場合は、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）を提供します。サーバーが NOSASL 認証を使用する場合、ユーザーマッピングの作成時に OPTIONS 句を省略します。

パラメータ

role_name

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

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するには、server_name を使用します。

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、以下を提供します。

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。

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の作成時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 'lsafepwd');
```

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、以下を参照してください。

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

5.4 外部テーブルの作成

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 ホストに存在するテーブルに対応する列を持つ Postgres サーバー上のテーブルを定義します。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' [, ... ] ) ]
```

ここで、column_constraint は次のとおりです。

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

そして table_constraint は:

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

パラメータ

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

同じ名前のテーブルが既に存在する場合にエラーをスローしないようにサーバーに指示するには、IF NOT EXISTS 句を含めます。同じ名前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する必要があります。

data_type

列のデータ型を指定します。可能であれば、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 サーバーはデータ型を Hive または Spark 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

列に照合順序を割り当てるには、COLLATE 句を含めます。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、プレーンテーブルでも外部テーブルでもかまいません。

CONSTRAINT constraint_name

列制約または表制約のオプション名を指定します。指定しない場合、サーバーは制約名を生成します。

NOT NULL

カラムにヌル値を含めることが許可されていないことを示すために、NOT NULL キーワードを含めます。

NULL

列にヌル値を含めることが許可されていることを示すために、NULL キーワードを含めます。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のあるブール結果を生成する式を指定します。列制約として指定されたチェック制約はその列の値のみを参照する必要がありますが、表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、サブクエリを含めることも、現在の行の列以外の変数を参照することもできません。

制約が子テーブルに伝播しないように指定するには、NO INHERIT キーワードを含めます。

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 を指定します。

OPTIONS 句を使用して、次の options とそれに対応する値を指定します。

オプション	説明
dbname	Hive サーバー上のデータベースの名前。データベース名が必要です。
table_name	Hive サーバー上のテーブルの名前。デフォルトは外部テーブルの名前です。

例

分散ファイルシステムに格納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 Postgres サーバーで実行する必要があります。

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

Hadoop ファイルシステムに格納されているデータベースの名前（webdata）および Postgres サーバーのテーブルに対応するテーブルの名前（weblogs）を指定するには、SERVER 句を含めます。

CREATE FOREIGN TABLE コマンドの使用の詳細については、以下を参照してください。

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

5.4.1 データ型マッピング

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 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

Hadoop 外部データラッパーの使用

ApacheHive または ApacheSpark を介して HadoopForeignDataWrapper を使用できます。Hive と Spark は両方とも、構成されたメタストアにメタデータを格納します。ここで、データベースとテーブルは HiveQL を使用して作成されます。

6.1 Hadoop 上の ApacheHive で HDFSFDW を使用する

Apache Hive™ data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time, this language allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Hive には 2 つのバージョンがあります- HiveServer1 と HiveServer2 は 'ApacheHive ウェブサイト<<https://hive.apache.org/downloads.html>>' からダウンロードできます。

注釈: HadoopForeignDataWrapper は HiveServer2 のみをサポートします。

Hadoop の上で ApacheHive で HDFSFDW を使用するには:

ステップ 1: 'weblogs_parse<http://wiki.pentaho.com/download/attachments/23531451/weblogs_parse.zip?version=1&modificationDate=1388888888000>' をダウンロードし、'WikiPentaho ウェブサイト<<https://wiki.pentaho.com/display/BAD/Transforming+Data+within+Hive/>>' 。

ステップ 2: これら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 weblogs_parse.txt /weblogs/parse/part-00000
```

ステップ 3: 次のコマンドを使用して、HiveServer を開始します（まだ実行されていない場合）。

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

または

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

ステップ 4: Hive ビーライン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
```

ステップ 5: Hive で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';
```

ステップ 6: ウェブログテーブルにデータをロードします。

```
hadoop fs -cp /weblogs/parse/part-00000 /user/hive/warehouse/weblogs/
```

ステップ 7: PostgreSQL からデータにアクセスします。これで、PostgreSQL のウェブログテーブルを使用できます。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');

-- 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
```

(次のページに続く)

(前のページからの続き)

```

-----+-----
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;

                                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)

```

6.2 Hadoop 上の ApacheSpark で HDFSFDW 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。リアルタイムストリームとバッチ処理を、スピード、使いやすさ、高度な分析とともに提供します。Spark は、Hadoop、HBASE、Cassandra、S3 などのサードパーティ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。SparkSQL は、HiveQL と 100 % 互換性があり、Spark Thrift Server を使用して Hiveserver2 の代替として使用できます。

Hadoop 上で ApacheSpark で HDFSFDW を使用するには:

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

ステップ 2: フォルダー \$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体の両方にダービーを使用します (spark ではウェアハウスと呼ばれます)。Spark でウェアハウスとして hadoop を使用するには、このプロパティを追加する必要があります。

ステップ 3: SparkThriftServer を起動します。

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

ステップ 4: ログファイルを使用して、Sparkthrift サーバーが実行されていることを確認します。

ステップ 5: 以下のデータでローカルファイル names.txt を作成します。

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

ステップ 6: sparkbeeline クライアントを使用して SparkThriftServer2 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
```

ステップ 7: 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;
+-----+-----+---+
| a      | name  |
+-----+-----+---+
| 1      | abcd  |
| 2      | pqrs  |
| 3      | wxyz  |
| 4      | a_b_c |
| 5      | p_q_r |
| NULL   | NULL  |
+-----+-----+---+

```

Hadoop の対応するファイルは次のとおりです。

```
$ 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
```

ステップ 8:PostgreSQL からデータにアクセスします。psql を使用して Postgres に接続します。

```
-- set the GUC variables appropriately, e.g. :
hdfs_fdw.jvmppath='/home/edb/Projects/hadoop_fdw/jdk1.8.0_111/jre/lib/amd64/server/'
hdfs_fdw.classpath='/usr/local/edb/pgsql/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');

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

(次のページに続く)

(前のページからの続き)

```
-- 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: SparkThriftServer は HiveThriftServer と互換性があるため、外部サーバーの作成中に同じポートが使用されていました。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 に置き換える場合は、ALTER SERVER を使用して client_type オプションを変更することをお勧めします。

Hadoop 外部データラッパーバージョンの特定

HadoopForeignDataWrapper には、データラッパーの .so ファイルの現在インストールされている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();
```

関数はバージョン番号を返します。

```
hdfs_fdw_version
-----
20005
```

第 8

結論

Hadoop 外部データラッパーガイド

Copyright©2017-2020EnterpriseDBCorporation。

無断複写・転載を禁じます。

EnterpriseDB®Corporation

34CrosbyDrive、Suite201、Bedford、MA01730、アメリカ

T+17813573390F+19784671307E

info@enterprisedb.com

www.enterprisedb.com

- EnterpriseDB および PostgresEnterpriseManager は、EnterpriseDBCorporation の登録商標です。EDB および EDBPostgres は、EnterpriseDBCorporation の商標です。Oracle は、Oracle、Inc の登録商標です。その他の商標は、それぞれの所有者の商標です。
- EDB は、存在する場合、EDBPostgres 製品のログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および入力検証の検証を行います。EDB は追加の入力コンポーネントに対して同じアプローチに従いますが、製品の性質上、限定された検証が可能な信頼できるユーザーによって入力および送信されるフリーフォーム SQL、WMI またはその他の文字列を受け入れる必要がある場合があります。このような場合、ユーザーが誤った入力やその他の危険な入力を防ぐことはできません。
- EDB は、将来、認証された信頼できるユーザーからのフリーフォーム SQL、WMI、またはその他の潜在的に危険な入力を受け入れる製品に機能を追加する権利を留保しますが、そのような機能はすべて、可能な限

り最小のリスクを提供するように設計およびテストされます可能であれば、スーパーユーザーまたは同等の特権が必要です。

- EDB は、すべての潜在的な脅威を予測できることを保証しません。したがって、すべての潜在的な脆弱性が対処または検討されたことをプロセスが完全に保証することはできません。

索引

Hadoop 外部データラッパーのインストール, 8
Hadoop 外部データラッパーの使用, 22
Hadoop 外部データラッパーの設定, 11
Hadoop 外部データラッパーバージョンの特定, 30

Installing Hadoop Data Adapter on Debian, 10
Installing Hadoop Data Adapter on Ubuntu, 10

supported platforms, 2
supported server versions, 2

アーキテクチャの概要, 3

サポートされている認証方法, 5

結論, 31
要件の概要, 2