Mpp数据库平台扩展柜

MPP平台扩展框架(PXF)

MPP平台扩展框架(PXF) 通过内置连接器提供跨异构数据源的并行、高吞吐量的数据访问和联合跨异构数据源的查询，该连接器将MPP数据库外部表定义映射到外部数据源。PXF源于Apache HAWQ项目。

- PXF介绍
  本主题介绍PXF概念和用法。
- PXF管理手册
  介绍了PXF的管理，包括安装，配置，初始化，升级和管理过程。
- 使用PXF访问hadoop
  介绍了PXF Hadoop连接器，它们支持的数据类型以及可用于读取和写入HDFS的配置文件。
- 使用PXF访问Azure，Google云端存储，Minio和S3对象存储
  介绍了PXF对象存储连接器，它们支持的数据类型以及可用于从对象库读取数据和向对象库写入数据的配置文件。
- 使用PXF访问SQL数据库(JDBC)
  介绍如何使用PXF JDBC连接器读取和写入外部SQL数据库，如Postgres或MySQL等。
- PXF故障排除
  介绍了PXF的服务和数据库级日志记录配置过程。它还标识了一些常见的PXF错误，并描述了如何解决PXF内存问题。
- PXF实用程序手册
  介绍了工具命令的使用信息

PXF介绍

In this topic:

- 架构概述
- 连接器(Connector),服务器(Servers)和配置文件(Profiles)
- 创建一个外部表
- PXF的其他特性

MPP Platform Extension Framework(PXF)提供的连接器(connectors)可以用于访问存储在MPP数据库外部源中的数据。这些连接器将外部数据源映射到MPP数据库的外部表(external table)中。创建MPP数据库外部表时，你可以通过在命令中提供服务器名称和配置文件名称来标识外部数据存储和数据格式。

您可以通过MPP数据库查询外部表引用的数据,您也可以使用外部表将数据加载到MPP数据库中以获得更高的性能。

架构概述

GPDB集群包含一个master节点(master node)和多个segment主机(segment host)。GPDB segment主机上的PXF客户端进程为对外部表进行查询的每个segment instance分配工作线程。多个segment主机的PXF代理与外部数据存储并行通信。

连接器(Connector),服务器(Servers)和配置文件(Profiles)

连接器是一个通用的术语,他封装了读取和写入外部数据存储所需要的实现细节。PXF提供创建了与Hadoop (HDFS,Hive,Hbase),对象存储(Azure,Google Cloud Storage, Minio, S3)和sql数据库(通过jdbc)的连接器。

PXF服务器是连接器的命名配置。服务器定义提供PXF访问外部数据源所需的信息。此配置信息是特定于数据存储的，并且可以包括服务器位置，访问凭据和其他相关属性。

MPP数据库管理员将为每个允许MPP数据库用户访问的外部数据存储配置至少一个服务器定义，并在适当时发布可用的服务器名称。

默认的PXF服务是default(保留)，在没有配置SERVER=时提供外部数据源的位置和访问信息。 创建外部表时，可以指定 `SERVER=<server_name>` 设置，以标识从中获取配置的服务器配置和访问外部数据存储的凭据。

GPDB数据库管理员将为每个允许GPDB用户访问的外部数据存储配置至少一个服务定义,并将根据需求发布可用的服务名。 默认的PXF服务器名为 `default` （保留），在配置后，如果没有 `SERVER=<server_name>` 设置，则将提供外部数据源的位置和访问信息。

最后，PXF配置文件是一个命名映射，用于标识特定外部数据存储支持的特定数据格式或协议。PXF支持
text，Avro，JSON，RCFile，Parquet，SequenceFile和ORC数据格式以及JDBC协议，并提供了一些内置配置文件，如以下部分所述。

创建一个外部表

PXF实现了一个叫做pxf的GPDB协议,你可以使用这个协议去创建一个外部表。指定pxf协议的 `CREATE EXTERNAL TABLE` 命令语法如下:

```
CREATE [WRITABLE] EXTERNAL TABLE <table_name>
        ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION('pxf://<path-to-data>?PROFILE=<profile_name>[&SERVER=<serve
FORMAT '[TEXT|CSV|CUSTOM]' (<formatting-properties>);
```

在创建语句 `CREATE EXTERNAL TABLE` 中的 `LOCATION` 子句是一个URI。这个URI标识描述外部数据位置的路径和其他信息。例如:如果外部数据存储的是HDFS,则填写指定HDFS文件的绝对路径。如果外部数据存储的是HIVE，则需要指定符合模式的HIVE表名称。

使用问号(?)引入的URI的查询部分来标识PXF服务器和配置文件名称。

PXF可能需要额外的信息来读取和写入某些数据格式，可以使用LOCATION字符串的可选组件 = 来提供配置文件的信息，并通过字符串的组件提供格式信息。

Table 1. CREATE EXTERNAL TABLE参数值和描述

Keyword	Value and Description
<path-to-data>	目录，文件名，通配符模式，表名等。的语法取决于外部数据源
PROFILE=<profile_name>	PXF用于访问数据的配置文件。 PXF支持Hadoop services, object stores, and other SQL databases.
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的; 如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	配置文件或服务器支持的其他选项及其值
FORMAT <value>	PXF支持 `TEXT` , `CSV` 和 `CUSTOM` 格式
<formatting-properties>	格式化配置文件支持的属性; 例如, `FORMATTER` 或 `delimiter`

Note: 创建PXF外部表时，不能在格式化程序规范中使用HEADER选项

PXF的其他特性

某些PXF连接器和配置文件支持谓词下推和列投影。 有关此支持的详细信息，请参阅以下主题： - About PXF Filter Pushdown - About Column Projection in PXF

PXF谓词下推

PXF支持谓词下推。启用谓词下推时,可以提取SELECT查询中WHERE子句的约束并将其传递到外部数据源进行过滤。此过程可以提高查询性能，还可以减少传输到MPP数据库的数据量。

通过设置 `gp_external_enable_filter_pushdown` 服务器配置参数，可以为所有外部表协议(包括pxf)启用或禁用谓词下推。此配置参数的默认值为on;将其设置为关闭以禁用谓词下推。例如：

```
SHOW gp_external_enable_filter_pushdown;
SET gp_external_enable_filter_pushdown TO 'on';
```

Note: 某些外部数据源不支持谓词下推。此外，某些数据类型或运算符可能不支持谓词下推。如果查询访问不支持谓词下推的数据源，则执行查询时不进行谓词下推(数据传输到MPP数据库后进行过滤)

PXF使用不同的连接器访问数据源，谓词下推支持由特定的连接器实现方式决定。以下PXF连接器支持谓词下推：

- Hive Connector, all profiles
- HBase Connector

- JDBC Connector
- S3 Connector using the Amazon S3 Select service to access CSV and Parquet data

PXF谓词下推可与这些数据类型一起使用(特定连接器)：

- `INT2`, `INT4`, `INT8`
- `CHAR`, `TEXT`
- `FLOAT`
- `NUMERIC` (使用S3 Select时，不适用于S3连接器)
- `BOOL`
- `DATE`, `TIMESTAMP` (使用S3 Select时仅适用于JDBC连接器和S3连接器)

您可以通过以下运算符使用PXF过滤器下推功能：

- `<`, `<=`, `>=`, `>`
- `<>`, `=`
- `AND`, `OR`, `NOT`
- 数组`INT`和`TEXT`的运算符`IN` (仅JDBC连接器)
- `LIKE` (`TEXT`字段，仅JDBC连接器)

总而言之，必须满足以下所有条件才能进行谓词下推：

- 将`gp_external_enable_filter_pushdown`服务配置参数设置为`'on'`来启用外部表谓词下推
- 访问外部数据源的MPP数据库协议必须支持谓词下推.PXF外部表协议支持下推
- 访问的外部数据源必须支持下推。 例如，HBase和Hive支持下推
- 使用pxf协议创建的外部表查询，基础PXF连接器还必须支持谓词下推。例

如，PXF Hive，HBase和JDBC连接器支持下推。

- 有关Hive支持此功能的更多信息，请参考Hive分区过滤器下推。

PXF列投影

PXF支持列投影，始终是启用状态。使用列投影时，只从外部数据源返回外部表上的 `SELECT` 查询所需的列。此过程可以提高查询性能，还可以减少传输到MPP数据库的数据量。

Note: 某些外部数据源不支持列投影。 如果查询访问不支持列投影的数据源，则在没有列投影的情况下执行查询，并且在将数据传输到MPP数据库之后对数据进行过滤。

`pxf` 外部表协议自动启用列投影。PXF使用不同的连接器访问外部数据源，列投影支持也由特定的连接器实现决定。 以下PXF连接器和配置文件组合支持读取操作的列投影: - PXF Hive Connector, `HiveORC` profile - PXF JDBC Connector, `Jdbc` profile - PXF Hadoop 和 Object Store Connectors, `hdfs:parquet` , `adl:parquet` , `gs:parquet` , `s3:parquet` , 和 `wasbs:parquet` profiles - 使用Amazon S3 Select服务， `s3:parquet` 和 `s3:text` 配置文件的PXF S3连接器

Note: 如果无法成功序列化查询过滤条件，PXF可能会禁用列投影; 例如，当WHERE子句解析为布尔类型时。

总而言之，必须满足以下所有条件才能进行列投影：

- 外部数据源必须支持列投影。例如，Hive支持ORC格式数据的列投影，某些SQL数据库支持列投影。
- 底层PXF连接器和配置文件实现必须支持列投影。例如，上面标识的PXF Hive和JDBC连接器配置文件支持列投影，支持读取Parquet数据的PXF连接

器也支持列投影。

- PXF必须能够序列化查询过滤条件

配置PXF

GPDB集群包括一个master节点和多个segment节点。初始化和配置PXF时，可以在每个GPDB的segment上启动单个PXF JVM进程。

PXF提供到Hadoop，Hive，HBase，对象存储和外部SQL数据存储的连接器。您必须配置PXF以支持您计划使用的连接器。

配置PXF,你必须:

1. 在每个MPP数据库段主机上安装Java包,详情参见Installing Java for PXF。
2. Initialize the PXF Service。
3. 如果您计划使用Hadoop，Hive或HBase PXF连接器，则必须执行配置PXF Hadoop连接器中所述的配置过程。Configuring PXF Hadoop Connectors。
4. 如果您计划使用PXF连接器访问Azure，Google云端存储，Minio或S3对象存储，则必须执行配置Azure连接，Google云端存储，Minio和S3对象存储中所述的配置过程。详见Configuring Connectors to Azure, Google Cloud Storage, Minio, and S3 Object Stores。
5. 如果计划使用PXF JDBC Connector访问外部SQL数据库，请执行配置JDBC连接器中所述的配置过程。详见Configuring the JDBC Connector。
6. Start PXF

PXF安装和配置目录

In this topic:

- PXF安装目录
- PXF运行目录
- PXF用户配置目录

在你安装MPP数据库时，PXF被安装在master和segment节点上。

PXF安装目录

在安装MPP时，以下配置文件和目录将被安装在你的GPDB实例中。这些文件/目录在PXF的安装目录$GPHOME/pxf:

Directory	Description
apache-tomcat/	PXF的Tomcat目录
bin/	PXF脚本和可执行文件目录
conf/	PXF内部配置目录，这个目录包含 `pxf-env-default.sh` 和 `pxf-profiles-default.xml` 配置文件。在初始化pxf后，这个目录也会包括 `pxf-private.classpath` 文件

Directory	Description
lib/	PXF库目录
templates/	PXF的配置模板

PXF运行目录

在初始化和启动过程中，PXF在$GPHOME/pxf创建了如下内部目录:

Directory	Description
pxf-service/	PXF初始化之后的PXF服务实例目录
run/	启动pxf后，pxf的运行目录。包含PXF catalina进程ID文件

PXF用户配置目录

在pxf初始化过程中，PXF使用以下的子目录和模板文件填充你选择的用户配置目录(`$PXF_CONF`)

Directory	Description
conf/	用户可自定义的PXF配置文件位置： `pxf-env.sh` ， `pxf-log4j.properties` 和 `pxf-profiles.xml`
keytabs/	PXF服务Kerberos秘钥文件的默认位置
lib/	默认的PXF用户运行库目录

Directory	Description
logs/	PXF运行时间日志文件目录，包括 `pxf-service.log` 和 `Tomcat` 相关日志 `catalina.out` . logs目录和日志文件都是 `gpadmin` 用户创建的只读文件
servers/	服务配置目录，每一个子目录标识服务名称。默认服务器名为 `default` 。数据库管理员也可以配置其他服务
templates/	连接器服务器模板文件的配置目录

参数 初始化PXF and 启动PXF 了解PXF初始化和启动命令和过程的详细信息

为PXF安装JAVA环境

In this topic:

- 准备
- 过程

PXF是Java服务 它要求在每个MPP数据库主机上安装Java 8或Java 11。

如果已在每个MPP数据库主机上安装了适当版本的Java，则无需执行本主题中的过程。

准备

确保在每个MPP数据库主机上具有访问Java 8或Java 11的权限，或具有超级用户权限来安装Java 8或Java 11。

过程

执行以下过程，在master,standby和各个segment上安装Java，可以使用 `gpssh` 命令在多个主机上执行。

1. 登录mpp的master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 创建一个text文件列出你的gp集群的standby节点和segment节点，每行一个主机名。例如，一个叫做 `seghostfile` 的文件

```
mstandby
seghost1
seghost2
seghost3
```

3. 在master,standby节点和各个segment节点上安装Java，然后在每个主机上配置环境变量

    1. 安装java包。比如安装java 8:

```
gpadmin@gpmaster$ sudo yum -y install java-1.8.0-openjdk-1.8.0*
gpadmin@gpmaster$ gpssh -e -v -f gphostfile sudo yum -y install
```

    2. 标识Java基本安装目录。如果每台主机上的 `gpadmin` 用户的 `.bashrc` 文件不存在，请更新它以包含 `$JAVA_HOME` 设置。例如，如果您安装了Java 8：

```
gpadmin@gpmaster$ echo 'export JAVA_HOME=/usr/lib/jvm/java-1.8.
gpadmin@gpmaster$ gpssh -e -v -f gphostfile "echo 'export JAVA_
```

如果安装了Java 11, `JAVA_HOME` 可能是 `/usr/lib/jvm/java-11-openjdk-11.0.4.11-0.el7_6.x86_64` 。

注意: 如果超级用户选择了新安装的Java替代品作为系统缺省值，则为 `JAVA_HOME=/usr/lib/jvm/jre` 。

初始化PXF

In this topic:

- PXF配置属性
  - 过程
- 重置PXF
  - 步骤

你必须显式初始化PXF服务实例。 此一次性初始化将创建PXF服务Web应用程序并生成PXF配置文件和模板。

PXF提供了两个你可以在初始化时候使用的管理命令

- `pxf cluster init` - 初始化所有在GP集群中的pxf服务实例
- `pxf init` - 初始化当前节点上的pxf服务实例

PXF还提供了类似的 `reset` 命令，可用于重置PXF配置。

PXF配置属性

PXF同时支持内部的和用户自定义的配置属性。初始化PXF会生成PXF内部配置文件，并设置特定于您的配置的默认属性。初始化PXF还会为用户自定义设置生成配置文件模板，例如自定义配置文件和PXF运行时间和日志记录设置。

PXF内部配置文件位于 `$GPHOME/pxf/conf` 目录，在初始化PXF时，你可以通过指定环境变量 `$PXF_CONF` 来配置用户目录。如果在初始化PXF之前未设置 `$PXF_CONF` ，PXF可能会在初始化过程中提示您接受或拒绝默认用户配置目录 `$HOME/pxf` 。

Note: 选择可以备份的 `$PXF_CONF` 位置，并确保他在GPDB安装目录之外。

在初始化期间，PXF会根据需要创建 `$PXF_CONF` 目录然后使用子目录和模板文件填充他。有关这些目录和内容的列表，可以参照PXF User Configuration Directories

准备

GPDB实例初始化PXF之前，请确保:

- GPDB集群是启动并且运行的
- 可以通过 `$PXF_CONF` 指定PXF用户配置目录的文件位置，并且 `gpadmin` 用户也需要对这个目录有写权限

过程

执行一下流程确保在你的GPDB集群的每个节点初始化PXF。

1. 登录GPDB的master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 运行 `pxf cluster init` 命令会在master,standby节点和所有的segment节点主机上初始化PXF服务。如下命令指定 `/usr/local/mpp-pxf` 作为pxf初始化的用户配置目录

```
gpadmin@gpmaster$ PXF_CONF=/usr/local/mpp-pxf $GPHOME/pxf/bin/pxf
```

`init` 命令创建了pxf网页应用和pxf内部配置。init命令也创建了 `$PXF_CONF` 用户配置目录(如果目录不存在)并使用用户可自定义的配置模板填充 `conf` 和 `templates` 目录。如果 `$PXF_CONF` 存在，则PXF仅更新 `templates` 目录。
Note:PXF服务只运行在segment主机上，然而 `pxf cluster init` 命令也会安装在GPDB的master和standby主机上创建PXF用户配置目录。

重置PXF

如果需要，可以将PXF重置为其未初始化状态。如果您指定了错误的 `PXF_CONF` 目录，或者您想从头开始初始化过程，则可以选择重置PXF。

重置PXF时，PXF会提示您确认操作。如果确认，PXF将删除以下运行时文件和目录（其中 `PXF_HOME=$GPHOME/pxf` ）：

- `$PXF_HOME/conf/pxf-private.classpath`
- `$PXF_HOME/pxf-service`
- `$PXF_HOME/run`

重置操作期间PXF不会删除 `$PXF_CONF` 目录。

必须先在segment主机上停止PXF服务实例，然后才能在主机上重置PXF。

步骤

执行以下过程在MPP数据库集群中的每个segment主机上重置PXF。

1. 登录到MPP数据库master节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 在每个segment主机上停止PXF服务实例。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster stop
```

3. 在所有MPP主机上重置PXF服务实例。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster reset
```

注意: 重置PXF之后，必须初始化并启动PXF才能再次使用该服务。

配置PXF服务器

In this topic:

- 关于服务器模板文件
- 关于默认服务器
- 配置服务器
- 配置PXF用户
  - 过程
- 关于配置属性优先级
- 使用服务器配置

本主题概述了PXF服务器配置。要配置服务器，请参阅特定于您要配置的连接器的主题。

您可以通过PXF连接器读取数据或将数据写入外部数据存储。要访问外部数据存储，您必须提供服务器位置。您可能还需要提供客户端访问凭据和其他特定于外部数据存储的属性。PXF通过以下方式简化了对外部数据存储的配置访问：

- 支持基于文件的连接器和用户配置
- 提供特定于连接器的模板配置文件

PXF * Server *定义只是一个命名配置，提供对特定外部数据存储的访问。PXF服务器名称是位于 `$PXF_CONF/servers/` 中的目录的名称。您在服务器配置中提供的信息是特定于连接器的。例如，PXF JDBC连接器服务器定义可以包括JDBC驱动程序类名称，URL，用户名和密码的设置。您还可以在JDBC服务器定义中配置特定于连接的属性和特定于会话的属性。

PXF为每个连接器提供一个服务器模板文件。此模板标识必须配置才能使用连接器的典型属性集。

您将为MPP数据库用户需要访问的每个外部数据存储配置服务器定义。例如，如果您需要访问两个Hadoop集群，则将为每个集群创建PXF Hadoop服务器配置。如果需要访问Oracle和MySQL数据库，则将为每个数据库创建一个或多个PXF JDBC服务器配置。

服务器配置可以包括用户访问凭据的默认设置以及外部数据存储的其他属性。您可以允许MPP数据库用户使用默认设置访问外部数据存储，也可以基于每个用户配置访问权限和其他属性。这使您可以在单个PXF服务器定义中使用不同的外部数据存储访问凭据来配置不同的MPP数据库用户。 您可以允许MPP数据库用户使用默认设置访问外部数据存储，也可以基于每个用户配置访问权限和其他属性。这使您可以在单个PXF服务器定义中使用不同的外部数据存储访问凭据来配置不同的MPP数据库用户。

关于服务器模板文件

PXF服务器的配置信息位于 `$PXF_CONF/servers/<server_name>/` 中的一个或多个 `<connector> -site.xml` 文件中。

PXF为每个连接器提供一个模板配置文件。初始化PXF后，这些服务器模板配置文件位于 `$PXF_CONF/templates/` 目录中：

```
gpadmin@gpmaster$ ls $PXF_CONF/templates
adl-site.xml   hbase-site.xml  jdbc-site.xml    s3-site.xml
core-site.xml  hdfs-site.xml   mapred-site.xml  wasbs-site.xml
gs-site.xml    hive-site.xml   minio-site.xml   yarn-site.xml
```

例如， `s3-site.xml` 模板文件的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>fs.s3a.access.key</name>
        <value>YOUR_AWS_ACCESS_KEY_ID</value>
    </property>
    <property>
        <name>fs.s3a.secret.key</name>
        <value>YOUR_AWS_SECRET_ACCESS_KEY</value>
    </property>
    <property>
        <name>fs.s3a.fast.upload</name>
        <value>true</value>
    </property>
</configuration>
```

您可以在配置文件中以明文形式为PXF指定凭据。

Note: Hadoop连接器的模板文件不打算被修改并用于配置，因为它们仅提供所需信息的示例。您无需修改Hadoop模板，而是将几个Hadoop `* -site.xml` 文件从Hadoop集群复制到PXF Hadoop服务器配置。

关于默认服务器

PXF定义了一个名为 `default` 的特殊服务器。初始化PXF时，它会自动创建一个 `$PXF_CONF/servers/default/` 目录。此目录最初为空，标识默认的PXF服务器配置。您可以配置默认PXF服务器并将其分配给任何外部数据源。例如，您可以将PXF默认服务器分配给Hadoop集群，或者分配给用户经常访问的MySQL数据库。

如果您在 `CREATE EXTERNAL TABLE` 命令 `LOCATION` 子句中省略了 `SERVER=<server_name>` 设置，则PXF将自动使用 `default` 服务器配置。

Note: 当您的Hadoop集群使用Kerberos身份验证时，您*必须*将Hadoop服务器配

置为PXF `default` 服务器。

配置服务器

将PXF连接器配置为外部数据存储时，将为该连接器添加命名的PXF服务器配置。在执行的任务中，您可以：

1. 确定是要配置 `default` PXF服务器，还是为服务器配置选择新名称。

2. 创建目录 `$PXF_CONF/servers/<server_name>` 。

3. 将模板或其他配置文件复制到新的服务器目录。

4. 为模板文件中的属性填写适当的默认值。

5. 添加环境所需的所有其他配置属性和值。

6. 如关于配置PXF用户中所述为服务器配置配置一个或多个用户。

7. 将服务器和用户配置同步到MPP数据库集群。

Note: 添加或更新PXF服务器配置后，必须将PXF配置重新同步到MPP数据库集群。

配置PXF服务器后，将服务器名称发布给需要访问数据存储的MPP数据库用户。用户只需要在创建访问外部数据存储的外部表时提供服务器名称即可。PXF从驻留在由服务器名称标识的服务器配置目录中的服务器和用户配置文件中获取外部数据源位置和访问凭据。

要配置PXF服务器，请参考连接器配置主题：

- 要为Hadoop配置PXF服务器，请参考配置PXF Hadoop连接器.

- 要为对象存储配置PXF服务器，请参阅为Azure，Google Cloud Storage，Minio和S3对象存储配置连接器.
- 要配置PXF JDBC服务器，请参阅配置JDBC连接器.

配置PXF用户

您可以基于每个服务器，每个MPP用户配置对外部数据存储的访问。

您可以通过在PXF服务器配置目录 `$PXF_CONF/servers/<server_name>/` 中提供 `<mpp_user_name>-user.xml` 用户配置文件来为特定的MPP数据库用户配置外部数据存储用户访问凭据和属性。例如，您在 `$PXF_CONF/servers/<server_name>/bill-user.xml` 文件中为名为 `bill` 的MPP数据库用户指定属性。您可以在PXF服务器配置中配置零个，一个或多个用户。

您在用户配置文件中指定的属性是特定于连接器的。您可以在 `<mpp_user_name>-user.xml` 配置文件中指定PXF连接器服务器支持的任何配置属性。

例如，假设您已经在名为 `pgsrv1` 的PXF JDBC服务器配置中配置了对PostgreSQL数据库的访问。要允许名为 `bill` 的MPP数据库用户以名为 `pguser1` 的PostgreSQL用户(密码为 `changeme`)访问该数据库，请使用以下命令创建用户包含如下属性的配置文件 `$PXF_CONF/servers/pgsrv1/bill-user.xml` ：

```
<configuration>
    <property>
        <name>jdbc.user</name>
        <value>pguser1</value>
    </property>
    <property>
        <name>jdbc.password</name>
        <value>changeme</value>
    </property>
</configuration>
```

如果要为 bill 配置特定的搜索路径和较大的读取大小，则还应将以下属性添加到 bill-user.xml 用户配置文件中：

```
    <property>
        <name>jdbc.session.property.search_path</name>
        <value>bill_schema</value>
    </property>
    <property>
        <name>jdbc.statement.fetchSize</name>
        <value>2000</value>
    </property>
```

过程

对于要配置的每个PXF用户，您将：

1. 标识MPP数据库用户的名称。

2. 标识要为其配置用户访问权限的PXF服务器定义。

3. 标识要为用户配置的每个属性的名称和值。

4. 创建/编辑文件 $PXF_CONF/servers/<server_name>/<mpp_user_name>-user.xml ，并

添加外部配置块：

```
<configuration>
</configuration>
```

5. 将在步骤3中标识的每个属性/值对添加到 `<mpp_user_name>-user.xml` 文件中的配置块中。

6. 如果要将PXF用户配置添加到以前配置的PXF服务器定义中，请将该用户配置同步到MPP数据库集群。

关于配置属性优先级

PXF服务器配置可能包括用于用户访问凭据的默认设置以及用于访问外部数据存储的其他属性。一些PXF连接器(例如S3和JDBC连接器)允许您通过 `CREATE EXTERNAL TABLE` 命令 `LOCATION` 子句中的自定义选项直接指定某些服务器属性。 `<mpp_user_name>-user.xml` 文件为MPP数据库用户指定了外部数据存储的属性设置。

对于给定的MPP数据库用户，PXF使用以下优先级规则(从高到低)获取该用户的配置属性设置：

1. 您在 `<server_name>/<mpp_user_name>-user.xml` 中配置的属性将覆盖其他位置的属性设置。

2. 通过 `CREATE EXTERNAL TABLE` 命令的 `LOCATION` 子句中的自定义选项指定的属性将覆盖PXF服务器配置中该属性的任何设置。

3. 您在PXF服务器定义 `<server_name>` 中配置的属性标识默认属性值。

这些优先级规则使您可以创建一个可由多个MPP数据库用户访问的外部表，每个用户都有各自独特的外部数据存储用户凭据。

使用服务器配置

为了访问外部数据存储，MPP数据库用户在 `CREATE EXTERNAL TABLE` 命令中的 `LOCATION` 子句 `SERVER=<server_name>` 选项中指定服务器名称。用户提供的 `<server_name>` 标识服务器配置目录，PXF从该目录获取配置和凭据以访问外部数据存储。

例如，以下命令使用 `$PXF_CONF/servers/s3srvcfg/s3-site.xml` 文件中定义的服务器配置访问S3对象存储：

```
CREATE EXTERNAL TABLE pxf_ext_tbl(name text, orders int)
  LOCATION ('pxf://BUCKET/dir/file.txt?PROFILE=s3:text&SERVER=s3srvcfg')
FORMAT 'TEXT' (delimiter=E',');
```

如果未提供 `SERVER=<server_name>` 设置，则PXF自动使用 `default` 服务器配置。

例如，如果 `default` 服务器配置标识了Hadoop集群，则以下示例命令将引用位于 `/path/to/file.txt` 的HDFS文件：

```
CREATE EXTERNAL TABLE pxf_ext_hdfs(location text, miles int)
  LOCATION ('pxf://path/to/file.txt?PROFILE=hdfs:text')
FORMAT 'TEXT' (delimiter=E',');
```

查询或写入外部表的MPP数据库用户使用为 `<server_name>` 用户配置的凭据访问外部数据存储。如果没有为 `<server_name>` 配置任何用户专用的凭据，则MPP用户将使用为 `<server_name>` 配置的默认凭据访问外部数据存储。

配置PXF HADOOP连接器

In this topic:

- 准备
- 过程
- 更新hadoop配置

PXF与Cloudera，Hortonworks Data Platform，MapR和通用Apache Hadoop发行版兼容。 本主题描述如何配置PXF Hadoop，Hive和HBase连接器。

如果您不想使用与Hadoop相关的PXF连接器，则无需执行此过程。

准备

配置PXF Hadoop连接器需要将配置文件从Hadoop群集复制到MPP master主机。如果你使用的是MapR Hadoop发行版，则还必须将某些JAR文件复制到master主机。在配置PXF Hadoop连接器之前，请确保可以将文件从Hadoop群集中的主机复制到MPP数据库master服务器。

过程

在GPDB master主机上执行以下操作去配置PXF映射hadoop的连接器。在你配置连接器完成后，需要运行 `pxf cluster sync` 命令拷贝PXF的配置到MPP数据库集群。 在此过程中，您将使用 `default` ，或创建新的PXF服务器配置。您将Hadoop配置文件复制到MPP数据库master主机上的服务器配置目录。您也可以将库复制到 `$PXF_CONF/lib` 以获取MapR支持。然后，您可以将master主机上的PXF配置同步到standby和segment主机。(当您运行 `pxf cluster init` 时，PXF将创建 `$PXF_CONF/*` 目录。)

1. 登录到GPDB master节点:

```
$ ssh gpadmin@<gpmaster>
```

2. 确定您的PXF Hadoop服务器配置的名称。如果您的Hadoop集群是Kerberized，则必须使用 `default` PXF服务器。

3. 如果您没有使用默认的PXF服务器，请创建 `$PXF_HOME/servers/<server_name>` 目录。例如，使用以下命令创建名为 `hdp3` 的Hadoop服务器配置：

```
gpadmin@gpmaster$ mkdir $PXF_CONF/servers/hdp3
```

4. 转到服务器目录。例如：

```
gpadmin@gpmaster$ cd $PXF_CONF/servers/default
```

或，

```
gpadmin@gpmaster$ cd $PXF_CONF/servers/hdp3
```

5. PXF服务需要从 `core-site.xml` 和其他Hadoop配置文件中获取需要的信息。使用你选择的工具从你的Hadoop集群namenode节点主机上拷贝 `core-site.xml` 、 `hdfs-site.xml` 、 `mapred-site.xml` 和 `yarn-site.xml` 文件到当前主机上(您的文件路径可能会根据使用的Hadoop发行版而有所不同)。例如，这些命令使用 `scp` 去拷贝文件：

```
gpadmin@gpmaster$ cd $PXF_CONF/servers/default
gpadmin@gpmaster$ scp hdfsuser@namenode:/etc/hadoop/conf/core-site
gpadmin@gpmaster$ scp hdfsuser@namenode:/etc/hadoop/conf/hdfs-site
gpadmin@gpmaster$ scp hdfsuser@namenode:/etc/hadoop/conf/mapred-si
gpadmin@gpmaster$ scp hdfsuser@namenode:/etc/hadoop/conf/yarn-site
```

6. 如果你想要使用PXF的HIVE连接器访问hive表的数据，同样拷贝hive的配置

到GPDB master上。例如：

```
gpadmin@gpmaster$ scp hiveuser@hivehost:/etc/hive/conf/hive-site.x
```

7. 如果你想要使用PXF的HBASE连接器访问hbase表数据，同样需要拷贝hbase的配置到GPDB master上。例如:

```
gpadmin@gpmaster$ scp hbaseuser@hbasehost:/etc/hbase/conf/hbase-si
```

8. 如果你要使用PXF访问MapR Hadoop发行版，你必须要从你的MapR拷贝匹配的JAR文件到GPDB master上(您的文件路径可能会根据使用的MapR版本而有所不同)。例如：这些是使用 `scp` 命令拷贝文件

```
gpadmin@gpmaster$ cd $PXF_CONF/lib
gpadmin@gpmaster$ scp mapruser@maprhost:/opt/mapr/hadoop/hadoop-2.
gpadmin@gpmaster$ scp mapruser@maprhost:/opt/mapr/hadoop/hadoop-2.
gpadmin@gpmaster$ scp mapruser@maprhost:/opt/mapr/hadoop/hadoop-2.
```

9. 同步PXF的配置到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

0. MPP数据库最终用户访问Hadoop服务。默认情况下，PXF服务尝试使用GPDB的用户去验证访问HDFS, Hive, and HBase。为了支持此功能，如果要使用这些PXF连接器，则必须为Hadoop以及Hive和HBase配置代理设置。参照Configuring User Impersonation and Proxying 中的过程为Hadoop服务配置用户模拟和代理，或关闭PXF用户模拟。

1. 授予HDFS文件和目录的读取权限，这些文件和目录将作为MPP数据库中的外部表进行访问。 如果启用了用户模拟(默认设置)，则必须向每个MPP数据库用户/角色名称授予此权限，这些用户/角色名称将使用引用HDFS文件的外部表。如果未启用用户模拟，则必须将此权限授予gpadmin用户。

2. 如果您的Hadoop集群使用Kerberos保护，则为安全HDFS配置PXF必须为每个segment主机生成Kerberos主体和密钥表。

更新hadoop配置

在PXF服务运行时，如果你想要更新Hadoop、Hive或者HBase的配置，你必须在你的GPDB集群上重新同步PXF的配置并且在每个segment节点上重启pxf服务。例如：

```
gpadmin@gpmaster$ cd $PXF_CONF/servers/<server_name>
gpadmin@gpmaster$ scp hiveuser@hivehost:/etc/hive/conf/hive-site.xml
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

配置用户模拟和代理

In this topic:

- 配置PXF用户模拟
- 配置Hadoop代理
- Hive用户模拟
- Hbase用户模拟

PXF使用MPP数据库最终用户访问Hadoop服务。默认情况下，PXF尝试使用经过GPDB身份验证的用户身份并且通过PXF连接器配置去访问数据源服务(HDFS, Hive, HBase)。注意，PXF在访问Hadoop服务时仅使用用户的登录身份。例如，如果用户以jane身份登录MPP数据库，然后执行SET ROLE或SET

SESSION AUTHORIZATION来假定其他用户身份，则所有PXF请求仍将使用jane身份来访问Hadoop服务。

使用默认的PXF配置，您必须显式配置每个Hadoop数据源(HDFS，Hive，HBase)来允许PXF进程所有者(通常为 `gpadmin`)充当模拟用户或组的代理。详情参阅Configuring Hadoop Proxying, Hive User Impersonation, and HBase User Impersonation。

或者，您可以禁用PXF用户模拟功能。禁用用户模拟后，PXF将使用PXF进程的所有者(通常为 `gpadmin`)来执行所有Hadoop服务请求。此行为与PXF的早期版本相匹配，但是它没有提供任何方法来控制Hadoop中不同于MPP数据库用户对Hadoop服务的访问。它要求 `gpadmin` 用户有权访问HDFS中的所有文件和目录，以及在PXF外部表定义中引用的Hive和HBase中的所有表的权限。参照Configuring PXF User Impersonation for information about disabling user impersonation。

配置PXF用户模拟

执行以下流程在你的GPDB集群上打开或者关闭用户模拟功能。如果你第一次配置PXF，默认用户模拟是打开的。你不需要执行这个过程。

1. 以管理员用户身份登录GPDB master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 重新进入PXF用户配置目录 `$PGF_CONF` 。用文本编辑器打开配置文件 `$PXF_CONF/conf/pxf-env.sh` 。例如：

```
gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
```

3. 在 `pxf-env.sh` 文件中找到 `PXF_USER_IMPERSONATION` 设置。 将该值设置为

true可打开PXF用户模拟，或将其设置为false可将其关闭。 例如：

```
PXF_USER_IMPERSONATION="true"
```

4. 使用 `pxf cluster sync` 命令拷贝更新的文件 `pxf-env.sh` 文件到MPP数据库集群。例如

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

5. 如果先前已启动PXF，请按照Restarting PXF中的说明在每个MPP数据库segment主机上重新启动它来使新设置生效。

配置Hadoop代理

启用PXF用户角色(默认设置)后，您必须配置Hadoop `core-site.xml` 配置文件以打开用户模拟PXF。 步骤如下：

1. 在你的Hadoop集群中，使用文本编辑器打开 `core-site.xml` 配置文件，或者使用Ambari添加或编辑此过程中描述的Hadoop属性值。

2. 设置属性值 `hadoop.proxyuser.<name>.hosts` 指定允许代理请求的PXF主机名列表。将PXF代理用户(通常为 `gpadmin`)替换为，并在用逗号分隔的列表中提供多个PXF主机名。 例如：

```
<property>
    <name>hadoop.proxyuser.gpadmin.hosts</name>
    <value>pxfhost1,pxfhost2,pxfhost3</value>
</property>
```

3. 设置属性值 `hadoop.proxyuser.<name>.groups` 指定能被PXF模拟的HDFS组列表。您应该将此列表限制为仅需要从PXF访问HDFS数据的那些组。 例如：

```
<property>
    <name>hadoop.proxyuser.gpadmin.groups</name>
    <value>group1,group2</value>
</property>
```

4. 在修改过 core-site.xml 配置文件之后，你必须重启Hadoop使你的配置生效。

5. 将更新后的 core-site.xml 文件复制到master服务器上的PXF Hadoop服务器配置目录 $PXF_CONF/servers/default 中，并将配置同步到standby服务器和每个MPP数据库segment主机。

Hive用户模拟

PXF Hive连接器使用Hive MetaStore确定Hive表的HDFS位置，然后直接访问基础HDFS文件。Hive不需要特定的模拟配置，因为 core-site.xml 中的Hadoop代理配置也适用于以这种方式访问的Hive表。

Hbase用户模拟

为了使用户模拟功能能够与HBase一起使用，必须在HBase配置中启用 AccessController 并重新启动群集。有关 hbase-site.xml 配置设置，请参阅《Apache HBase参考手册》中的61.3 Server-side Configuration for Simple User Access Operation以进行简单的用户访问操作。

为安全**HDFS**配置**PXF**

In this topic:

- 准备
- 过程

为您的HDFS文件系统启用Kerberos时，作为HDFS客户端的PXF需要principal文件和keytab文件来验证对HDFS的访问。要在安全的HDFS上读取或写入文件，必须创建和部署PXF的Kerberos主体和keytab文件，并确保Kerberos身份验证已启用并起作用。

准备

在你配置PXF去访问需要安全访问的HDFS文件系统之前，确保你已经做到：

- 使用默认的PXF服务器配置配置了Hadoop连接器。
- 按照Configuring PXF章节秒速初始化、配置和启动PXF，包括PXF功能和Hadoop用户模拟功能启用
- 根据特定分发的说明为Hadoop集群启用Kerberos，并验证配置。
- 确保HDFS的配置参数 `dfs.block.access.token.enable` 已经被设置成了 `true` 。你可以在你hadoop集群的 `hdfs-site.xml` 配置文件中找到该配置。
- 注意每个MPP数据库segment主机(<seghost>)和Kerberos密钥分发中心(KDC) <kdc-server>主机的主机名或IP地址
- 注意群集所在的Kerberos <realm> 的名称。

过程

按照如下流程为有安全的HDFS配置PXF。您将在Kerberos KDC服务器和MPP数据库segment主机上执行操作。

登录到Kerberos KDC服务主机执行如下操作:

1. 使用 root 账户登录到 Kerberos KDC 服务主机。

```
$ ssh root@<kdc-server>
root@kdc-server$
```

2. 如果你的集群主机不存在配置文件的话，需要在KDC服务主机上分发 /etc/krb5.conf 配置文件到GPDB的每一个的segment节点，例如:

```
root@kdc-server$ scp /etc/krb5.conf seghost:/etc/krb5.conf
```

3. 使用 kadmin.local 命令为GPDB的每个segment节点主机创建一个Kerberos PXF服务主体。服务主体的格式应为 gpadmin/<seghost>@<realm> ，其中<seghost>是segment主机系统的DNS可解析的全限定主机名(hostname -f 命令的输出)。例如，以下命令在名为EXAMPLE.COM的Kerberos领域中为名为 host1.example.com，host2.example.com 和 host3.example.com 的主机创建PXF服务主体:

```
root@kdc-server$ kadmin.local -q "addprinc -randkey -pw changeme g
root@kdc-server$ kadmin.local -q "addprinc -randkey -pw changeme g
root@kdc-server$ kadmin.local -q "addprinc -randkey -pw changeme g
```

4. 为每个Kerberos PXF服务主体生成一个密钥表文件。 将密钥表文件保存在任何方便的位置(本示例使用目录 /etc/security/keytabs)。 您将在以后的步骤中将keytab文件部署到它们各自的MPP数据库segment主机。 例如：

```
root@kdc-server$ kadmin.local -q "xst -norandkey -k /etc/security/
root@kdc-server$ kadmin.local -q "xst -norandkey -k /etc/security/
root@kdc-server$ kadmin.local -q "xst -norandkey -k /etc/security/
```

根据需要重复 xst 命令用以在每个PXF服务主体上生成一个秘钥表。

5. 列出principals:

```
root@kdc-server$ kadmin.local -q "listprincs"
```

6. 将每个PXF服务主体的密钥表文件复制到其各自的segment主机。 例如，以下命令在 PXF_CONF=/usr/local/mpp-pxf 时，将在步骤4中生成的每个主体复制到segment主机上的PXF默认keytab目录中

```
root@kdc-server$ scp /etc/security/keytabs/pxf-host1.service.keyta
root@kdc-server$ scp /etc/security/keytabs/pxf-host2.service.keyta
root@kdc-server$ scp /etc/security/keytabs/pxf-host3.service.keyta
```

   Note the file system location of the keytab file on each PXF host; you will need this information for a later configuration step.

7. 更改 pxf.service.keytab 文件的属主和权限。 这些文件必须仅由 gpadmin 用户拥有并可读。 例如:

```
root@kdc-server$ ssh host1.example.com chown gpadmin:gpadmin /usr/
root@kdc-server$ ssh host1.example.com chmod 400 /usr/local/mpp-px
root@kdc-server$ ssh host2.example.com chown gpadmin:gpadmin /usr/
root@kdc-server$ ssh host2.example.com chmod 400 /usr/local/mpp-px
root@kdc-server$ ssh host3.example.com chown gpadmin:gpadmin /usr/
root@kdc-server$ ssh host3.example.com chmod 400 /usr/local/mpp-px
```

在每个GPDB的segment节点执行以下步骤:

1. 登录segment节点，例如:

```
$ ssh gpadmin@<seghost>
```

2. 如果在MPP的各个segment节点上Kerberos客户端包还没有被安装，那么首先安装他们。您必须具有超级用户权限才能安装操作系统软件包。 例如:

```
root@seghost$ rpm -qa | grep krb
root@seghost$ yum install krb5-libs krb5-workstation
```

3. 打开 `pxf-env.sh` 用户配置文件。例如使用vi命令打开 `PXF_CONF=/usr/local/mpp-pxf`:

```
gpadmin@seghost$ vi /usr/local/mpp-pxf/conf/pxf-env.sh
```

4. 更新 `PXF_KEYTAB` 和 `PXF_PRINCIPAL` 配置. 指定keytab 文件 和 Kerberos principal的位置。这些设置的默认值如下所示:

```
export PXF_KEYTAB="${PXF_CONF}/keytabs/pxf.service.keytab"
export PXF_PRINCIPAL="gpadmin/_HOST@EXAMPLE.COM"
```

   PXF automatically replaces `_HOST` with the FQDN of the segment host.

5. 在segment上重启PXF服务:

```
gpadmin@seghost$ $GPHOME/pxf/bin/pxf restart
```

配置Minio和S3对象存储的连接器（可选）

In this topic:

- 关于对象存储配置
- Minio服务器配置
- S3服务器配置
  - 配置S3服务器端加密
- 示例服务器配置过程

您可以使用PXF访问S3兼容的对象存储。 本主题描述如何将PXF连接器配置为这些外部数据源。

如果您不打算使用这些PXF对象存储连接器，则不需要执行此过程。

关于对象存储配置

要访问对象存储中的数据，必须提供服务器位置和客户端凭据。 配置PXF对象库连接器时，请为连接器添加至少一个名为PXF服务器的配置，如配置PXF服务器中所述。

PXF为每个对象存储连接器提供了一个模板配置文件。 这些模板文件位于`$PXF_CONF/templates/`目录中。

Minio服务器配置

Minio的模板配置文件为`$PXF_CONF/templates/minio-site.xml`。 配置Minio服务器时，必须提供以下服务器配置属性，并用凭据替换模板值：

属性	描述	值
fs.s3a.endpoint	要连接的Minio S3端点。	您的端点。

属性	描述	值
fs.s3a.access.key	Minio帐户访问密钥ID。	您的访问密钥。
fs.s3a.secret.key	与Minio访问密钥ID相关联的密钥。	您的密钥。

S3服务器配置

S3的模板配置文件为 `$PXF_CONF/templates/s3-site.xml` 。 配置S3服务器时，必须提供以下服务器配置属性，并用凭据替换模板值：

属性	描述	值
fs.s3a.access.key	AWS账户访问密钥ID。	您的访问密钥。
fs.s3a.secret.key	与AWS访问密钥ID关联的密钥。	您的密钥。

如果需要，可以通过指定 `s3-site.xml` 服务器配置文件中的Hadoop-AWS模块文档的S3A部分来调优PXF S3连接。

您可以通过CREATE EXTERNAL TABLE命令LOCATION子句中的自定义选项直接指定S3访问ID和密钥，从而覆盖S3服务器配置的凭据。 有关其他信息，请参考使用DDL覆盖S3服务器配置。

配置S3服务器端加密

PXF支持使用可读写的MPP数据库外部表（指定 `pxf` 协议和 `s3:*` 配置文件）访问的S3文件的Amazon Web Service S3服务器端加密（SSE）。 AWS S3服务器端加密可安静地保护您的数据； 它会在将对象数据写入磁盘时对其进行加密，并在您访问数据时为您透明地解密数据。

PXF支持以下AWS SSE加密密钥管理方案：

- 具有S3受管密钥的SSE(SSE-S3) - Amazon管理数据和master加密密钥。
- 具有密钥管理服务托管密钥的SSE(SSE-KMS) - Amazon管理数据密钥，而您在AWS KMS中管理加密密钥。
- SSE与客户提供的密钥(SSE-C) - 您设置和管理加密密钥。

无论数据是否经过加密，您的S3访问密钥和加密密钥都将控制您对所有S3存储桶对象的访问。

在您通过指定 `pxf` 协议和 `s3:*` 配置文件创建的可读外部表访问的加密文件的读取操作期间，S3透明地解密数据。 无需其他配置。

要加密通过这种类型的外部表写入S3的数据，您有两个选择：

- 通过AWS控制台或命令行工具（建议），在每个S3存储桶的基础上配置默认的SSE加密密钥管理方案。
- 在PXF S3服务器的 `s3-site.xml` 配置文件中配置SSE加密选项。

通过S3存储桶策略配置SSE（推荐）

您可以创建S3桶策略</ i>，以标识要加密的对象，加密密钥管理方案以及在这些对象上允许的写入操作。 请参阅AWS S3文档中的使用服务器端加密保护数据，以获取有关SSE加密密钥管理方案的更多信息。 如何为S3存储桶启用默认加密？介绍了如何设置默认加密存储桶策略 。

在PXF S3服务器配置中指定SSE选项

您必须在 `s3-site.xml` 中包含某些属性，才能在PXF S3服务器配置中配置服务器端加密。 您添加到文件中的属性和值取决于SSE加密密钥管理方案。

SSE-S3

要在写入任何S3存储桶的任何文件上启用SSE-S3，请在 `s3-site.xml` 文件中设置以下加密算法属性和值：

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>AES256</value>
</property>
```

要为特定的S3存储桶启用SSE-S3，请使用包含存储桶名称的属性名称变体。例如：

```
<property>
  <name>fs.s3a.bucket.YOUR_BUCKET1_NAME.server-side-encryption-algori
  <value>AES256</value>
</property>
```

将 `YOUR_BUCKET1_NAME` 替换为S3存储桶的名称。

SSE-KMS

要在您写入任何S3存储桶的任何文件上启用SSE-KMS，请同时设置加密算法和加密密钥ID。 要在 `s3-site.xml` 文件中设置这些属性：

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>SSE-KMS</value>
</property>
<property>
  <name>fs.s3a.server-side-encryption.key</name>
  <value>YOUR_AWS_SSE_KMS_KEY_ARN</value>
</property>
```

用您的密钥资源名称替换 `YOUR_AWS_SSE_KMS_KEY_ARN` 。 如果您未指定加密密钥，则使用Amazon KMS中定义的默认密钥。 KMS密钥示例：

`arn:aws:kms:us-west-2:123456789012:key/1a23b456-7890-12cc-d345-6ef7890g12f3` 。

注意: 确保在同一Amazon可用区中创建存储桶和密钥。

要为特定的S3存储桶启用SSE-KMS，请使用包含存储桶名称的属性名称变体。例如：

```
<property>
  <name>fs.s3a.bucket.YOUR_BUCKET2_NAME.server-side-encryption-algor
  <value>SSE-KMS</value>
</property>
<property>
  <name>fs.s3a.bucket.YOUR_BUCKET2_NAME.server-side-encryption.key</
  <value>YOUR_AWS_SSE_KMS_KEY_ARN</value>
</property>
```

将 `YOUR_BUCKET2_NAME` *替换为S3存储桶的名称。*

SSE-C

要在写入任何S3存储桶的任何文件上启用SSE-C，请同时设置加密算法和加密密钥（base-64编码）。 所有客户端必须共享相同的密钥。

要在 `s3-site.xml` *文件中设置这些属性：*

```
<property>
  <name>fs.s3a.server-side-encryption-algorithm</name>
  <value>SSE-C</value>
</property>
<property>
  <name>fs.s3a.server-side-encryption.key</name>
  <value>YOUR_BASE64-ENCODED_ENCRYPTION_KEY</value>
</property>
```

要为特定的S3存储桶启用SSE-C，请使用包含存储桶名称的属性名称变体，如

SSE-KMS示例中所述。

示例服务器配置过程

在配置对象存储连接器服务器之前，请确保已初始化PXF。

在此过程中，您将在S3云存储连接器的MPP数据库master主机上的 `$PXF_CONF/servers` 目录中命名并添加PXF服务器配置。 然后，您可以使用 `pxf cluster sync` 命令将服务器配置同步到MPP数据库集群。

1. 登录到您的MPP数据库主节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 选择服务器的名称。 您将为需要引用对象存储中文件的最终用户提供名称。

3. 创建 `$PXF_HOME/servers/<server_name>` 目录。 例如，使用以下命令为名为 `s3_user1cfg` 的S3服务器创建服务器配置：

```
gpadmin@gpmaster$ mkdir $PXF_CONF/servers/s3_user1cfg
```

4. 将S3的PXF模板文件复制到服务器配置目录。 例如：

```
gpadmin@gpmaster$ cp $PXF_CONF/templates/s3-site.xml $PXF_CONF/ser
```

5. 在您选择的编辑器中打开模板服务器配置文件，并为您的环境提供适当的属性值。 例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>fs.s3a.access.key</name>
        <value>access_key_for_user1</value>
    </property>
    <property>
        <name>fs.s3a.secret.key</name>
        <value>secret_key_for_user1</value>
    </property>
    <property>
        <name>fs.s3a.fast.upload</name>
        <value>true</value>
    </property>
</configuration>
```

6. 保存更改并退出编辑器。

7. 使用 `pxf cluster sync` 命令将新的服务器配置复制到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

配置Azure和Google Cloud Storage对象存储的连接器（可选）

In this topic:

- 对象存储配置
  - Azure Blob存储服务配置
  - Azure数据湖服务配置
  - Google Cloud Storage服务配置
- 示例服务器配置过程

您可以使用PXF访问Azure Data Lake，Azure Blob存储和Google Cloud Storage对象存储。这个章节主要描述如何配置PXF连接器访问这个外部数据源。

如果您不打算使用PXF对象存储连接器，则不需要执行此过程。

对象存储配置

要访问对象存储中的数据，必须提供服务器位置和客户端凭据。配置PXF对象库连接器时，请为连接器添加至少一个名为PXF服务器的配置，如配置PXF服务器中所述。

PXF为每个对象存储连接器提供了一个模板配置文件。 这些模板文件位于 `$PXF_CONF/templates/` 目录中。

Azure Blob存储服务配置

Azure Blob存储服务的默认配置文件是 `$PXF_CONF/templates/wasbs-site.xml` 。如果你想要配置Azure Blob存储服务，你需要提供如下的配置项并且替换模板中的值。

Property	Description	Value
fs.adl.oauth2.access.token.provider.type	令牌类型	必须指定 `ClientCredential`.
fs.azure.account.key.<YOUR_AZURE_BLOB_STORAGE_ACCOUNT_NAME>.blob.core.windows.net	Azure帐户密钥	用你的账户秘钥替换
fs.AbstractFileSystem.wasbs.impl	文件系统类名称	必须指定 `org.apache.hadoop.fs.azure.Wasbs`

Azure数据湖服务配置

Azure数据湖服务默认模板文件是 `$PXF_CONF/templates/adl-site.xml` ，当你配置Azure数据湖服务时需要配置如下的配置项并替换模板中的值:

Property	Description	Value
fs.adl.oauth2.access.token.provider.type	令牌类型	必须指定 `ClientCredential`.
fs.adl.oauth2.refresh.url	要连接的Azure结束位置	Your refresh URL.
fs.adl.oauth2.client.id	Azure账户的客户端ID	Your client ID (UUID).
fs.adl.oauth2.credential	Azure账户的客户端ID的密码	Your password.

Google Cloud Storage服务配置

Google Cloud Storage服务默认配置模板文件是 `$PXF_CONF/templates/gs-site.xml` ，当你配置Google Cloud Storage服务的时候需要修改如下的配置项和值：

Property	Description	Value
google.cloud.auth.service.account.enable	启用服务帐户授权	Must specify `true` .
google.cloud.auth.service.account.json.keyfile	Google Storage的秘钥文件	Path to your key file.
fs.AbstractFileSystem.gs.impl	文件系统类名称	Must specify `com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS` .

示例服务器配置过程

在配置对象存储连接器服务器之前，请确保已初始化PXF。

在此过程中，您需要在MPP数据库master主机上的 `$PXF_CONF/servers` 目录中为Google Cloud Storate（GCS）连接器命名并添加PXF服务器配置。 然后，您可以使用 `pxf cluster sync` 命令将服务器配置同步到MPP数据库集群。

1. 登录到GPDB master主机

```
$ ssh gpadmin@<gpmaster>
```

2. 选择服务器的名称。 您将为需要引用对象存储中文件的最终用户提供名称。

3. 创建 `$PXF_HOME/servers/<server_name>` 目录。 例如，使用以下命令为名为

`gs_public`的Google Cloud Storage服务器创建服务器配置：

```
gpadmin@gpmaster$ mkdir $PXF_CONF/servers/gs_public
```

4. 将GCS的PXF模板文件复制到服务器配置目录。 例如：

```
gpadmin@gpmaster$ cp $PXF_CONF/templates/gs-site.xml $PXF_CONF/ser
```

5. 在您选择的编辑器中打开模板服务器配置文件，并为您的环境提供适当的属性值。 例如，如果您的Google Cloud Storage密钥文件位于`/home/gpadmin/keys/gcs-account.key.json`中：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>google.cloud.auth.service.account.enable</name>
        <value>true</value>
    </property>
    <property>
        <name>google.cloud.auth.service.account.json.keyfile</name
        <value>/home/gpadmin/keys/gcs-account.key.json</value>
    </property>
    <property>
        <name>fs.AbstractFileSystem.gs.impl</name>
        <value>com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS</valu
    </property>
</configuration>
```

6. 保存更改并退出编辑器。

7. 使用`pxf cluster sync`命令将新的服务器配置复制到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

配置JDBC连接器（可选）

In this topic:

- 关于JDBC配置
- JDBC驱动程序JAR注册
- JDBC服务器配置
  - 连接级属性
  - 连接事务隔离属性
  - 语句级属性
  - 会话级属性
  - 关于JDBC用户模拟
  - 关于JDBC连接池
- JDBC命名查询配置
  - 定义命名查询
  - 查询命名
- 覆盖JDBC服务器配置
- 示例配置过程
- 配置Hive访问
  - JDBC服务器配置
  - 示例配置步骤

您可以使用PXF访问外部SQL数据库，包括MySQL，ORACLE，PostgreSQL，Hive和Apache Ignite。本主题描述如何配置PXF JDBC连接器以访问这些外部数据源。

如果您不打算使用PXF JDBC连接器，则不需要执行此过程。

关于JDBC配置

要使用PXF JDBC连接器访问外部SQL数据库中的数据，您必须：

- 注册兼容的JDBC驱动程序JAR文件
- 指定JDBC驱动程序类名称，数据库URL和客户端凭据

在以前的MPP数据库版本中，您可能已通过 `CREATE EXTERNAL TABLE` 命令中的选项指定了JDBC驱动程序类名称，数据库URL和客户端凭据。PXF现在支持JDBC连接器的基于文件的服务器配置。如下所述，此配置使您可以在文件中指定这些选项和凭据。

注意：您先前创建的直接指定JDBC连接选项的PXF外部表将继续起作用。如果要移动这些表以使用基于JDBC文件的服务器配置，则必须创建服务器配置，删除外部表，然后重新创建表，并指定适当的 `SERVER=<server_name>` 子句。

JDBC驱动程序JAR注册

PXF JDBC连接器随 `postgresql-42.2.5.jar` JAR文件一起安装。如果您需要其他JDBC驱动程序，请确保在每个segment主机的 `$PXF_CONF/lib` 目录中为外部SQL数据库安装JDBC驱动程序JAR文件。确保安装与您的JRE版本兼容的JDBC驱动程序JAR文件。有关更多信息，请参见注册PXF JAR依赖项。

JDBC服务器配置

配置PXF JDBC连接器时，请为连接器添加至少一个名为PXF服务器的配置，如配置PXF服务器中所述。您还可以配置一个或多个静态定义的查询以对远程SQL数据库运行。

PXF提供了JDBC连接器的模板配置文件。该服务器模板配置文件位于

`$PXF_CONF/templates/jdbc-site.xml` 中，标识可以配置以建立与外部SQL数据库的连接的属性。该模板还包括可选属性，您可以在执行查询或在外部数据库会话中插入命令之前设置这些可选属性。

服务器模板文件 `jdbc-site.xml` 中的必需属性如下：

属性	描述	值
jdbc.driver	JDBC驱动类名称	JDBC驱动程序Java类名；例如 `org.postgresql.Driver` 。
jdbc.url	JDBC驱动程序用于连接数据库的URL	数据库连接URL(特定于数据库)，例如： `jdbc:postgresql://phost:pport/pdatabase` 。
jdbc.user	数据库用户名	连接数据库的用户名。
jdbc.password	`jdbc.user` 的密码	连接数据库的密码。

配置PXF JDBC服务器时，可以在配置文件中以明文形式为PXF指定外部数据库用户凭据。

连接级属性

要设置其他JDBC连接级别的属性，请将 `jdbc.connection.property.<CPROP_NAME>` 属性添加到 `jdbc-site.xml` 中。当PXF建立到外部SQL数据库（ `DriverManager.getConnection()` ）的连接时，会将这些属性传递给JDBC驱动程序。

将 `<CPROP_NAME>` 替换为连接属性名称，并指定其值：

属性	描述	值
jdbc.connection.property.<CPROP_NAME>	当PXF建立与外部SQL数据库的连接时，传递给JDBC驱动程序的属性名称(<CPROP_NAME>)。	<CPROP_NAME>属性的值。

示例：要在与PostgreSQL数据库的JDBC连接上设置 `createDatabaseIfNotExist` 连接属性，请在 `jdbc-site.xml` 中包含以下属性块：

```
<property>
    <name>jdbc.connection.property.createDatabaseIfNotExist</name>
    <value>true</value>
 </property>
```

确保外部SQL数据库的JDBC驱动程序支持您指定的任何连接级属性。

连接事务隔离属性

SQL标准定义了四个事务隔离级别。您为与外部SQL数据库的给定连接指定的级别决定了对另一连接如何和何时可见在该连接上执行的一个事务所做的更改。

PXF JDBC连接器公开了一个名为 `jdbc.connection.transactionIsolation` 的可选服务器配置属性，该属性使您可以指定事务隔离级别。建立与外部SQL数据库的连接后，PXF会设置级别（ `setTransactionIsolation()` ）。

JDBC连接器支持以下 `jdbc.connection.transactionIsolation` 属性值：

SQL 级别	PXF属性值
未提交读	READ_UNCOMMITTED
已提交读	READ_COMMITTED
可重复读	REPEATABLE_READ
可串行化	SERIALIZABLE

例如，要将事务隔离级别设置为未提交读，请将以下属性块添加到 `jdbc-site.xml` 文件中：

```
<property>
    <name>jdbc.connection.transactionIsolation</name>
    <value>READ_UNCOMMITTED</value>
</property>
```

不同的SQL数据库支持不同的事务隔离级别。确保外部数据库支持您指定的级别。

语句级属性

PXF JDBC连接器通过语句在外部SQL数据库表上执行查询或插入命令。连接器公开的属性使您可以在外部数据库中执行命令之前配置语句的某些方面。连接器支持以下语句级属性：

属性	描述	值
jdbc.statement.batchSize	批量写入外部数据库表的行数。	行数。默认的写入批处理大小为100。
jdbc.statement.fetchSize	从外部数据库表读取时要提取/缓冲的行数。	行数。默认读取大小为1000。
jdbc.statement.queryTimeout	JDBC驱动程序等待语句执行的时间（以秒为单位）。此超时适用于为读取和写入操作创建的语句。	超时时间（以秒为单位）。默认等待时间是无限的。

对于您未明确配置的任何语句级属性，PXF使用默认值。

示例：要将读取获取大小设置为5000，请将以下属性块添加到 `jdbc-site.xml` 中：

```
<property>
    <name>jdbc.statement.fetchSize</name>
    <value>5000</value>
</property>
```

确保外部SQL数据库的JDBC驱动程序支持您指定的任何语句级属性。

会话级属性

要设置会话级别的属性，请在 `jdbc-site.xml` 中添加 `jdbc.session.property。<SPROP_NAME>` 属性。在执行查询之前，PXF将在外部数据库中 `SET` 这些属性。

将 `<SPROP_NAME>` 替换为会话属性名称，并指定其值：

属性	描述	值
jdbc.session.property.<SPROP_NAME>	在执行查询之前要设置的会话属性的名称(<SPROP_NAME>)。	<SPROP_NAME>属性的值。

注意: PXF JDBC连接器完全按照 `jdbc-site.xml` 服务器配置文件中的指定，将会话属性名和属性值都传递给外部SQL数据库。为了限制SQL注入的潜在威胁，连接器拒绝包含 `;`，`\n`，`\b`，或 `\0` 字符的任何属性名称或值。

PXF JDBC连接器为所有支持的外部SQL数据库处理会话属性 `SET` 语法。

示例：要在PostgreSQL数据库中运行查询之前设置 `search_path` 参数，请将以下属性块添加到 `jdbc-site.xml` 中：

```
<property>
    <name>jdbc.session.property.search_path</name>
    <value>public</value>
</property>
```

确保外部SQL数据库的JDBC驱动程序支持您指定的任何属性。

关于JDBC用户模拟

PXF JDBC连接器使用 `jdbc.user` 设置或 `jdbc.url` 中的信息来确定连接到外部数据存储的用户身份。禁用PXF JDBC用户模拟时（默认设置），JDBC连接器的行为进一步取决于外部数据存储。例如，如果您使用JDBC连接器访问Hive，则连接器将使用某些Hive身份验证和模拟属性的设置来确定用户。您可能需要提供 `jdbc.user` 设置，或在服务器 `jdbc-site.xml` 文件中的 `jdbc.url` 设置中添加属性。

启用PXF JDBC用户模拟时，PXF JDBC连接器代表MPP数据库最终用户访问外部数据存储。连接器使用访问PXF外部表的MPP数据库用户的名称来尝试连接到外部数据存储。

`pxf.impersonation.jdbc` 属性控制JDBC用户模拟。默认情况下，禁用JDBC用户模拟。要为服务器配置启用JDBC用户模拟，请将属性设置为true：

```
<property>
    <name>pxf.impersonation.jdbc</name>
    <value>true</value>
</property>
```

为PXF服务器启用JDBC用户模拟时，PXF会覆盖在 `jdbc-site.xml` 或 `<mpp_user_name>-user.xml` 中定义或在外部表DDL中指定的 `jdbc.user` 属性设置的值，以及MPP数据库用户名。为了在外部数据存储区需要密码来验证连接用户的身份时有效地模拟用户，必须为可以模拟在该用户的 `<mpp_user_name>-user.xml` 属性覆盖文件中的每个用户指定 `jdbc.password` 设置。

有关每个服务器，每个MPP用户配置的更多信息，请参考配置PXF用户。

关于JDBC连接池

PXF JDBC连接器使用由[HikariCP]（https://github.com/brettwooldridge/HikariCP）实现的JDBC连接池。当用户查询或写入外部表时，连接器在首次遇到`jdbc.url`，`jdbc.user`，`jdbc.password`，连接属性和池属性设置的唯一组合时，将为关联的服务器配置建立连接池。连接器会根据某些连接和超时设置重用池中的连接。

对于给定的服务器配置，可能存在一个或多个连接池，并且用户访问指定同一服务器的不同外部表可能会共享一个连接池。

注意: 如果在服务器配置中启用了JDBC用户模拟，则JDBC连接器将为每个MPP数据库用户创建一个单独的连接池，该用户访问指定该服务器配置的任何外部表。

`jdbc.pool.enabled`属性控制着服务器配置的JDBC连接池。默认情况下启用连接池。要为服务器配置禁用JDBC连接池，请将属性设置为false：

```
<property>
    <name>jdbc.pool.enabled</name>
    <value>false</value>
</property>
```

如果为服务器配置禁用JDBC连接池，则PXF不会为该服务器重用JDBC连接。PXF为每个分区的查询创建到远程数据库的连接，并在该分区的查询完成时关闭该连接。

PXF公开了可以在JDBC服务器定义中配置的连接池属性。这些属性以`jdbc.pool.property.`前缀命名，并且应用于每个PXF JVM。JDBC连接器自动设置以下连接池属性和默认值：

属性	描述	默认值
jdbc.pool.property.maximumPoolSize	与数据库后端的最大连接数。	5
jdbc.pool.property.connectionTimeout	等待来自池的连接的最长时间（以毫秒为单位）。	30000
jdbc.pool.property.idleTimeout	最长时间（以毫秒为单位），在此时间之后，不活动的连接被视为空闲。	30000
jdbc.pool.property.minimumIdle	连接池中维护的最小空闲连接数。	0

您可以通过指定 `jdbc.pool.property。<HIKARICP_PROP_NAME>` 以及服务器的 `jdbc-site.xml` 配置文件中的所需值来为服务器配置设置其他HikariCP特定的连接池属性。还要注意，当JDBC连接器请求来自JDBC `DriverManager` 的连接时，它会传递您用 `jdbc.connection.property.` 前缀指定的任何属性。请参考上面的连接级属性。

调整最大连接池大小

为了不超过目标数据库所允许的最大连接数，并同时确保每个PXF JVM服务公平共享的JDBC连接，请根据MPP数据库集群的大小确定 `maxPoolSize` 的最大值。如下：

```
max_conns_allowed_by_remote_db / #_mpp_segment_hosts
```

例如，如果您的MPP数据库集群具有16个segment主机，并且目标数据库允许160个并发连接，则按以下方式计算 `maxPoolSize` ：

```
160 / 16 = 10
```

实际上，您可以选择将 `maxPoolSize` 设置为较低的值，因为每个JDBC查询的并发连接数取决于查询中使用的分区数。当查询不使用分区时，单个PXF JVM将为查询提供服务。如果查询使用12个分区，则PXF将建立与远程数据库的12个并发JDBC连接。理想情况下，这些连接在PXF JVM之间平均分配，但这不能保证。

JDBC命名查询配置

PXF*命名查询*是您配置的静态查询，并且PXF在远程SQL数据库中运行。

要配置和使用PXF JDBC命名查询：

1. 您定义查询在文本文件中。

2. 您向MPP数据库用户提供查询名称。

3. MPP数据库用户在MPP数据库外部表定义中引用查询。

每当用户在MPP数据库外部表上调用 `SELECT` 命令时，PXF都会运行查询。

定义命名查询

通过将查询语句添加到具有以下命名格式的文本文件中来创建命名查询： `<query_name>.sql` 。您可以为JDBC服务器配置定义一个或多个命名查询。每个查询必须驻留在单独的文本文件中。

您必须将查询文本文件放置在PXF JDBC服务器配置目录中，从该目录可以访问该查询文本文件。如果要使查询可用于多个JDBC服务器配置，则必须将查询文

本文件复制到每个JDBC服务器的配置目录中。

查询文本文件必须包含要在远程SQL数据库中运行的单个查询。您必须根据数据库支持的语法来构造查询。

例如，如果一个MySQL数据库有一个 `customers` 表和一个 `orders` 表，则可以在查询文本文件中包含以下SQL语句：

```
SELECT c.name, c.city, sum(o.amount) AS total, o.month
  FROM customers c JOIN orders o ON c.id = o.customer_id
  WHERE c.state = 'CO'
GROUP BY c.name, c.city, o.month
```

您可以选择为SQL语句提供结尾分号(`;`)。

查询命名

MPP数据库用户通过指定不带扩展名的查询文件名来引用命名查询。例如，如果您在名为 `report.sql` 的文件中定义查询，则该查询的名称为 `report` 。

命名查询与特定的JDBC服务器配置相关联。您将向MPP数据库用户提供可用的查询名称，允许您使用服务器配置创建外部表。

引用命名查询

当创建外部表时，MPP数据库用户指定 `query:<query_name>` 而不是远程SQL数据库表的名称。例如，如果查询在文件 `$PXF_CONF/servers/mydb/report.sql` 中定义，则 `CREATE EXTERNAL TABLE` `LOCATION` 子句将包含以下组件：

```
LOCATION ('pxf://query:report?PROFILE=JDBC&SERVER=mydb ...')
```

有关使用PXF JDBC命名查询的信息，请参考关于使用命名查询。

覆盖JDBC服务器配置

您可以通过在 `CREATE EXTERNAL TABLE` 命令 `LOCATION` 子句中通过自定义选项直接指定某些JDBC属性来覆盖JDBC服务器配置。有关其他信息，请参考通过DDL覆盖JDBC服务器配置。

示例配置过程

在配置JDBC连接器服务器之前，请确保已初始化PXF。

在此过程中，您将命名并添加PostgreSQL数据库的PXF JDBC服务器配置，并将服务器配置同步到MPP数据库集群。

1. 登录到您的MPP数据库主节点：

   ```
   $ ssh gpadmin@<gpmaster>
   ```

2. 选择JDBC服务器的名称。您将名称提供给MPP用户，您可以选择这些用户允许其以配置用户身份引用外部SQL数据库中的表。
   注意: 服务器名称 `default` 已保留。

3. 创建 `$PXF_HOME/servers/<server_name>` 目录。例如，使用以下命令来创建名为 `pg_user1_testdb` 的JDBC服务器配置：

   ```
   gpadmin@gpmaster$ mkdir $PXF_CONF/servers/pg_user1_testdb
   ```

4. 将PXF JDBC服务器模板文件复制到服务器配置目录。例如：

```
gpadmin@gpmaster$ cp $PXF_CONF/templates/jdbc-site.xml $PXF_CONF/s
```

5. 在您选择的编辑器中打开模板服务器配置文件，并为您的环境提供适当的属性值。例如，如果要在名为 `pgserverhost` 的主机上运行的PostgreSQL实例上为名为 `user1` 的用户配置对名为 `testdb` 的PostgreSQL数据库的访问：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>jdbc.driver</name>
        <value>org.postgresql.Driver</value>
    </property>
    <property>
        <name>jdbc.url</name>
        <value>jdbc:postgresql://pgserverhost:5432/testdb</value>
    </property>
    <property>
        <name>jdbc.user</name>
        <value>user1</value>
    </property>
    <property>
        <name>jdbc.password</name>
        <value>changeme</value>
    </property>
</configuration>
```

6. 保存更改并退出编辑器。

7. 使用 `pxf cluster sync` 命令将新的服务器配置复制到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

配置Hive访问

您可以使用PXF JDBC连接器从Hive检索数据。您还可以使用JDBC命名查询向Hive提交自定义SQL查询，并使用JDBC连接器检索结果。

本主题描述如何配置PXF JDBC连接器以访问Hive。使用JDBC配置Hive访问时，必须考虑Hive用户模拟设置，以及是否使用Kerberos保护Hadoop群集。

JDBC服务器配置

PXF JDBC连接器安装了通过JDBC访问Hive所需的JAR文件，`hive-jdbc-<version>.jar`和`hive-service-<version>.jar`，并自动注册了这些JAR。

为Hive访问配置PXF JDBC服务器时，必须像配置与SQL数据库的客户端连接时一样指定JDBC驱动程序类名称，数据库URL和客户端凭据。

要通过JDBC访问Hive，您必须在`jdbc-site.xml`服务器配置文件中指定以下属性和值：

属性	值
jdbc.driver	org.apache.hive.jdbc.HiveDriver
jdbc.url	jdbc:hive2://<hiveserver2_host>:<hiveserver2_port>/<database>

HiveServer2身份验证(`hive.server2.authentication`)和模拟(`hive.server2.enable.doAs`)属性的值以及Hive服务是否正在使用Kerberos身份验证将通知其他JDBC服务器的设置配置属性。这些属性在Hadoop集群的`hive-site.xml`配置文件中定义。您将需要获取这些属性的值。

下表枚举了PXF JDBC连接器支持的Hive2身份验证和模拟组合。它标识可能的Hive用户身份以及每个身份所需的JDBC服务器配置。

表标题键：

- *authentication -> Hive hive.server2.authentication设置*
- *enable.doAs -> Hive hive.server2.enable.doAs设置*
- *User Identity -> HiveServer2将用于访问数据的身份*
- *Configuration Required -> User Identity需要的PXF JDBC连接器或Hive配置*

authentication	enable.doAs	User Identity	Configuration Required
NOSASL	n/a	无认证	必须设置成 jdbc.connection.property.auth = noSasl
NONE，或未指定	TRUE	您提供的用户名	设置 jdbc.user
NONE，或未指定	TRUE	MPP用户名	设置 pxf.impersonation.jdbc = true
NONE，或未指定	FALSE	启动Hive的用户名，通常是 hive	None
KERBEROS	TRUE	PXF Kerberos主体中提供的身份，通常是 gpadmin	None
KERBEROS	TRUE	您提供的用户名	设置 hive.server2.proxy.user 中的 jdbc.url
KERBEROS	TRUE	MPP用户名	设置 pxf.impersonation.jdbc = true
KERBEROS	FALSE	PXF Kerberos主体中提供的身份，通常是 gpadmin	None

注意: Hive利用Kerberos身份验证时，还需要其他配置步骤。

示例配置步骤

执行以下过程为Hive配置PXF JDBC服务器：

1. 登录到您的MPP数据库主节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 选择JDBC服务器的名称。

3. 创建 `$PXF_HOME/servers/<server_name>` 目录。例如，使用以下命令创建名为 `hivejdbc1` 的JDBC服务器配置：

```
gpadmin@gpmaster$ mkdir $PXF_CONF/servers/hivejdbc1
```

4. 将PXF JDBC服务器模板文件复制到服务器配置目录。例如：

```
gpadmin@gpmaster$ cp $PXF_CONF/templates/jdbc-site.xml $PXF_CONF/s
```

5. 在您选择的编辑器中打开 `jdbc-site.xml` 文件，并设置 `jdbc.driver` 和 `jdbc.url` 属性。确保指定您的Hive主机，端口和数据库名称：

```
<property>
    <name>jdbc.driver</name>
    <value>org.apache.hive.jdbc.HiveDriver</value>
</property>
<property>
    <name>jdbc.url</name>
    <value>jdbc:hive2://<hiveserver2_host>:<hiveserver2_port>/<dat
</property>
```

6. 从Hadoop群集中获取 `hive-site.xml` 文件并检查该文件。

7. 如果 `hive-site.xml` 中的 `hive.server2.authentication` 属性设置为 `NOSASL` ，则HiveServer2不执行身份验证。将以下连接级别属性添加到 `jdbc-site.xml` 中：

   ```
   <property>
       <name>jdbc.connection.property.auth</name>
       <value>noSasl</value>
   </property>
   ```

   或者，您可以选择将 `;auth=noSasl` 添加到 `jdbc.url` 中。

8. 如果 `hive-site.xml` 中的 `hive.server2.authentication` 属性设置为 `NONE` ，或者未指定该属性，则必须设置 `jdbc.user` 属性。设置 `jdbc.user` 属性的值取决于 `hive-site.xml` 中的 `hive.server2.enable.doAs` 模拟设置：

   1. 如果将 `hive.server2.enable.doAs` 设置为 `TRUE` （默认值），则Hive代表连接到Hive的用户运行Hadoop操作。选择/执行以下选项之一：
      设置 `jdbc.user` 以指定对MPP数据库访问的所有Hive数据具有读取权限的用户。例如，要连接到Hive并以 `gpadmin` 用户身份运行所有请求：

      ```
      <property>
          <name>jdbc.user</name>
          <value>gpadmin</value>
      </property>
      ```

或，打开JDBC级用户模拟，以便PXF自动使用MPP数据库用户名连接到Hive：

```
<property>
    <name>pxf.impersonation.jdbc</name>
    <value>true</value>
</property>
```

如果以这种方式启用JDBC模拟，则既不能指定 `jdbc.user` *也不能在* `jdbc.url` *中包含设置。*

2. *如果需要，创建一个PXF用户配置文件来管理密码设置。*
3. *如果将* `hive.server2.enable.doAs` *设置为* `FALSE` *，则Hive将以启动HiveServer2进程的用户（通常是用户* `hive` *）运行Hadoop操作。在这种情况下，PXF会忽略* `jdbc.user` *设置。*

9. *如果* `hive-site.xml` *中的* `hive.server2.authentication` *属性设置为* `KERBEROS` *：*

    1. *确保按照为安全HDFS配置PXF中的说明为PXF启用了Kerberos身份验证。*
    2. *确保已将Hadoop集群配置为* `default` *PXF服务器。*
    3. *确保* `$PXF_CONF/servers/default/core-site.xml` *文件包含以下设置：*

    ```
    <property>
        <name>hadoop.security.authentication</name>
        <value>kerberos</value>
    </property>
    ```

    4. *在* `jdbc.url` *中添加* `saslQop` *属性，并将其设置为与* `hive-site.xml` *中的* `hive.server2.thrift.sasl.qop` *属性设置匹配。例如，如果* `hive-site.xml` *文件包含以下属性设置：*

```
<property>
    <name>hive.server2.thrift.sasl.qop</name>
    <value>auth-conf</value>
</property>
```

你可以将 `;saslQop=auth-conf` 添加到 `jdbc.url` 。

5. 将HiverServer2 `principal` 名称添加到 `jdbc.url` 中。例如：

```
jdbc:hive2://hs2server:10000/default;principal=hive/hs2server@REALM;saslQop=aut
```

6. 如果将 `hive.server2.enable.doAs` 设置为 `TRUE` （默认值），则Hive代表连接到Hive的用户运行Hadoop操作。选择/执行以下选项之一：
不要指定任何其他属性。在这种情况下，PXF使用PXF Kerberos主体（通常是 `gpadmin` ）中提供的身份来启动所有Hadoop访问。
或，在 `jdbc.url` 中设置 `hive.server2.proxy.user` 属性，以指定对所有Hive数据具有读取权限的用户。例如，要连接到Hive并以名为 `integration` 的用户身份运行所有请求，请使用以下 `jdbc.url` ：

```
jdbc:hive2://hs2server:10000/default;principal=hive/hs2server@REALM;saslQop=auth-
```

或，在 `jdbc-site.xml` 文件中启用PXF JDBC模拟，以便PXF自动使用MPP数据库用户名连接到Hive。例如：

```
<property>
    <name>pxf.impersonation.jdbc</name>
    <value>true</value>
</property>
```

如果启用JDBC模拟，则不得在 `jdbc.url` 中显式指定 `hive.server2.proxy.user` 。

7. 如果需要，创建一个PXF用户配置文件来管理密码设置。
8. 如果将 `hive.server2.enable.doAs` 设置为 `FALSE` ，则Hive将使用PXF Kerberos主体提供的身份（通常为 `gpadmin` ）运行Hadoop操作。

0. 保存更改并退出编辑器。

1. 使用 `pxf cluster sync` 命令将新的服务器配置复制到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

配置PXF客户端主机和端口(可选)

In this topic:

- 过程

默认情况下，在segment主机上启动的PXF代理侦听localhost上的5888端口。您可以将PXF配置为从其他端口号启动，或使用其他主机名或IP地址。要更改默认配置，您将设置以下标识的一个或两个环境变量：

Environment Variable	Description
PXF_HOST	主机名或IP地址。 默认主机名是 `localhost`
PXF_PORT	PXF代理用来侦听主机上的请求的端口号。 默认端口号是 `5888`

在每个segment主机上的gpadmin用户的.bashrc shell登录文件中设置环境变量。

以这种方式配置代理程序主机(和/或)端口时，必须重新启动MPP数据库和PXF。考虑在计划的停机时间内执行此配置。

过程

执行以下过程以在一台或多台MPP数据库主机上配置PXF代理主机(和/或)端口号：

1. 登录到您的MPP数据库主节点

```
$ ssh gpadmin@<gpmaster>
```

2. 对于每个MPP数据库segment主机

    1. 指定PXF代理的主机名或IP地址。
    2. 指定要在其上运行PXF代理的端口号。
    3. 登录到MPP segment主机：

    ```
    $ ssh gpadmin@<seghost>
    ```

    4. 使用编辑器中打开 `~/.bashrc` 文件
    5. 设置 `PXF_HOST` (和/或) `PXF_PORT` 环境变量。例如，要将PXF代理端口号设置为5998，请将以下内容添加到 `.bashrc` 文件中

    ```
    export PXF_PORT=5998
    ```

    6. 保存修改并退出。

3. 重启数据库。参阅Restarting MPP Database.

4. 重启每个segment上部署的PXF,参阅Restarting PXF.

升级PXF

In this topic:

- Step 1: PXF Pre-Upgrade Actions
- Step 2: Upgrading PXF

如果在当前的MPP数据库中安装使用PXF，则必须在升级到新版本MPP数据库时升级PXF服务。

PXF升级过程描述了如何在MPP数据库安装中升级PXF。 此过程使用PXF.from来表示当前安装的PXF版本, PXF.to表示升级新版本的MPP数据库时安装的PXF版本。

大多数PXF安装不需要修改PXF配置文件，并且可以进行无缝升级

Note: 从MPP Database版本5.12开始，PXF不再需要安装Hadoop客户端。PXF现在捆绑了它所依赖的所有JAR文件，并在运行时加载这些JAR。

PXF升级过程分为两部分。您在升级到MPP数据库的新版本之前执行一个过程，之后执行一个过程：

- Step 1: PXF Pre-Upgrade Actions
- GPDB升级到一个新的版本
- Step 2: Upgrading PXF GPDB升级到一个新的版本

Step 1: PXF Pre-Upgrade Actions

在升级到MPP数据库的新版本之前执行此过程：

1. 登录到GPDB master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 安装Stopping PXF 章节描述停止每个segment节点上的PXF

3. **如果你想从GPDB版本5.14或更早的版本开始升级**:

    1. 备份 PXF.from 在 `$GPHOME/pxf/conf/` 目录的配置文件。所有segment主机上的这些文件应该相同，因此您只需要从其中一个主机进行复制。 例如：

        ```
        gpadmin@gpmaster$ mkdir -p /save/pxf-from-conf
        gpadmin@gpmaster$ scp gpadmin@seghost1:/usr/local/mpp-db/pxf/co
        ```

    2. 请注意您可能已添加到PXF.from安装的任何自定义JAR文件的位置。 保存这些JAR文件的副本

4. 升级到新版本的MPP数据库，然后参照Step 2: Upgrading PXF.

Step 2: Upgrading PXF

升级到新版本的MPP数据库后，请执行以下步骤以升级和配置PXF.to软件：

1. 登录到GPDB master节点

    ```
    $ ssh gpadmin@<gpmaster>
    ```

2. 按照Initializing PXF的描述在每个segment主机上初始化PXF。

3. 默认情况下，在MPP数据库版本5.5.0及更高版本中启用PXF用户模拟。如果要从较旧的PXF.from版本升级，则必须为基础Hadoop服务配置用户模拟。有关说明，请参阅Configuring User Impersonation and Proxying，包括关闭PXF用户模拟的配置过程。

4. *如果你想从GPDB版本5.14或更早的版本开始升级：*

    1. *如果更新了PXF.from安装中的 `pxf-env.sh` 配置文件，请将这些更改重新应用于 `$PXF_CONF/conf/pxf-env.sh` 。 例如：*

        ```
        gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
           <update the file>
        ```

    2. *同样，如果您在PXF.from安装中更新了 `pxf-profiles.xml` 配置文件，请将这些更改重新应用到主机上的 `$PXF_CONF/conf/pxf-profiles.xml` 。*
    ***Note:** 从MPP Database版本5.12开始，PXF类的包名称已更改为使用前缀 `org.mpp.*` 。如果要从较旧的PXF.from版本升级并自定义 `pxf-profiles.xml` 文件，则必须在重新应用时更改对 `org.mpp.pxf.*` 的任何 `org.apache.hawq.pxf.*` 的引用。*
    3. *如果更新了PXF.from安装中的 `pxf-log4j.properties` 配置文件，请将这些更改重新应用到主机上的 `$PXF_CONF/conf/pxf-log4j.properties`*
    4. *如果在PXF.from安装中更新了 `pxf-public.classpath` 配置文件，请将文件中引用的每个JAR复制到naster主机的 `$PXF_CONF/lib`*
    5. *如果你将其他JAR文件添加到PXF.from中，请将它们复制到master主机上的 `$PXF_CONF/lib`*
    6. *从MPP Database版本 5.15 开始，PXF需要Hadoop配置文件放置在 `$PXF_CONF/servers/default` 目录中。如果在PXF.from安装中配置了PXF Hadoop连接器，请将 `/etc/<hadoop_service>/conf` 中的Hadoop配置文件复制到MPP Database主机上的 `$PXF_CONF/servers/default`*
    7. *从MPP Database版本 5.15 开始，PXF的默认Kerberos keytab文件位置是 `$PXF_CONF/keytabs` 。如果先前已将PXF配置为安全HDFS 且PXF密钥表文件位于PXF.from安装目录中(例如， `$GPHOME/pxf/conf`)，请考虑将keytab文件重定位到 `$PXF_CONF/keytabs` ,或者，更新 `$PXF_CONF/conf/pxf-env.sh` 文件中的 `PXF_KEYTAB` 属性设置以引用您的keytab文件。*

5. *如果要从MPP数据库5.18或更早版本升级：*

1. 现在，PXF捆绑了Hadoop 2.9.2版相关的JAR文件。 如果您在 `$PXF_CONF/lib` 中注册了其他与Hadoop相关的JAR文件，请确保这些库与Hadoop 2.9.2版兼容。
2. 现在，PXF JDBC连接器支持基于文件的服务器配置。如果选择将此新功能与引用外部SQL数据库的现有外部表一起使用，请参阅配置JDBC连接器中的配置说明，以获取更多信息。
3. 现在，PXF JDBC连接器支持语句查询超时。此特性需要JDBC驱动程序的显式支持。默认查询超时为 `0` ，请耐心等待。一些JDBC驱动程序在不完全支持语句查询超时功能的情况下支持 `0` 超时值。确保已注册的所有JDBC驱动程序都支持默认超时，或者更好的是，它完全支持此特性。 您可能需要更新JDBC驱动程序版本以获得此支持。有关使用PXF注册JAR文件的信息，请参考JDBC驱动程序JAR注册中的配置说明。
4. 如果您打算对整数类型使用Hive分区过滤，则必须在Hadoop群集的 `hive-site.xml` 和PXF用户配置 `$PXF_CONF/servers/default/hive-site.xml` 中设置 `hive.metastore.integral.jdo.pushdown` Hive属性。请参阅更新Hadoop配置。

6. ***如果您要从MPP数据库5.21.1或更早版本升级***：当您创建指定 `HiveText` 或 `HiveRC` 配置文件的外部表时，PXF Hive连接器不再支持在 `LOCATION` URI中提供 `DELIMITER=<delim>` 选项。如果您以前创建了一个在 `LOCATION` URI中指定 `DELIMITER` 的外部表，则必须删除该表，然后从 `LOCATION` 省略 `DELIMITER` 来重新创建它。您仍然需要在外部表格式设置选项中提供非默认定界符。

7. 将PXF配置从master主机同步到standby和每个MPP数据库segment主机。例如：

    ```
    gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
    ```

8. 按照Starting PXF章节描述在每个segment主机上启动PXF。

PXF的启动、停止和重启

In this topic:

- 启动PXF
- 停止PXF
- 重启PXF

PXF提供了两个管理命令:

- `pxf cluster` - 管理MPP数据库集群中的所有PXF服务实例
- `pxf` - 在特定的MPP数据库主机上管理PXF服务实例

`pxf cluster` 支持 `init`, `start`, `status`, `stop`, and `sync` 子命令。在MPP数据库master主机上运行 `pxf cluster` 子命令时，将在MPP数据库集群中的所有segment主机上执行该操作。PXF还在standby主机上运行 `init` 和 `sync` 命令。

`pxf` 支持 `init`, `start`, `stop`, `restart`, `status` 操作。这些操作在本地执行，也就是说，如果要在特定的MPP数据库segment主机上启动或停止PXF代理，需要登录到该主机并运行命令。

启动PXF

初始化PXF之后，必须在MPP数据库集群中的每个segment主机上启动PXF。PXF服务启动后，将以 `gpadmin` 用户身份在默认端口5888上运行。只有 `gpadmin` 用户可以启动和停止pxf服务。

如果要更改默认的PXF配置，则必须在启动PXF之前更新配置。

`$PXF_CONF/conf` 包含用户自定义的配置文件：

- `pxf-env.sh` - 运行时配置参数
- `pxf-log4j.properties` - 日志记录配置参数
- `pxf-profiles.xml` - 自定义配置文件定义

`pxf-env.sh` 包含以下用户可自定义的配置文件:

Parameter	Description	Default Value
JAVA_HOME	Java JRE家目录	/usr/java/default
PXF_LOGDIR	PXF日志目录	$PXF_CONF/logs
PXF_JVM_OPTS	PXF Java虚拟机的默认选项	-Xmx2g -Xms1g
PXF_KEYTAB	PXF服务Kerberos主体密钥表文件的绝对路径	$PXF_CONF/keytabs/pxf.service.keytab
PXF_PRINCIPAL	PXF服务Kerberos主体	gpadmin/_HOST@EXAMPLE.COM

您必须将对 `pxf-env.sh` ， `pxf-log4j.properties` 或 `pxf-profiles.xml` 所做的所有变更同步到MPP数据库集群，并在每个segment节点上(重新)启动PXF。

准备

在MPP数据库集群中启动PXF之前，请确保：

- MPP数据库集群已启动并正在运行
- PXF已经被初始化

过程

执行以下过程以在MPP数据库集群中的每个segment主机上启动PXF。

1. 登录mpp master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 在每个segment主机上运行 `pxf cluster start` 命令启动pxf服务

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster start
```

停止PXF

如果必须停止PXF,例如如果要升级PXF，则必须在MPP数据库集群中的每个segment主机上停止PXF。只有 `gpadmin` 用户可以停止PXF服务

准备

在MPP数据库集群中停止PXF之前，请确保MPP数据库集群已启动并正在运行。

过程

执行以下过程在MPP数据库集群中的每个segment主机上停止PXF。

1. 登录mpp master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 在每个segment主机上运行 `pxf cluster stop` 命令停止pxf服务。 例如:

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster stop
```

重启PXF

如果必须重新启动PXF，例如在 `$PXF_CONF/conf` 中更新了PXF用户配置文件，则可以在MPP数据库集群中先停止服务然后再启动PXF服务。

只有 `gpadmin` 用户可以重启PXF服务。

准备

在MPP数据库集群中重新启动PXF之前，请确保MPP数据库集群已启动并正在运行。

过程

执行以下过程在MPP数据库集群中的每个segment上重启PXF。

1. 登录mpp master节点:

```
$ ssh gpadmin@<gpmaster>
```

2. 重启PXF:

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster stop
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster start
```

授权用户访问PXF

In this topic:

- 在数据库中启用PXF
- 在数据库中禁用PXF
- 授权一个用户访问PXF

MPP平台扩展框架(PXF)实现了一个名为 `pxf` 的协议，您可以使用该协议创建引用外部数据存储中数据的外部表。PXF协议和Java服务打包作为MPP数据库扩展。

您必须在将要使用框架访问外部数据的每个数据库实例中启用PXF扩展。您还必须向需要访问权限的用户/角色明确授予 `pxf` 协议的 `GRANT` 权限。

在数据库中启用PXF

您必须在计划使用该扩展的每个MPP数据库中显式注册PXF扩展。您必须具有MPP数据库管理员权限才能注册扩展。

对要在其中使用PXF的**每个**数据库执行以下过程：

1. 使用 `gpadmin` 用户连接数据库

```
gpadmin@gpmaster$ psql -d <database-name> -U gpadmin
```

2. 创建PXF扩展名。您必须具有MPP数据库管理员权限才能创建扩展。 例如：

```
database-name=# CREATE EXTENSION pxf;
```

创建 `pxf` 扩展来注册 `pxf` 协议和PXF访问外部数据所需的调用处理程序。

在数据库中禁用PXF

当您不再希望在特定数据库上使用PXF时，必须显式删除该数据库的PXF扩展名。您必须具有MPP数据库管理员权限才能删除扩展。

1. 使用 `gpadmin` 用户连接数据库

```
gpadmin@gpmaster$ psql -d <database-name> -U gpadmin
```

2. 删除pxf扩展

```
database-name=# DROP EXTENSION pxf;
```

如果当前使用 `pxf` 协议定义了任何外部表，则 `DROP` 命令将失败。 如果选择强制删除这些外部表，请添加 `CASCADE` 选项。

授权一个用户访问PXF

要使用PXF读取外部数据，请使用 `CREATE EXTERNAL TABLE` 命令创建一个外部表，该命令指定 `pxf` 协议。您必须向所有需要此类访问权限的非 `SUPERUSER` 的

MPP数据库角色明确授予对pxf协议的 `SELECT` 权限。

要授予特定角色对 `pxf` 协议的访问权限，请使用 `GRANT` 命令。例如，要授予名为 `bill` 的角色对使用 `pxf` 协议创建的外部表引用的数据的读取访问权限，请执行以下操作：

```
GRANT SELECT ON PROTOCOL pxf TO bill;
```

要使用PXF将数据写入外部数据存储，请使用 `CREATE WRITABLE EXTERNAL TABLE` 命令创建一个外部表，该命令指定 `pxf` 协议。您必须向需要此类访问的所有非 `SUPERUSER` 的MPP数据库角色明确授予对 `pxf` 协议的 `INSERT` 权限。 例如：

```
GRANT INSERT ON PROTOCOL pxf TO bill;
```

注册PXF的jar依赖

您使用PXF访问外部系统上存储的数据。根据外部数据的存储，此访问可能需要您安装和/或配置外部数据存储的其他组件或服务。

PXF取决于这些附加组件提供的JAR文件和其他配置信息。 `$GPHOME/pxf/conf/pxf-private.classpath` 文件标识PXF内部JAR依赖性。在大多数情况下，PXF将管理 `pxf-private.classpath` 文件，并根据您使用的连接器根据需要添加条目。

如果您需要为PXF添加其他JAR依赖关系，例如JDBC驱动程序JAR文件，则必须登录到MPP数据库master主机，将JAR文件复制到PXF用户配置运行时库目录(`$PXF_CONF/lib`)， 将PXF配置同步到MPP数据库集群，然后在每个主机上重

新启动PXF。 例如：

```
$ ssh gpadmin@<gpmaster>
gpadmin@gpmaster$ cp new_dependent_jar.jar $PXF_CONF/lib/
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster stop
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster start
```

监控PXF

要显示PXF状态，必须在MPP数据库集群中的每个segment主机上显式请求PXF服务实例的状态。 `pxf cluster status` 命令可显示MPP数据库集群中所有segment主机上PXF服务实例的状态。 `pxf status` 显示本地(segment)主机上PXF服务实例的状态。

只有 `gpadmin` 用户可以请求PXF服务的状态。

执行以下步骤哦，以请求MPP数据库集群的PXF状态。

1. 登录gpdb集群的master主机

    ```
    $ ssh gpadmin@<gpmaster>
    ```

2. 运行 `pxf cluster status` 命令：

    ```
    gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster status
    ```

使用PXF访问Hadoop

In this topic:

- 架构
- 先决条件
- HDFS Shell 命令入门
- 连接器,数据格式和配置文件

PXF与Cloudera，Hortonworks Data Platform，MapR和通用Apache Hadoop发行版兼容。 PXF安装有HDFS，Hive和HBase连接器。 您可以使用这些连接器从以上Hadoop发行版访问各种格式的数据。

在先前版本的MPP数据库中，您可能使用 `gphdfs` 协议的外部表来访问存储在Hadoop中的数据。 MPP数据库6.0.0版本移除了 `gphdfs` 协议。 在MPP数据库6.x版本中使用PXF和 `pxf` 协议外部表来访问Hadoop。

架构

HDFS是Apache Hadoop使用的主要分布式存储机制。 当用户或应用程序在引用了HDFS文件的PXF外部表上执行查询时，MPP数据库master节点将查询分发给所有的segment节点。 每个Segment实例请求运行在其主机上的PXF代理. 当它从segment实例接受请求时, PXF代理:

1. 分配工作线程以处理来自segment实例的请求。
2. 调用HDFS Java API从HDFS NameNode请求HDFS文件的元数据信息。
3. 提供HDFS NameNode返回的元数据信息给segment实例。

图: PXF-到-Hadoop 架构

Segment实例使用它在MPP数据库中的 `gp_segment_id` 和元数据描述的文件块信息将查询所需数据的特定部分分配给自己。 然后，segment实例向PXF代理发送请求来读取分配的数据。 该数据可以存储在一个或多个HDSFS数据节点上。

PXF代理调用HDFS Java API来读取数据并将其传递给segment实例。 segment实例将其部分数据传递给MPP数据库master节点。 此通信跨segment节点和segment实例并行发生。

先决条件

在使用PXF处理Hadoop数据之前，请确保：

- 您已经配置并初始化PXF, 并且PXF正在每台segment主机上运行. 更多详情，请参阅配置 PXF。
- 您已经配置了计划使用的PXF Hadoop连接器。有关说明，请参阅配置PXF Hadoop 连接器。 如果您计划访问存储在Cloudera Hadoop集群中的JSON格式数据， 则PXF需要Cloudera 5.8或者更高的Hadoop发行版。
- 如果开启了用户模拟(默认), 确保您已将MPP数据库外部表需要访问到的HDFS文件及目录的读取(并根据需要写入) 权限，授予给了每个需要访问这些文件和目录的MPP数据库用户/角色的名称。 如果未开启用户模拟，您必须要将权限授予给 `gpadmin` 用户。
- MPP数据库segment主机和外部Hadoop系统之间的时间是同步的。

HDFS Shell 命令入门

Hadoop包含与HDFS文件系统直接交互的命令行工具。 这些工具支持典型的文件系统操作，包括复制和列出文件，更改文件权限等。

HDFS 文件系统命令语法为 `hdfs dfs <选项> [<文件>]` 。 在没有选项的情况下调用， `hdfs dfs` 将列出该工具支持的文件系统选项。

调用 `hdfs dfs` 命令的用户必须具有HDFS存储数据的读取权限才能列出和查看目录及文件内容, 并具有写入权限才能创建目录和文件。

PXF Hadoop主题使用的 `hdfs dfs` 选项包括:

选项	描述
`-cat`	显示文件内容
`-mkdir`	在HDFS中创建目录
`-put`	将文件从本地文件系统复制到HDFS中

例:

在HDFS中创建目录:

```
$ hdfs dfs -mkdir -p /data/exampledir
```

将文本文件从本地文件系统复制到HDFS:

```
$ hdfs dfs -put /tmp/example.txt /data/exampledir/
```

显示位于HDFS中的文本文件的内容:

```
$ hdfs dfs -cat /data/exampledir/example.txt
```

连接器,数据格式和配置文件

PXF Hadoop连接器提供内置配置文件以支持以下数据格式:

- Text
- Avro
- JSON
- ORC
- Parquet
- RCFile
- SequenceFile
- AvroSequenceFile

PXF Hadoop连接器公开以下配置文件以读取这些支持的数据格式，并在许多情况下写入:

数据源	数据格式	配置文件名称	弃用的配置文件名称
HDFS	单行的分隔文本	hdfs:text	HdfsTextSimple
HDFS	含有被双引号引起来的换行符的分隔文本	hdfs:text:multi	HdfsTextMulti
HDFS	Avro	hdfs:avro	Avro

数据源	数据格式	配置文件名称	弃用的配置文件名称
HDFS	JSON	hdfs:json	Json
HDFS	Parquet	hdfs:parquet	Parquet
HDFS	AvroSequenceFile	hdfs:AvroSequenceFile	n/a
HDFS	SequenceFile	hdfs:SequenceFile	SequenceWritable
Hive	TextFile 存储格式	Hive, HiveText	n/a
Hive	SequenceFile 存储格式	Hive	n/a
Hive	RCFile 存储格式	Hive, HiveRC	n/a
Hive	ORC 存储格式	Hive, HiveORC, HiveVectorizedORC	n/a
Hive	Parquet 存储格式	Hive	n/a
HBase	任意存储格式	HBase	n/a

使用 `CREATE EXTERNAL TABLE` 命令指定 `pxf` 协议时，提供配置文件名称，以创建引用Hadoop文件、目录或表的MPP数据库外部表。例如，以下命令创建一个使用默认服务器的外部表，并指定名为 `hdfs:text` 的配置文件：

```
CREATE EXTERNAL TABLE pxf_hdfs_text(location text, month text, num_or
   LOCATION ('pxf://data/pxf_examples/pxf_hdfs_simple.txt?PROFILE=hdf
FORMAT 'TEXT' (delimiter=E',');
```

读写HDFS文本数据

In this topic:

- 先决条件
- 读取文本数据
  - 示例：读取HDFS中的文本数据
- 读取含有双引号引起来的换行符的文本数据
  - 示例：在HDFS上读取多行文本数据
- 将文本文件写入HDFS
  - 示例：将文本数据写入HDFS

PXF HDFS连接器支持纯分隔和逗号分隔的值表单文本数据。 本节介绍如何使用PXF访问HDFS文本数据，包括如何创建引用了HDFS文件的外部表，查询外部表和向外部表写入数据。

先决条件

在尝试从HDFS读取或向HDFS写入数据之前，请确保已满足PXF Hadoop先决条件。

读取文本数据

当您在读取每行都是单条记录的纯文本分隔或csv数据时，请使用 `hdfs:text` 配置文件。以下语法创建了一个MPP数据库可读外部表，该表引用了HDFS上的此类文本文件：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-file>?PROFILE=hdfs:text[&SERVER=<serve
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-file>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE` 关键字必须指定为 `hdfs:text`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用 `default` 服务器。
FORMAT	当<path-to-hdfs-file>引用纯文本分隔数据时，请使用 `FORMAT 'TEXT'`。 当<path-to-hdfs-file>引用逗号分隔值数据时，请使用 `FORMAT 'CSV'`。
delimiter	数据中的分隔符。对于 `FORMAT 'CSV'`，默认的<delim_value>是逗号 `,`。当分隔符为转义序列时，在<delim_value>前面加上 `E`。示例：`(delimiter=E'\t')`，`(delimiter ':')`。

示例：读取HDFS中的文本数据

执行以下过程来创建示例文本文件，将文件复制到HDFS，然后使用 `hdfs:text` 配置文件和默认的PXF服务器创建两个PXF外部表来查询数据：

1. 为PXF示例数据文件创建HDFS目录。例如：

```
$ hdfs dfs -mkdir -p /data/pxf_examples
```

2. 创建名为 `pxf_hdfs_simple.txt` 的纯文本分割数据文件：

```
$ echo 'Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67' > /tmp/pxf_hdfs_simple.txt
```

   注意使用逗号 `,` 来分隔四个数据字段。.

3. 将数据文件添加到HDFS:

```
$ hdfs dfs -put /tmp/pxf_hdfs_simple.txt /data/pxf_examples/
```

4. 显示存储在HDFS中的 `pxf_hdfs_simple.txt` 文件的内容:

```
$ hdfs dfs -cat /data/pxf_examples/pxf_hdfs_simple.txt
```

5. 启动 `psql` 子系统:

```
$ psql -d postgres
```

6. 使用PXF `hdfs:text` 配置文件创建一个引用刚刚创建并添加到HDFS的 `pxf_hdfs_simple.txt` 文件的MPP数据库外部表:

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_textsimple(location text
            LOCATION ('pxf://data/pxf_examples/pxf_hdfs_simple.txt
          FORMAT 'TEXT' (delimiter=E',');
```

7. 查询外部表:

```
postgres=# SELECT * FROM pxf_hdfs_textsimple;
```

```
   location    | month | num_orders | total_sales
---------------+-------+------------+-------------
 Prague        | Jan   |        101 |     4875.33
 Rome          | Mar   |         87 |     1557.39
 Bangalore     | May   |        317 |     8936.99
 Beijing       | Jul   |        411 |    11600.67
(4 rows)
```

8. 创建第二个引用 `pxf_hdfs_simple.txt` 的外部表，这一次指定 `FORMAT` 为 `CSV` ：

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_textsimple_csv(location
            LOCATION ('pxf://data/pxf_examples/pxf_hdfs_simple.txt
          FORMAT 'CSV';
postgres=# SELECT * FROM pxf_hdfs_textsimple_csv;
```

当您为逗号分隔值数据指定 `FORMAT 'CSV'` 时，不需要提供 `delimiter` 分隔符选项，因为默认的分隔符是逗号。

读取含有双引号引起来的换行符的文本数据

使用 `hdfs:text:multi` 配置文件读取数据中含有引号引起来的换行符，单行或多行的分隔文本数据。以下语法创建一个引用此类文件的MPP外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-file>?PROFILE=hdfs:text:multi[&SERVER=
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-file>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE`关键字必须指定为`hdfs:text:multi`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用`default`服务器。
FORMAT	当<path-to-hdfs-file>引用纯文本分隔数据时，请使用`FORMAT 'TEXT'`。 当<path-to-hdfs-file>引用逗号分隔值数据时，请使用`FORMAT 'CSV'`。
delimiter	数据中的分隔符。对于`FORMAT 'CSV'`，默认的<delim_value>是逗号`,`。 当分隔符为转义序列时，在<delim_value>前面加上`E`。示例：`(delimiter=E'\t')`，`(delimiter ':')`。

示例：在HDFS上读取多行文本数据

执行以下步骤来创建示例文本文件，将文件复制到HDFS，然后使用PXF `hdfs：text：multi`配置文件和默认的PXF服务器创建MPP数据库可读的外部表来

查询数据：

1. 创建第二个分隔的纯文本文件：

```
$ vi /tmp/pxf_hdfs_multi.txt
```

2. 将以下数据复制/黏贴到 `pxf_hdfs_multi.txt` 中：

```
"4627 Star Rd.
San Francisco, CA  94107":Sept:2017
"113 Moon St.
San Diego, CA  92093":Jan:2018
"51 Belt Ct.
Denver, CO  90123":Dec:2016
"93114 Radial Rd.
Chicago, IL  60605":Jul:2017
"7301 Brookview Ave.
Columbus, OH  43213":Dec:2018
```

注意使用冒号 `:` 分隔三个字段。 另外请注意第一个(地址)字段周围的引号。 这个字段包含了一个嵌入的换行符，用于将街道地址与城市和州分开。

3. 将文本文件复制到HDFS：

```
$ hdfs dfs -put /tmp/pxf_hdfs_multi.txt /data/pxf_examples/
```

4. 使用 `hdfs:text:multi` 配置文件创建一个引用HDFS文件 `pxf_hdfs_multi.txt` 的外部表， 确保将 `:` (冒号) 标识为字段分隔符：

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_textmulti(address text,
             LOCATION ('pxf://data/pxf_examples/pxf_hdfs_multi.txt?
           FORMAT 'CSV' (delimiter ':');
```

Notice the alternate syntax for specifying the `delimiter`.

5. 查询 `pxf_hdfs_textmulti` 表：

```
postgres=# SELECT * FROM pxf_hdfs_textmulti;
```

```
         address          | month | year
--------------------------+-------+------
 4627 Star Rd.            | Sept  | 2017
 San Francisco, CA  94107
 113 Moon St.             | Jan   | 2018
 San Diego, CA  92093
 51 Belt Ct.              | Dec   | 2016
 Denver, CO  90123
 93114 Radial Rd.         | Jul   | 2017
 Chicago, IL  60605
 7301 Brookview Ave.      | Dec   | 2018
 Columbus, OH  43213
(5 rows)
```

将文本文件写入HDFS

PXF HDFS连接器 “hdfs:text” 配置文件支持将单行纯文本数据写入HDFS。 当您使用PXF HDFS连接器创建可写外部表时，可以指定在HDFS上的目录名称。 当您向可写外部表写入数据时，您写入的数据块将写入到指定目录中的一个或多个文件。

注意: 使用可写配置文件创建的外部表只能用于 `INSERT` 操作。如果要查询写入的数据，则必须另外创建一个引用HDFS目录的可读外部表。

使用以下语法创建一个引用HDFS目录的MPP可写外部表：

```
CREATE WRITABLE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-dir>
    ?PROFILE=hdfs:text[&SERVER=<server_name>][&<custom-option>=<value
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	*值*
<path-to-hdfs-dir>	*HDFS数据存储中目录的绝对路径*
PROFILE	*`PROFILE`关键字必须指定为`hdfs:text`*
SERVER=<server_name>	*PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用`default`服务器。*
<custom-option>	*<custom-option>描述见下方*
FORMAT	*当<path-to-hdfs-file>引用纯文本分隔数据时，请使用`FORMAT 'TEXT'`。 当<path-to-hdfs-file>引用逗号分隔值数据时，请使用`FORMAT 'CSV'`。*
delimiter	*数据中的分隔符。对于`FORMAT 'CSV'`，默认的<delim_value>是逗号`,`。 当分隔符为转义序列时，在<delim_value>前面加上`E`。示例：`(delimiter=E'\t')`，`(delimiter ':')`。*

关键字	值
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用MPP表相同的分布策略或<字段名>。 这样做可以避免数据加载操作中segment节点间额外的数据移动。

使用 `hdfs:text` 配置文件创建的可写外部表可以选择使用记录或块压缩。 PXF `hdfs:text` 配置文件支持以下压缩编解码器：

- org.apache.hadoop.io.compress.DefaultCodec
- org.apache.hadoop.io.compress.GzipCodec
- org.apache.hadoop.io.compress.BZip2Codec

您可以通过 `CREATE EXTERNAL TABLE` `LOCATION` 子句中自定义选择指定压缩编解码器。 `hdfs:text` 配置文件支持以下自定义写入选项：

选项	值描述

COMPRESSION_CODEC	压缩编解码器Java类名。 如果未提供此选项，MPP数据库不会执行压缩编码。支持的压缩编解码器包括：
	`org.apache.hadoop.io.compress.DefaultCodec`
	`org.apache.hadoop.io.compress.BZip2Codec`
	`org.apache.hadoop.io.compress.GzipCodec`

COMPRESSION_TYPE	采用的压缩类型; 支持的值为 `RECORD` (默认) 或 `BLOCK`
THREAD-SAFE	确定表查询是否可以在多线程模式下运行的布尔值。 默认为 `TRUE`。 将此选项设置为 `FALSE` 以处理单个线程中所有非线程安全操作 (例如，压缩)的请求。

示例：将文本数据写入HDFS

此示例使用了示例：读取HDFS中的文本数据中的数据格式。

列名	数据类型
location	text
month	text
number_of_orders	int
total_sales	float8

在此示例中您还可以选择在该练习创建的 `pxf_hdfs_textsimple` MPP数据库外部表。.

步骤

执行以下步骤，使用与上述相同的数据模式创建MPP数据库可写外部表，其中一个表将使用压缩。您将使用PXF `hdfs：text` 配置文件和默认的PXF服务器将数据写入基础HDFS目录。您还将创建一个单独的可读外部表，以读取您写入HDFS目录的数据。

1. 使用上述数据格式创建MPP数据库可写外部表。 写入HDFS目录 `/data/pxf_examples/pxfwritable_hdfs_textsimple1` 。 使用逗号 `,` 作为分隔符创建表:

   ```
   postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_hdfs_writabletbl_1(l
                 LOCATION ('pxf://data/pxf_examples/pxfwritable_hdfs_te
               FORMAT 'TEXT' (delimiter=',');
   ```

   您将 `FORMAT` 子句 `delimiter` 的值指定为单个ascii逗号字符 `,` 。

2. 通过在 `pxf_hdfs_writabletbl_1` 上调用SQL `INSERT` 命令，将一些单独的记录写入 `pxfwritable_hdfs_textsimple1` HDFS目录:

```
postgres=# INSERT INTO pxf_hdfs_writabletbl_1 VALUES ( 'Frankfurt'
postgres=# INSERT INTO pxf_hdfs_writabletbl_1 VALUES ( 'Cleveland'
```

3. (可选) 写入您在示例：读取HDFS中的文本数据创建的 `pxf_hdfs_textsimple` 表中的数据到 `pxf_hdfs_writabletbl_1`：

```
postgres=# INSERT INTO pxf_hdfs_writabletbl_1 SELECT * FROM pxf_hd
```

4. 在另一个终端窗口，显示刚添加到HDFS的数据:

```
$ hdfs dfs -cat /data/pxf_examples/pxfwritable_hdfs_textsimple1/*
Frankfurt,Mar,777,3956.98
Cleveland,Oct,3812,96645.37
Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67
```

因为您在创建可写外部表时使用了逗号 `,` 作为分隔符, 这个字符是HDFS数据中每条记录的字段分隔符.

5. MPP数据库不支持直接查询可写外部表。要查询刚刚添加到HDFS的数据，你必须创建一个引用这个HDFS目录的MPP可读外部表：

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_textsimple_r1(location t
           LOCATION ('pxf://data/pxf_examples/pxfwritable_hdfs_te
           FORMAT 'CSV';
```

在创建可读外部表时使用 `'CSV'` `FORMAT`，因为您在创建可写外部表时使用逗号 `,` 作为分隔符，即 `'CSV'` `FORMAT` 的默认分隔符。

6. 查询可读外部表:

```
postgres=# SELECT * FROM pxf_hdfs_textsimple_r1 ORDER BY total_sal
```

```
 location  | month | num_orders | total_sales
-----------+-------+------------+-------------
 Rome      | Mar   |         87 |     1557.39
 Frankfurt | Mar   |        777 |     3956.98
 Prague    | Jan   |        101 |     4875.33
 Bangalore | May   |        317 |     8936.99
 Beijing   | Jul   |        411 |    11600.67
 Cleveland | Oct   |       3812 |    96645.37
(6 rows)
```

`pxf_hdfs_textsimple_r1` 表包含您单独插入的记录，以及执行可选步骤时 `pxf_hdfs_textsimple` 表的完整内容。

7. 创建第二个MPP数据库可写外部表，这次使用Gzip压缩并使用冒号 `:` 作为分隔符：

```
postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_hdfs_writabletbl_2 (
            LOCATION ('pxf://data/pxf_examples/pxfwritable_hdfs_te
          FORMAT 'TEXT' (delimiter=':');
```

8. 通过直接写入 `pxf_hdfs_writabletbl_2` 表，将一些记录写入 `pxfwritable_hdfs_textsimple2` HDFS目录：

```
gpadmin=# INSERT INTO pxf_hdfs_writabletbl_2 VALUES ( 'Frankfurt',
gpadmin=# INSERT INTO pxf_hdfs_writabletbl_2 VALUES ( 'Cleveland',
```

9. 在另一个终端窗口中，显示您刚添加到HDFS的数据；使用 `hdfs dfs` `-text` 选项以文本的形式查看压缩数据：

```
$ hdfs dfs -text /data/pxf_examples/pxfwritable_hdfs_textsimple2/*
Frankfurt:Mar:777:3956.98
Cleveland:Oct:3812:96645.3
```

请注意冒号 `:` 是此HDFS数据的字段分隔符。
要从名为 `pxfwritable_hdfs_textsimple2` 的新创建的HDFS目录中查询数据，您可以创建一个MPP可读外部表并指定 `FORMAT 'CSV' (delimiter=':')` 。

从HDFS中读取Avro数据

In this topic:

- 先决条件
- 使用Avro数据
  - 数据类型映射
  - Avro 模式和数据
- 创建外部表
- 示例：读取Avro数据
  - 创建模式
  - 创建Avro数据文件(JSON)
  - 使用hdfs:avro配置文件查询

使用PXF HDFS连接器读取Avro格式数据。本节描述如何使用PXF访问HDFS中的Avro数据，包括如何创建和查询引用HDFS中Avro文件的外部表。

先决条件

在尝试从HDFS读取或向HDFS写入数据之前，请确保已满足PXF Hadoop先决条件。

使用Avro数据

Apache Avro是一个数据序列化框架，其中的数据都以压缩二进制格式序列化。 Avro在Json中指定数据类型。 Avro格式数据具有独立的模式, 这也在Json中定义。 Avro模式及其数据完全是自描述的。

数据类型映射

Avro支持原生数据类型和复杂数据类型。

要在MPP数据库中表示Avro原生数据类型，请将数据值映射到相同类型的MPP列。

Avro支持复杂数据类型数组(array), 映射(map), 记录(record), 枚举(enumeration)以及固定长度类型(fixed type)。 将这些复杂数据类型的顶层字段映射为MPP数据库的 `TEXT` 类型。虽然MPP本身并不支持这些类型，您可以创建MPP数据库函数或应用程序代码来提取或进一步处理这些复杂数据类型的子部件。

下表总结了Avro数据的外部映射规则。

Avro数据类型	PXF/MPP 数据类型
boolean	boolean
bytes	bytea
double	double
float	real

Avro数据类型	PXF/MPP 数据类型
int	int or smallint
long	bigint
string	text
复杂类型: Array, Map, Record, or Enum	text, 并在集合项、映射的键值对和记录数据之间插入分隔符。
复杂类型: Fixed	bytea
并集	对于原始数据类型或复杂数据类型，遵循上述约定，具体取决于并集；支持Null值。

Avro 模式和数据

Avro 模式使用json定义，由上面数据类型映射部分中的原生类型和复杂类型组成。Avro 模式文件通常具有 `.avsc` 后缀。

Avro模式文件中的字段是通过对象数组定义的，每个对象都由名称和类型指定。

创建外部表

使用 `hdfs:avro` 配置文件读取HDFS中的Avro格式数据，以下语法创建了引用该配置文件的可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-file>?PROFILE=hdfs:avro[&<custom-optio
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-file>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE`关键字必须指定为`hdfs:avro`.
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用`default`服务器。
<custom-option>	<custom-option>描述见下方
FORMAT 'CUSTOM'	在`hdfs:avro`配置文件中使用`FORMAT` `'CUSTOM'` 。`CUSTOM` `FORMAT`要求您指定为`(FORMATTER='pxfwritable_import')`。

对于复杂数据类型，PXF `hdfs:avro`配置文件在集合项和值之间插入默认分隔符。您可以通过指定在`CREATE EXTERNAL TABLE`命令中指定`hdfs:avro`自定义选项来使用非默认分隔符。

`hdfs:avro`配置文件支持以下<custom-option>：

选项关键字	描述
COLLECTION_DELIM	当PXF将Avro复杂数据类型映射到文本列时，放置在顶层数组、映射或记录字段中的条目之间的分隔符。默认为逗号`,`字符。
MAPKEY_DELIM	当PXF将Avro复杂数据类型映射到文本列时，放置在映射项的键和值之间的分隔符。默认为冒号`:`字符。

选项关键字	描述
RECORD KEY_DELIM	当PXF将Avro复杂数据类型映射到文本列时，放置在字段名称和记录条目之间的分隔符。 默认为冒号 `:` 字符。

示例：读取Avro数据

本节中的示例将对具有以下字段名称和数据类型模式的Avro数据进行操作:

- id - 长整型
- username - 字符串
- followers - 字符串数组
- fmap - 长整型映射
- relationship - 枚举类型
- address - 由街道号码(整型)、街道名称(字符串)、以及城市(字符串)组成的记录类型

创建模式

执行以下操作创建一个Avro模式，以表示上述模式。

1. 创建一个名为 `avro_schema.avsc` 的文件:

```
$ vi /tmp/avro_schema.avsc
```

2. 将以下文本复制并粘贴到 `avro_schema.avsc` 中：

```
{
"type" : "record",
  "name" : "example_schema",
  "namespace" : "com.example",
  "fields" : [ {
    "name" : "id",
    "type" : "long",
    "doc" : "Id of the user account"
  }, {
    "name" : "username",
    "type" : "string",
    "doc" : "Name of the user account"
  }, {
    "name" : "followers",
    "type" : {"type": "array", "items": "string"},
    "doc" : "Users followers"
  }, {
    "name": "fmap",
    "type": {"type": "map", "values": "long"}
  }, {
    "name": "relationship",
    "type": {
        "type": "enum",
        "name": "relationshipEnum",
        "symbols": ["MARRIED","LOVE","FRIEND","COLLEAGUE","STRANGE
    }
  }, {
    "name": "address",
    "type": {
        "type": "record",
        "name": "addressRecord",
        "fields": [
            {"name":"number", "type":"int"},
            {"name":"street", "type":"string"},
            {"name":"city", "type":"string"}]
    }
  } ],
  "doc:" : "A basic schema for storing messages"
}
```

创建Avro数据文件(JSON)

执行以下步骤来创建符合上述模式的示例Avro文件。

1. 创建一个名为 `pxf_avro.txt` 的文本文件:

```
$ vi /tmp/pxf_avro.txt
```

2. 在 `pxf_avro.txt` 中输入以下数据：

```
{"id":1, "username":"john","followers":["kate", "santosh"], "relat

{"id":2, "username":"jim","followers":["john", "pam"], "relationsh
```

示例数据使用逗号 `,` 分隔顶层字段，并使用冒号 `:` 分隔键-值对以及记录字段和值.

3. 将文本文件转换为Avro格式文件。有多种方法可以通过编程方式和通过命令行转换。在这个例子中，我们使用Java Avro tools。 `avro-tools-1.8.1.jar` jar文件放置在当前目录中：

```
$ java -jar ./avro-tools-1.8.1.jar fromjson --schema-file /tmp/avr
```

生成的Avro二进制文件被写入 `/tmp/pxf_avro.avro` 。

4. 将生成的Avro文件复制到HDFS：

```
$ hdfs dfs -put /tmp/pxf_avro.avro /data/pxf_examples/
```

使用hdfs:avro配置文件查询

执行以下来创建和查询引用您在上一节添加到HDFS的`pxf_avro.avro`文件。创建表时：

- 使用PXF默认服务器。
- 将顶层原生字段`id`(长整型)和`username`(字符串类型)映射到其等效的MPP的数据库类型(bigint和text)。
- 将剩下的复杂类型映射为text类型
- 使用`hdfs:avro`配置文件的自定义选项设置记录(record)、映射(map)和集合(collection)的分隔符

1. 使用`hdfs:avro`配置文件从`pxf_avro.avro`文件创建可查询外部表：

```
postgres=# CREATE EXTERNAL TABLE pxf_hdfs_avro(id bigint, username
            LOCATION ('pxf://data/pxf_examples/pxf_avro.avro?PROFI
          FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

2. 对`pxf_hdfs_avro`表执行简单查询：

```
postgres=# SELECT * FROM pxf_hdfs_avro;
```

```
 id | username |   followers    |        fmap          | relationsh
----+----------+----------------+----------------------+------------
  1 | john     | [kate,santosh] | {kate:10,santosh:4} | FRIEND
  2 | jim      | [john,pam]     | {pam:3,john:3}      | COLLEAGUE
(2 rows)
```

外部表的简单查询显示了复杂数据类型的组成部分，这些组成部分由`CREATE EXTERNAL TABLE`中指定的分隔符分隔。

3. 根据您的应用程序处理文本列中被分隔的部分。例如，以下命令使用MPP数据库内部的`string_to_array`函数将`followers`字段中的条目转换为新视图中的文本数组列。

```
postgres=# CREATE VIEW followers_view AS
SELECT username, address, string_to_array(substring(followers FROM
    AS followers
  FROM pxf_hdfs_avro;
```

4. 根据特定的关注者是否出现在视图中来执行查询并过滤行：

```
postgres=# SELECT username, address FROM followers_view WHERE foll
```

```
 username |                    address
----------+-----------------------------------------------
 jim      | {number:9,street:deer creek,city:palo alto}
```

从HDFS从读取JSON数据

In this topic:

- 先决条件
- 使用JSON数据
  - JSON到MPP数据库数据类型映射
  - JSON数据读取模式
- 将样本JSON数据加载到HDFS
- 创建外部表
- 示例：读取单行记录的JSON文件

使用PXF HDFS连接器读取JSON格式数据的数据。本节描述了如何使用PXF访问HDFS中的JSON数据，包括如何创建和查询引用HDFS中JSON文件的外部表。

先决条件

在尝试从HDFS读取或向HDFS写入数据之前，请确保已满足PXF Hadoop先决条件。

使用JSON数据

JSON是基于文本的数据交换格式。JSON数据通常存储在带有 `.json` 后缀的文件中。

`.json` 文件将包含对象的集合。JSON对象是无序键/值对的集合。值可以是字符串(string)、数字(number)、true、false、null或对象(object)或数组(array)。您可以定义嵌套的JSON对象和数组。

样本JSON数据文件内容：

```
{
  "created_at":"MonSep3004:04:53+00002013",
  "id_str":"384529256681725952",
  "user": {
    "id":31424214,
    "location":"COLUMBUS"
  },
  "coordinates":{
    "type":"Point",
    "values":[
       13,
       99
    ]
  }
}
```

在上述的示例中， `user` 是一个由名为 `id` 和 `location` 的字段组成的对象。要将 `user` 对象中的嵌套字段指定为MPP数据库外部表列，请使用 `.` 映射：

```
user.id
user.location
```

`coordinates`是一个由名为`type`字段和名为`values`的整数数组组成的对象。使用`[]`来指定特定`values`数组中特定的元素作为MPP数据库外部表的列：

```
coordinates.values[0]
coordinates.values[1]
```

有关JSON语法的详细信息，请参阅JSON 简介。

JSON到MPP数据库数据类型映射

要在MPP数据库中表示JSON数据，请将使用基本数据类型的值映射到相同类型的MPP数据库列。JSON支持复杂的数据类型包括投影和数组。 使用N级投影将嵌套对象和数组的成员映射到基本数据类型。

下表总结了JSON数据的外部映射规则。

表1. JSON映射

JSON 数据类型	PXF/MPP Data Type
基本类型(integer, float, string , boolean, null)	使用相应的MPP数据库内置数据类型；请参阅MPP数据库数据类型.
数组(Array)	使用`[]`括号标识基本数据类型数组特定成员的索引。
对象(Object)	使用`.`点号指定基本类型的每一层投影(嵌套)。

JSON数据读取模式

PXF支持两种数据读取模式。默认模式是每行一条完成的JSON记录。PXF还支持对跨多行的JSON记录进行操作的读取模式。

在接下来的示例中，您将使用两种模式对样本数据集进行操作。样本数据集的模式定义具有以下成员名称和数据类型的对象：

- “created_at” - text
- “id_str” - text
- “user” - object
  - “id” - integer
  - “location” - text
- “coordinates” - object (可选)
  - “type” - text
  - “values” - array
    - [0] - integer
    - [1] - integer

每行单条JSON记录数据如下：

```
{"created_at":"FriJun0722:45:03+00002013","id_str":"3431365513221365
"id":395504494,"location":"NearCornwall"},"coordinates":{"type":"Poi
: [ 6, 50 ]}},
{"created_at":"FriJun0722:45:02+00002013","id_str":"343136547115253
"id":26643566,"location":"Austin,Texas"}, "coordinates": null},
{"created_at":"FriJun0722:45:02+00002013","id_str":"343136547136233
"id":287819058,"location":""}, "coordinates": null}
```

这是用于多行JSON数据的数据集:

```
{
  "root":[
    {
      "record_obj":{
        "created_at":"MonSep3004:04:53+00002013",
        "id_str":"384529256681725952",
        "user":{
          "id":31424214,
          "location":"COLUMBUS"
        },
        "coordinates":null
      },
      "record_obj":{
        "created_at":"MonSep3004:04:54+00002013",
        "id_str":"384529260872228864",
        "user":{
          "id":67600981,
          "location":"KryberWorld"
        },
        "coordinates":{
          "type":"Point",
          "values":[
             8,
             52
          ]
        }
      }
    }
  ]
}
```

在下一节中您将为示例数据集创建JSON文件，并将其添加到HDFS中。

将样本JSON数据加载到HDFS

PXF HDFS连接器读取存储在HDFS本地的JSON文件。在使用MPP数据库查询JSON格式数据之前，该数据必须存在于HDFS中。

将上面的单行JSON记录样本数据集复制并黏贴到名为 `singleline.json` 的文件中。同样的，将多行JSON记录数据集复制并黏贴到名为 `multiline.json` 的文件中。

注意: 确保JSON文件中**没有**空白行。

将您刚创建的文件复制到HDFS中。如果您在上一个练习中没有创建 `/data/pxf_examples` 目录，请创建它。例如：

```
$ hdfs dfs -mkdir /data/pxf_examples
$ hdfs dfs -put singleline.json /data/pxf_examples
$ hdfs dfs -put multiline.json /data/pxf_examples
```

数据加载到HDFS后，您可以使用MPP数据库和PXF查询和分析JSON数据。

创建外部表

使用 `hdfs:json` 配置文件从HDFS中读取JSON格式的文件。以下语法创建一个引用此类文件的MPP数据库可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-file>?PROFILE=hdfs:json[&SERVER=<serve
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-file>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE` 关键字必须指定为 `hdfs:json`

关键字	值
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用 default 服务器。
<custom-option>	<custom-option>描述见下方
FORMAT 'CUSTOM'	在 hdfs:json 配置文件中使用 FORMAT 'CUSTOM' 。CUSTOM FORMAT 要求您指定为 (FORMATTER='pxfwritable_import') 。

PXF支持单行和多行JSON记录。当您想读取多行JSON记录时，必须提供 IDENTIFIER <custom-option> 和值。使用这个<custom-option>标识JSON记录对象中第一个字段的成员名称：

选项关键字	语法、示例	描述
IDENTIFIER	&IDENTIFIER=<value> &IDENTIFIER=created_at	仅在访问多行JSON记录时，才必须在 LOCATION 字符串中指定 IDENTIFIER 关键字和值。使用这个值标识JSON记录对象中第一个字段的成员名称。

示例：读取单行记录的JSON文件

使用以下CREATE EXTERNAL TABLESQL命令来创建可读的外部表，该表引用每条记录单行JSON数据文件并使用PXF默认服务器。

```
CREATE EXTERNAL TABLE singleline_json_tbl(
  created_at TEXT,
  id_str TEXT,
  "user.id" INTEGER,
  "user.location" TEXT,
  "coordinates.values[0]" INTEGER,
  "coordinates.values[1]" INTEGER
)
LOCATION('pxf://data/pxf_examples/singleline.json?PROFILE=hdfs:json'
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

注意使用 `.` 投影访问 `user` 和 `coordinates` 对象中的嵌套字段。还要注意使用 `[]` 来访问 `coordinates.values[]` 数组中的特定元素。

查询外部表中的JSON数据:

```
SELECT * FROM singleline_json_tbl;
```

示例：读取多行记录的JSON文件

从多行JSON记录文件中创建可读外部表的SQL命令和上面单行JSON记录的非常类似。当您要读取多行JSON记录时，您必须额外指定 `LOCATION` 子句的 `IDENTIFIER` 关键字和值。例如：

```
CREATE EXTERNAL TABLE multiline_json_tbl(
  created_at TEXT,
  id_str TEXT,
  "user.id" INTEGER,
  "user.location" TEXT,
  "coordinates.values[0]" INTEGER,
  "coordinates.values[1]" INTEGER
)
LOCATION('pxf://data/pxf_examples/multiline.json?PROFILE=hdfs:json&IL
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

`created_at`标识示例数据模式中JSON记录`record_obj`中第一个字段的成员名称。

查询外部表中的JSON数据：

```
SELECT * FROM multiline_json_tbl;
```

读写HDFS Parquet数据

In this topic:

- 先决条件
- 数据类型映射
- 创建外部表
- 示例

使用PXF HDFS连接器读写Parquet格式数据。本节介绍如何读写以Parquet格式存储的HDFS文件，包括如何创建、查询和写入引用HDFS文件的外部表。

PXF当前仅支持读写基本Parquet数据类型。

PXF Parquet写入支持是一个Beta功能。

先决条件

在尝试从HDFS读取或向HDFS写入数据之前，请确保已满足PXF Hadoop先决条件。

数据类型映射

要在MPP数据库中读写Parquet基本数据类型，请将Parquet数据值映射到相同类型的MPP数据库列。下表总结了外部映射规则：

Parquet数据类型	PXF/MPP数据类型
boolean	Boolean
byte_array	Bytea, Text
double	Float8
fixed_len_byte_array	Numeric
float	Real
int_8, int_16	Smallint, Integer
int64	Bigint
int96	Timestamp, Timestamptz

写入Parquet时：

- PXF将 `timestamp` 本地化为当前系统时区，并将其转换为通用时间(UTC)，然后最终转换为 `int96` 。
- PXF将 `timestamp` 转换为UTC `timestamp` ，然后转换为 `int96` 。在此转换过程中，PXF会丢失时区信息。

创建外部表

PXF HDFS连接器 `hdfs:parquet` 配置文件支持读写Parquet格式的HDFS数据。当您将记录插入可写外部表中时，您插入的数据块将写入指定目录中一个或多个文件。

使用以下语法创建引用HDFS目录的MPP数据库外部表：

```
CREATE [WRITABLE] EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-dir>
    ?PROFILE=hdfs:parquet[&SERVER=<server_name>][&<custom-option>=<va
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import'|'pxfwritable_export'
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-file>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE` 关键字必须指定为 `hdfs:parquet`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	<custom-option>描述见下方

关键字	值
FORMAT 'CUSTOM'	使用 FORMAT 'CUSTOM' 时指定 (FORMATTER='pxfwritable_export') (写入) 或 (FORMATTER='pxfwritable_import') (读取)。
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用与MPP表相同的分布策略或<字段名>。 这样做可以避免数据加载操作中segment节点间额外的数据移动。

PXF hdfs:parquet 配置文件支持与编码和压缩有关的写入选项。您可以在 CREATE WRITABLE EXTERNAL TABLE LOCATION 子句中指定这些写入选项。 hdfs:parquet 配置文件支持以下自定义选项：

写入选项	值描述
COMPRESSION_CODEC	压缩编码器别名。 用于写入Parquet数据受支持的压缩编码器包括： snappy , gzip , lzo , 和 uncompressed 。 如果未提供此选项，PXF将使用 snappy 压缩编码器来压缩数据。
ROWGROUP_SIZE	Parquet文件由一个或多个行组组成，将数据逻辑划分为行。 ROWGROUP_SIZE 标识行组的大小(以字节为单位)。默认的行组大小为 8 * 1024 * 1024 字节。
PAGE_SIZE	行组由划分为页面的列块组成。 PAGE_SIZE 是此类页面的大小(以字节为单位)。默认的页面大小为 1024 * 1024 字节。
DICTIONARY_PAGE_SIZE	当PXF写入Parquet文件时，默认情况下启用字典编码。每列、每行组只有一个字典页面。 DICTIONARY_PAGE_SIZE 与 PAGE_SIZE 类似，但是对于字典而言：默认字典页面大小为 512 * 1024 字节。
PARQUET_VERSION	Parquet版本；支持 v1 和 v2 。默认的Parquet版本为 v1 。

注意: 如果您不希望PXF压缩数据，则必须明确指定 `uncompressed` *。*

使用PXF写入HDFS的Parquet文件具有以下命名格式： `<file>.<compress_extension>.parquet` *，例如* `1547061635-0000004417_0.gz.parquet` *.*

示例

本示例采用在示例：读取HDFS中的文本数据中介绍的数据模式。

列名	*数据类型*
location	text
month	text
number_of_orders	int
total_sales	float8

在此示例中，您创建一个Parquet格式的可写外部表，该表使用默认的PXF服务器引用HDFS中的Parquet格式的数据，将一些数据插入表中，然后创建一个可读外部表以读取数据。

1. *使用* `hdfs:parquet` *配置文件创建一个可写外部表。例如：*

```
postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_tbl_parquet (locatio
    LOCATION ('pxf://data/pxf_examples/pxf_parquet?PROFILE=hdfs:pa
  FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

2. *通过直接写入* `pxf_tbl_parquet` *表将一些记录写入HDFS的* `pxf_parquet` *目录中。例如：*

```
postgres=# INSERT INTO pxf_tbl_parquet VALUES ( 'Frankfurt', 'Mar'
postgres=# INSERT INTO pxf_tbl_parquet VALUES ( 'Cleveland', 'Oct'
```

3. 回想一下，MPP 数据库不支持直接查询一个可写外部表。要读取 `pxf_parquet` 中的数据，请创建一个引用此HDFS目录的可读外部表：

```
postgres=# CREATE EXTERNAL TABLE read_pxf_parquet(location text, m
    LOCATION ('pxf://data/pxf_examples/pxf_parquet?PROFILE=hdfs:pa
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

4. 查询可读外部表 `read_pxf_parquet` ：

```
postgres=# SELECT * FROM read_pxf_parquet ORDER BY total_sales;
```

```
 location  | month | number_of_orders | total_sales
-----------+-------+------------------+-------------
 Frankfurt | Mar   |              777 |     3956.98
 Cleveland | Oct   |             3812 |     96645.4
(2 rows)
```

读写HDFS SequenceFile数据

In this topic:

- 先决条件
- 创建外部表
- 读写二进制数据
- 读取记录键
  - 示例： 使用记录键

PXF HDFS连接器支持SequenceFile格式的二进制数据。本节描述如何使用PXF读写HDFS SequenceFile数据，包括如何创建引用HDFS文件的外部表，查询和向外部表写入数据。

先决条件

在尝试从HDFS读取或向HDFS写入数据之前，请确保已满足PXF Hadoop先决条件。

创建外部表

PXF HDFS连接器 `hdfs:SequenceFile` 配置文件支持读写HDFS中SequenceFile格式的二进制数据。 当您将记录插入可写外部表中时，您插入的数据块将写入指定目录中的一个或多个文件。

注意: 使用可写配置创建的外部表仅INSERT操作。 如果要查询插入的数据，则必须单独创建一个引用相关HDFS目录的可读外部表。

使用以下语法创建一个引用HDFS目录的MPP数据库外部表：

```
CREATE [WRITABLE] EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-hdfs-dir>
    ?PROFILE=hdfs:SequenceFile[&SERVER=<server_name>][&<custom-optio
FORMAT 'CUSTOM' (<formatting-properties>)
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-hdfs-dir>	HDFS数据存储中目录或文件的绝对路径
PROFILE	`PROFILE` 关键字必须指定为 `hdfs:SequenceFile`.
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
<custom-option>	<custom-option> 描述见下方.
FORMAT	使用 `FORMAT` ’`CUSTOM`’ 时指定 `(FORMATTER='pxfwritable_export')` (写入) 或 `(FORMATTER='pxfwritable_import')` (读取).
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用与MPP表相同的分布策略或<字段名>。 这样做可以避免数据加载操作中segment节点间额外的数据移动。

SequenceFile 格式数据可以选择采用record或block压缩。 PXF `hdfs:SequenceFile` 配置支持以下压缩编解码器：

- org.apache.hadoop.io.compress.DefaultCodec
- org.apache.hadoop.io.compress.BZip2Codec

使用 hdfs:SequenceFile 配置文件写入SequenceFile格式数据时，则必须提供Java类的名称，以用于序列化/反序列化二进制数据。 这个类必须为数据模式中引用的每种数据类型提供读写方法。

您可以通过 CREATE EXTERNAL TABLE LOCATION 子句的自定义选项指定压缩编解码器和Java 序列化类。 hdfs:SequenceFile 配置文件支持以下自定义选项：

选项	值描述
COMPRESSION_CODEC	压缩编码器类名称。 如果此选项未提供，MPP 数据库不执行数据压缩。 支持的压缩编解码器包括： org.apache.hadoop.io.compress.DefaultCodec org.apache.hadoop.io.compress.BZip2Codec org.apache.hadoop.io.compress.GzipCodec
COMPRESSION_TYPE	采用的压缩类型; 支持的值有 RECORD (默认) 或 BLOCK.
DATA-SCHEMA	编写器序列化/反序列化类的名称。 此类所在的jar文件必须位于PXF类路径中。 hdfs:SequenceFile 配置文件需要此选项，并且没有默认值。
THREAD-SAFE	确定表查询是否可以在多线程模式下运行的布尔值。 默认为 TRUE。 将此选项设置为 FALSE 以处理单个线程中所有非线程安全操作 (例如，压缩)的请求。

读写二进制数据

当您要读写HDFS中 SequenceFile 格式的数据，请使用HDFS 连接器 hdfs:SequenceFile 配置文件。 这种类型的文件由二进制键/值对组成。

SequenceFile 格式是MapReduce作业之间常见的数据传输格式。

示例： 将二进制数据写入HDFS

在此示例中，您将创建一个名为 `PxfExample_CustomWritable` 的Java类，该类将对先前示例中使用的示例模式中的字段进行序列化/反序列化。然后，您将使用此类访问通过 `hdfs:SequenceFile` 配置文件创建的可写外部表，该表使用默认的PXF服务器。

执行以下步骤来创建Java类和可写外部表。

1. 准备创建示例Java类:

```
$ mkdir -p pxfex/com/example/pxf/hdfs/writable/dataschema
$ cd pxfex/com/example/pxf/hdfs/writable/dataschema
$ vi PxfExample_CustomWritable.java
```

2. 将以下文本复制并黏贴到 `PxfExample_CustomWritable.java` 文件中：

```
package com.example.pxf.hdfs.writable.dataschema;

import org.apache.hadoop.io.*;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.lang.reflect.Field;

/**
* PxfExample_CustomWritable class - used to serialize and deserial
* text, int, and float data types
*/
public class PxfExample_CustomWritable implements Writable {

public String st1, st2;
public int int1;
public float ft;
```

```
public PxfExample_CustomWritable() {
    st1 = new String("");
    st2 = new String("");
    int1 = 0;
    ft = 0.f;
}

public PxfExample_CustomWritable(int i1, int i2, int i3) {

    st1 = new String("short_string___" + i1);
    st2 = new String("short_string___" + i1);
    int1 = i2;
    ft = i1 * 10.f * 2.3f;

}

String GetSt1() {
    return st1;
}

String GetSt2() {
    return st2;
}

int GetInt1() {
    return int1;
}

float GetFt() {
    return ft;
}

@Override
public void write(DataOutput out) throws IOException {

    Text txt = new Text();
    txt.set(st1);
    txt.write(out);
    txt.set(st2);
    txt.write(out);

    IntWritable intw = new IntWritable();
```

```
        intw.set(int1);
        intw.write(out);

        FloatWritable fw = new FloatWritable();
        fw.set(ft);
        fw.write(out);
    }

    @Override
    public void readFields(DataInput in) throws IOException {

        Text txt = new Text();
        txt.readFields(in);
        st1 = txt.toString();
        txt.readFields(in);
        st2 = txt.toString();

        IntWritable intw = new IntWritable();
        intw.readFields(in);
        int1 = intw.get();

        FloatWritable fw = new FloatWritable();
        fw.readFields(in);
        ft = fw.get();
    }

    public void printFieldTypes() {
        Class myClass = this.getClass();
        Field[] fields = myClass.getDeclaredFields();

        for (int i = 0; i < fields.length; i++) {
            System.out.println(fields[i].getType().getName());
        }
    }
    }
```

3. 编译并为 `PxfExample_CustomWritable` 创建Java类JAR文件。 为您的Hadoop发行版提供一个包含 `hadoop-common.jar` 文件的类路径。 例如， 如果您安装了Hortonworks Data Platform Hadoop客户端：

```
$ javac -classpath /usr/hdp/current/hadoop-client/hadoop-common.ja
$ cd ../../../../../../
$ jar cf pxfex-customwritable.jar com
$ cp pxfex-customwritable.jar /tmp/
```

(您的Hadoop库类路径可能不同)

4. 将 `pxfex-customwritable.jar` 文件复制到MPP 数据库master节点。 例如：

```
$ scp pxfex-customwritable.jar gpadmin@gpmaster:/home/gpadmin
```

5. 登录到您的MPP数据库master节点：

```
$ ssh gpadmin@<gpmaster>
```

6. 将 `pxfex-customwritable.jar` JAR文件复制到用户运行时类库目录中，并记下位置。 例如， 如果 `PXF_CONF=/usr/local/mpp-pxf` ：

```
gpadmin@gpmaster$ cp /home/gpadmin/pxfex-customwritable.jar /usr/l
```

7. 将PXF配置同步到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

8. 如重启PXF中所述，在每个MPP数据库segment主机上重启PXF。

9. 使用PXF `hdfs:SequenceFile` 配置文件创建MPP数据库可写外部表。 在 `DATA-SCHEMA` 自定义选项 <custom-option> 中标识您在上面创建的序列化/反序列化Java类。 创建可写外部表时，使用 `BZip2` 压缩并指定 `BLOCK` 压缩模式。

```
postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_tbl_seqfile (locatio
            LOCATION ('pxf://data/pxf_examples/pxf_seqfile?PROFILE
          FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

注意 'CUSTOM' FORMAT 格式化属性 <formatting-properties> 指定为内置的 pxfwritable_export 格式化程序。

0. 通过直接插入 pxf_tbl_seqfile 表将一些数据写入 pxf_seqfile HDFS目录中。例如：

```
postgres=# INSERT INTO pxf_tbl_seqfile VALUES ( 'Frankfurt', 'Mar'
postgres=# INSERT INTO pxf_tbl_seqfile VALUES ( 'Cleveland', 'Oct'
```

1. 回想一下，MPP 数据库不支持直接查询一个可写外部表。要读取 pxf_seqfile 中的数据，请创建一个引用此HDFS目录的可读外部表：

```
postgres=# CREATE EXTERNAL TABLE read_pxf_tbl_seqfile (location te
            LOCATION ('pxf://data/pxf_examples/pxf_seqfile?PROFILE
          FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

当您通过 hdfs:SequenceFile 配置文件读取HDFS数据是必须指定 DATA-SCHEMA 自定义选项 <custom-option>。您无需提供压缩相关的选项。

2. 查询可读外部表 read_pxf_tbl_seqfile ：

```
gpadmin=# SELECT * FROM read_pxf_tbl_seqfile ORDER BY total_sales;
```

```
 location  | month | number_of_orders | total_sales
-----------+-------+------------------+-------------
 Frankfurt | Mar   |              777 |     3956.98
 Cleveland | Oct   |             3812 |     96645.4
(2 rows)
```

读取记录键

当MPP数据库外部表引用SequenceFile或其他以键-值对存储行的数据格式时，您可以通过使用 `recordkey` 关键字作为字段名称，在MPP查询中访问记录键的值。

`recordkey` 字段类型必须与记录键类型相对应，就像其他字段必须与HDFS数据匹配一样。.

您可以将 `recordkey` 定义为以下任何Hadoop类型：

- BooleanWritable
- ByteWritable
- DoubleWritable
- FloatWritable
- IntWritable
- LongWritable
- Text

如果没有为行定义记录键， MPP数据库将返回处理该行的segment的id。

示例： 使用记录键

创建一个可读外部表，以访问您在 示例： 将二进制数据写入HDFS 创建的可写外部表 `pxf_tbl_seqfile` 的记录键。 本例中将 `recordkey` 定义为 `int8` 类型。

```
postgres=# CREATE EXTERNAL TABLE read_pxf_tbl_seqfile_recordkey(recor
                LOCATION ('pxf://data/pxf_examples/pxf_seqfile?PROFIL
          FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
gpadmin=# SELECT * FROM read_pxf_tbl_seqfile_recordkey;
```

```
 recordkey |  location   | month | number_of_orders | total_sales
-----------+-------------+-------+------------------+-------------
         2 | Frankfurt   | Mar   |              777 |     3956.98
         1 | Cleveland   | Oct   |             3812 |     96645.4
(2 rows)
```

当您将行插入到可写外部表时，您没有定义记录键， 因此 `recordkey` 标识为处理该行的segment。

将多行文本文件读入单个表行

In this topic:

- 前提条件
- 读取多行文本和JSON文件
  - 示例：将HDFS文本文件读入单个表行

您可以使用PXF HDFS连接器以单个表行的形式读取HDFS中的一个或多个多行文本文件。当您想将多个文件读入同一MPP数据库外部表时，例如当每个JSON文件各自包含单独的记录时，这可能很有用。

PXF支持以这种方式仅读取文本和JSON文件。

Note: 如果要使用PXF读取包含多个记录的JSON文件，请参考从HDFS读取JSON数据主题。

前提条件

尝试从HDFS读取文件之前，请确保已满足PXF Hadoop 前提条件。

读取多行文本和JSON文件

您可以将单行和多行文件读取到单个表行中，包括具有嵌入式换行符的文件。如果要读取多个JSON文件，则每个文件必须是完整的记录，并且每个文件必须包含相同的记录类型。

PXF将完整的文件数据读取到单个行和列中。创建外部表以读取多个文件时，必须确保要读取的所有文件都具有相同的类型（文本或JSON）。您还必须指定一个 `text` 或 `json` 列，具体取决于文件类型。

以下语法创建了MPP数据库可读的外部表，该表引用HDFS上的一个或多个文本或JSON文件：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> text|json | LIKE <other_table> )
  LOCATION ('pxf://<path-to-files>?PROFILE=hdfs:text:multi[&SERVER=<s
FORMAT 'CSV');
```

下表描述了此CREATE EXTERNAL TABLE命令中使用的关键字和值。

关键词	值
<path-to-files>	HDFS数据存储中目录或文件的绝对路径。
PROFILE	`PROFILE` 关键字必须指定 `hdfs:text:multi` 。
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
FILE_AS_ROW=true	指示PXF将每个文件读入单个表行的必需选项。
FORMAT	`FORMAT` 必须指定为 `'CSV'` 。

Note: 当您指定 `FILE_AS_ROW=true` 选项时， `hdfs:text:multi` 配置文件不支持其他格式选项。

例如，如果 `/data/pxf_examples/jdir` 标识了包含多个JSON文件的HDFS目录，则以下语句将创建一个MPP数据库外部表，该表引用该目录中的所有文件：

```
CREATE EXTERNAL TABLE pxf_readjfiles(j1 json)
  LOCATION ('pxf://data/pxf_examples/jdir?PROFILE=hdfs:text:multi&FI
FORMAT 'CSV';
```

当您使用 `SELECT` 语句查询 `pxf_readjfiles` 表时，PXF将 `jdir/` 中的每个JSON文件的内容作为外部表中的单独行返回。

读取JSON文件时，可以使用MPP数据库中提供的JSON函数来访问JSON记录中的各个数据字段。例如，如果上面的 `pxf_readjfiles` 外部表读取包含此JSON记录的JSON文件：

```
{
  "root":[
    {
      "record_obj":{
        "created_at":"MonSep3004:04:53+00002013",
        "id_str":"384529256681725952",
        "user":{
          "id":31424214,
          "location":"COLUMBUS"
        },
        "coordinates":null
      }
    }
  ]
}
```

您可以使用 `json_array_elements()` 函数从表行中提取特定的JSON字段。例如，以下命令显示 `user->id` 字段：

```
SELECT json_array_elements(j1->'root')->'record_obj'->'user'->'id'
  AS userid FROM pxf_readjfiles;

  userid
----------
 31424214
(1 rows)
```

有关使用MPP数据库操作JSON数据的特定信息，请参考使用JSON数据。

示例：将HDFS文本文件读入单个表行

执行以下过程在HDFS目录中创建3个示例文本文件，并使用PXF `hdfs:text:multi` 配置文件和默认的PXF服务器在单个外部表查询中读取所有这些文本文件。

1. 为文本文件创建一个HDFS目录。例如：

```
$ hdfs dfs -mkdir -p /data/pxf_examples/tdir
```

2. 创建一个名为 `file1.txt` 的文本数据文件：

```
$ echo 'text file with only one line' > /tmp/file1.txt
```

3. 创建另一个名为 `file2.txt` 的文本数据文件：

```
$ echo 'Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67' > /tmp/file2.txt
```

这个文件有多行。

4. 创建一个名为 `/tmp/file3.txt` 的第三个文本文件：

```
$ echo '"4627 Star Rd.
San Francisco, CA  94107":Sept:2017
"113 Moon St.
San Diego, CA  92093":Jan:2018
"51 Belt Ct.
Denver, CO  90123":Dec:2016
"93114 Radial Rd.
Chicago, IL  60605":Jul:2017
"7301 Brookview Ave.
Columbus, OH  43213":Dec:2018' > /tmp/file3.txt
```

该文件包括嵌入式换行符。

5. 保存文件并退出编辑器。

6. 将文本文件复制到HDFS：

```
$ hdfs dfs -put /tmp/file1.txt /data/pxf_examples/tdir
$ hdfs dfs -put /tmp/file2.txt /data/pxf_examples/tdir
$ hdfs dfs -put /tmp/file3.txt /data/pxf_examples/tdir
```

7. 登录到MPP数据库系统并启动 `psql` 子系统。

8. 使用 `hdfs:text:multi` 配置文件来创建引用 `tdir` HDFS目录的外部表。例如：

```
CREATE EXTERNAL TABLE pxf_readfileasrow(c1 text)
  LOCATION ('pxf://data/pxf_examples/tdir?PROFILE=hdfs:text:multi&
FORMAT 'CSV';
```

9. 打开扩展显示并查询 `pxf_readfileasrow` 表：

```
postgres=# \x on
postgres=# SELECT * FROM pxf_readfileasrow;
```

```
-[ RECORD 1 ]--------------------------
c1 | Prague,Jan,101,4875.33
   | Rome,Mar,87,1557.39
   | Bangalore,May,317,8936.99
   | Beijing,Jul,411,11600.67
-[ RECORD 2 ]--------------------------
c1 | text file with only one line
-[ RECORD 3 ]--------------------------
c1 | "4627 Star Rd.
   | San Francisco, CA  94107":Sept:2017
   | "113 Moon St.
   | San Diego, CA  92093":Jan:2018
   | "51 Belt Ct.
   | Denver, CO  90123":Dec:2016
   | "93114 Radial Rd.
   | Chicago, IL  60605":Jul:2017
   | "7301 Brookview Ave.
   | Columbus, OH  43213":Dec:2018
```

读取Hive表数据

In this topic:

- 先决条件
- Hive数据格式
- 数据类型映射
- 样本数据集
- Hive命令行
- 查询外部Hive数据
- 访问TextFile格式的Hive表
- 访问RCFile格式的Hive表
- 访问ORC格式的Hive表
- 访问Parquet格式的Hive表
- 处理复杂数据类型
- 分区过滤下推
  - 示例: 使用Hive配置文件访问同构分区的数据
  - 示例: 使用Hive配置文件访问异构分区的数据
- 使用PXF访问Hive默认分区

Apache Hive是一个分布式数据仓库基础架构。 Hive有助于管理支持多种数据格式的大型数据集，包括逗号分隔值(.csv)、TextFile、RCFile、 ORC、和Parquet。

PXF Hive连接器读取Hive表中存储的数据。本节描述如何使用PXF Hive连接器。

先决条件

在使用PXF处理Hive表数据之前，请确保您已满足PXF Hadoop 先决条件。

如果您打算将PXF过滤器下推与Hive整数类型一起使用，请确保配置参数 `hive.metastore.integral.jdo.pushdown` 存在，并且在两个配置文件中的 `hive-site.xml` 中均设置为 `true` 。Hadoop集群和 `$PXF_CONF/servers/default/hive-site.xml` 。 有关更多信息，请参考更新Hadoop配置。

Hive数据格式

PXF Hive连接器支持多种数据格式，并定义了以下用于访问这些格式的配置文件：

文件格式	描述	配置文件
TextFile	平面文件，其数据以逗号，制表符或空格分隔的值格式或JSON表示法。	Hive, HiveText
SequenceFile	平面文件，由二进制键/值对组成。	Hive
RCFile	列式记录文件，由二进制键/值对组成; 高行压缩率。	Hive, HiveRC
ORC	优化了的行列数据文件，具有stripe、 footer 、和postscript部分; 减少数据大小.	Hive, HiveORC, HiveVectorizedORC
Parquet	压缩的列式数据。	Hive

注意: `Hive` 配置文件支持所有文件存储格式。 它将对底层文件类型使用最佳的 `Hive*` 配置文件。

数据类型映射

PXF Hive连接器支持基础和复杂数据类型。

基础数据类型

要在MPP数据库中展示Hive数据，请将使用基础数据类型的数据值映射到相同类型的MPP数据列。

下表总结了Hive基本数据类型的外部映射规则。

Hive 数据类型	MPP 数据类型
boolean	bool
int	int4
smallint	int2
tinyint	int2
bigint	int8
float	float4
double	float8
string	text
binary	bytea
timestamp	timestamp

注意: `HiveVectorizedORC` 配置文件不支持 timestamp 数据类型。

复杂数据类型

Hive支持的数据类型，包括数组(array), 结构(struct), 映射(map), 以及混合类型。PXF 将以上复杂数据类型映射为 `text` 。您可以创建MPP数据库函数或应用程序代码来提取这些复杂数据类型的子部分。

本章稍后提供通过 `Hive` 和 `HiveORC` 配置文件使用复杂数据类型的示例。

注意: `HiveVectorizedORC` 配置文件不支持复杂类型。

样本数据集

本主题中介绍的示例在公共数据集上运行。 这个简单的数据集为零售业务建模，并包含具有以下名称和数据类型的字段：

字段名	数据类型
location	text
month	text
number_of_orders	integer
total_sales	double

准备样本数据集以备用：

1. 首先，创建一个文本文件：

```
$ vi /tmp/pxf_hive_datafile.txt
```

2. 将以下数据添加到 `pxf_hive_datafile.txt` ; 注意使用逗号 `,` 分隔四个字段的值：

```
Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67
San Francisco,Sept,156,6846.34
Paris,Nov,159,7134.56
San Francisco,Jan,113,5397.89
Prague,Dec,333,9894.77
Bangalore,Jul,271,8320.55
Beijing,Dec,100,4248.41
```

记住 `pxf_hive_datafile.txt` 的路径;您将在后续的练习中使用它。

Hive命令行

Hive命令行是类似 `psql` 的子系统。启动Hive命令行：

```
$ HADOOP_USER_NAME=hdfs hive
```

默认的Hive数据库名为 `default` 。

示例：创建Hive表

创建一个Hive表以展现示例数据集。

1. 在 `default` 数据库中创建一个名为 `sales_info` 的Hive表：

```
hive> CREATE TABLE sales_info (location string, month string,
        number_of_orders int, total_sales double)
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
        STORED AS textfile;
```

注意：

- `STORED AS textfile` 子句指示Hive以 Textfile (默认) 格式创建表。 Hive Textfile格式支持逗号、制表符和空格分隔的值， 以及用JSON表示的数据。
- `DELIMITED FIELDS TERMINATED BY` 子句定义数据记录(行)中的字段分隔符。 `sales_info` 表字段分隔符是逗号(`,`)。

2. 将 `pxf_hive_datafile.txt` 示例数据文件加载到您刚刚创建的 `sales_info` 表中：

```
hive> LOAD DATA LOCAL INPATH '/tmp/pxf_hive_datafile.txt'
        INTO TABLE sales_info;
```

在本章稍后的例子中，您将通过PXF直接访问 `sales_info` Hive 表。 另外您还将在其他Hive文件格式类型的表中插入 `sales_info` 数据，并使用PXF直接访问这些数据。

3. 在 `sales_info` 上执行查询以验证是否成功加载了数据：

```
hive> SELECT * FROM sales_info;
```

确定Hive表的HDFS位置

如果需要指定Hive表的HDFS文件位置，请使用其HDFS文件路径引用它。 您可以通过 `DESCRIBE` 命令确定Hive表在HDFS中的位置。例如：

```
hive> DESCRIBE EXTENDED sales_info;
Detailed Table Information
...
location:hdfs://<namenode>:<port>/apps/hive/warehouse/sales_info
...
```

查询外部Hive数据

您可以创建一个MPP数据外部表以访问Hive表中的数据。 如前所述，PXF Hive连接器定义了特定的配置文件以支持不同的文件格式。 这些配置文件分别为 `Hive` 、 `HiveText` 、 `HiveRC` 、 `HiveORC` 和 `HiveVectorizedORC` 。

`HiveText` 和 `HiveRC` 配置文件分别针对文本和RCFile格式进行了优化。 `HiveORC` 和 `HiveVectorizedORC` 配置文件针对ORC文件格式进行了优化。 `Hive` 配置文件为所有文件存储类型进行了优化。 当Hive表底层由不同文件格式的多个分区组成时，您可以使用 `Hive` 配置文件。

使用以下语法创建引用Hive表的MPP数据库外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<hive-db-name>.<hive-table-name>
    ?PROFILE=Hive|HiveText|HiveRC|HiveORC|HiveVectorizedORC[&SERVER=<
FORMAT 'CUSTOM|TEXT' (FORMATTER='pxfwritable_import' | delimiter='<de
```

CREATE EXTERNAL TABLE 命令中Hive连接器使用的特定关键字和值见下表中描述。

关键字	值
<hive-db-name>	Hive数据库的名称。 如果省略，默认为名为 `default` 的Hive数据库
<hive-table-name>	Hive表的名称
PROFILE	`PROFILE` 关键字的值必须指定为 `Hive` , `HiveText` , `HiveRC` , `HiveORC` , 或 `HiveVectorizedORC` 之一
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。

关键字	值
FORMAT (`Hive` , `HiveORC` , 和 `HiveVectorizedORC` 配置文件)	`FORMAT` 子句必须指定为 `'CUSTOM'` 。 `CUSTOM` 格式需要内置的 `pxfwritable_import` `formatter`
FORMAT (`HiveText` 和 `HiveRC` 配置文件)	`FORMAT` 子句必须指定为 `TEXT` 。 在 `delimiter ='<delim>'` 格式选项中指定单个ascii字符字段定界符。

访问TextFile格式的Hive表

您可以使用 `Hive` 和 `HiveText` 配置文件来访问以TextFile格式存储的Hive表数据。

示例: 使用Hive配置文件

使用 `Hive` 配置文件创建一个可读的MPP数据库外部表，该表引用此前您创建的Hive文本格式表 `sales_info` 。

1. 创建外部表:

```
postgres=# CREATE EXTERNAL TABLE salesinfo_hiveprofile(location te
           LOCATION ('pxf://default.sales_info?PROFILE=Hive')
         FORMAT 'custom' (FORMATTER='pxfwritable_import');
```

2. 查询表:

```
postgres=# SELECT * FROM salesinfo_hiveprofile;
```

```
   location    | month | num_orders | total_sales
---------------+-------+------------+-------------
 Prague        | Jan   |        101 |     4875.33
 Rome          | Mar   |         87 |     1557.39
 Bangalore     | May   |        317 |     8936.99
 ...
```

示例: 使用HiveText配置文件

使用 `HiveText` 配置文件创建一个可读的MPP数据库外部表，该表引用此前您创建的Hive文本格式表 `sales_info` 。

1. 创建外部表:

```
postgres=# CREATE EXTERNAL TABLE salesinfo_hivetextprofile(locatic
             LOCATION ('pxf://default.sales_info?PROFILE=HiveText'
           FORMAT 'TEXT' (delimiter=E',');
```

注意， `FORMAT` 子句 `delimiter` 值指定为单个ASCII逗号字符 `','` 。 `E` 转义字符。

2. 查询外部表:

```
postgres=# SELECT * FROM salesinfo_hivetextprofile WHERE location=
```

```
 location | month | num_orders | total_sales
----------+-------+------------+-------------
 Beijing  | Jul   |        411 |    11600.67
 Beijing  | Dec   |        100 |     4248.41
(2 rows)
```

访问RCFile格式的Hive表

RCFile Hive表格式用于行列格式的数据。PXF `HiveRC` 配置文件提供对RCFile数据的访问。

示例: 使用HiveRC配置文件

使用 `HiveRC` 配置文件在Hive中查询RCFile格式的数据。

1. 启动 `hive` 命令行并创建一个以RCFile格式存储的Hive表:

```
$ HADOOP_USER_NAME=hdfs hive
```

```
hive> CREATE TABLE sales_info_rcfile (location string, month strin
        number_of_orders int, total_sales double)
      ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
      STORED AS rcfile;
```

2. 将 `sales_info` 表的数据写入到 `sales_info_rcfile`:

```
hive> INSERT INTO TABLE sales_info_rcfile SELECT * FROM sales_info
```

   样本数据集的副本现在以RCFile格式存储在Hive `sales_info_rcfile` 表中。

3. 查询 `sales_info_rcfile` Hive 表以验证数据是否正确加载:

```
hive> SELECT * FROM sales_info_rcfile;
```

4. 使用 PXF `HiveRC` 配置文件创建MPP数据库可读外部表，该表引用此前步骤您创建的Hive `sales_info_rcfile` 表。例如:

```
postgres=# CREATE EXTERNAL TABLE salesinfo_hivercprofile(location
             LOCATION ('pxf://default.sales_info_rcfile?PROFILE=Hi
           FORMAT 'TEXT' (delimiter=E',');
```

5. 查询外部表:

```
postgres=# SELECT location, total_sales FROM salesinfo_hivercprofi
```

```
   location    | total_sales
---------------+-------------
 Prague        |     4875.33
 Rome          |     1557.39
 Bangalore     |     8936.99
 Beijing       |    11600.67
 ...
```

访问ORC格式的Hive表

优化行列(ORC) 文件格式是一种列文件格式，提供了一种高效的方式来存储和访问HDFS数据。 ORC 格式在压缩和性能方面都优于文本和RCFile格式。 PXF支持ORC 1.2.1版本。

ORC 具有类型感知能力，且专门针对Hadoop工作负载而设计。 ORC文件存储文件中数据的类型和编码信息。一组行数据(也称为stripe)中的所有列一起以ORC格式文件存储于磁盘上。ORC 格式类型的列性质允许进行读取投影，从而有助于避免在查询期间访问不必要的列。

ORC 还支持在file, stripe, row 级别使用内置索引进行谓词下推，从而将过滤操作移至数据加载阶段。

有关ORC文件格式的详细信息，请参考Apache orc 和 Apache Hive

LanguageManual ORC 网站。

支持ORC文件格式的配置文件

选择支持ORC的配置文件时，请考虑以下事情:

- `HiveORC` 配置文件:
  - 一次读取一行
  - 支持列投影
  - 支持复杂类型。 您可以访问由数据(array), 映射(map), 结构(struct), 和联合数据类型组成的Hive表。 PXF 将这些复杂类型序列化为 `text` 。
- `HiveVectorizedORC` 配置文件:
  - 一次最多读取1024行
  - 不支持列投影
  - 不支持复杂类型或 timestamp 数据类型

示例: 使用HiveORC配置文件

在接下来的例子中，您将创建以ORC格式存储的Hive表，并使用 `HiveORC` 配置文件查询此Hive表。

1. 用ORC文件格式创建Hive表:

```
$ HADOOP_USER_NAME=hdfs hive
```

```
hive> CREATE TABLE sales_info_ORC (location string, month string,
        number_of_orders int, total_sales double)
      STORED AS ORC;
```

2. 将 `sales_info` 表的数据写入到 `sales_info_ORC`：

```
hive> INSERT INTO TABLE sales_info_ORC SELECT * FROM sales_info;
```

样本数据集的副本现在以ORC格式存储在 `sales_info_ORC` 中。

3. 在 `sales_info_ORC` 上执行Hive查询以验证数据是否正确加载：

```
hive> SELECT * FROM sales_info_ORC;
```

4. 启动 `psql` 子系统并打开计时：

```
$ psql -d postgres
```

```
postgres=> \timing
Timing is on.
```

5. 使用PXF `HiveORC` 配置文件创建MPP数据库外部表，该表引用此前您在步骤 1 创建的名为 `sales_info_ORC` 的Hive表。 `FORMAT` 子句必须指定为 `'CUSTOM'`。 `HiveORC` `CUSTOM` 格式仅支持内置的 `'pxfwritable_import'` `formatter`。

```
postgres=> CREATE EXTERNAL TABLE salesinfo_hiveORCprofile(location
             LOCATION ('pxf://default.sales_info_ORC?PROFILE=HiveC
             FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

6. 查询外部表：

```
postgres=> SELECT * FROM salesinfo_hiveORCprofile;
```

```
   location    | month | number_of_orders | total_sales
---------------+-------+------------------+-------------
 Prague        | Jan   |              101 |     4875.33
 Rome          | Mar   |               87 |     1557.39
 Bangalore     | May   |              317 |     8936.99
 ...

Time: 425.416 ms
```

示例: 使用HiveVectorizedORC配置文件

在以下示例中，您将使用 `HiveVectorizedORC` 配置文件查询您在此前示例中创建的 `sales_info_ORC` Hive表。

1. 启动 `psql` 子系统:

```
$ psql -d postgres
```

2. 使用PXF `HiveVectorizedORC` 配置文件创建MPP数据库外部表，该表引用您在此前示例步骤 1 中创建的名为 `sales_info_ORC` 的Hive表。 `FORMAT` 子句必须指定为 `'CUSTOM'`。 `HiveVectorizedORC` `CUSTOM` 格式仅支持内置的 `'pxfwritable_import'` `formatter`。

```
postgres=> CREATE EXTERNAL TABLE salesinfo_hiveVectORC(location te
           LOCATION ('pxf://default.sales_info_ORC?PROFILE=HiveV
           FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

3. 查询外部表:

```
postgres=> SELECT * FROM salesinfo_hiveVectORC;
```

```
   location    | month | number_of_orders | total_sales
---------------+-------+------------------+-------------
 Prague        | Jan   |              101 |     4875.33
 Rome          | Mar   |               87 |     1557.39
 Bangalore     | May   |              317 |     8936.99
 ...

Time: 425.416 ms
```

访问Parquet格式的Hive表

PXF `Hive` 配置文件支持使用Parquet存储格式的非分区和分区Hive表。 使用等效的MPP数据库数据类型映射Hive表列。 例如, 如果在 `default` 模式中使用以下命令创建一个Hive表:

```
hive> CREATE TABLE hive_parquet_table (location string, month string,
            number_of_orders int, total_sales double)
        STORED AS parquet;
```

定义MPP数据库外部表:

```
postgres=# CREATE EXTERNAL TABLE pxf_parquet_table (location text, mo
    LOCATION ('pxf://default.hive_parquet_table?profile=Hive')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

查询外部表:

```
postgres=# SELECT month, number_of_orders FROM pxf_parquet_table;
```

处理复杂数据类型

示例: 使用Hive配置文件处理复杂数据类型

本示例中使用 `Hive` 配置文件以及数组和映射复杂数据类型，特别是整数数组和字符串键/值对。

此示例中的数据模式包含具有以下名称和数据类型的字段:

字段名	数据类型
index	int
name	string
intarray	整数数组
propmap	字符串键/值对映射

当您在Hive表中指定数组字段，必须为集合中的每个项目标识终止符。 相似的，您还必须为映射键指定终止符。

1. 创建一个文本文件，从中加载数据:

    ```
    $ vi /tmp/pxf_hive_complex.txt
    ```

2. 将以下文本添加到 `pxf_hive_complex.txt` 。 此数据使用 逗号 `,` 分隔字段值，百分号 `%` 分隔集合项， 并使用 `:` 终止映射键值:

```
3,Prague,1%2%3,zone:euro%status:up
89,Rome,4%5%6,zone:euro
400,Bangalore,7%8%9,zone:apac%status:pending
183,Beijing,0%1%2,zone:apac
94,Sacramento,3%4%5,zone:noam%status:down
101,Paris,6%7%8,zone:euro%status:up
56,Frankfurt,9%0%1,zone:euro
202,Jakarta,2%3%4,zone:apac%status:up
313,Sydney,5%6%7,zone:apac%status:pending
76,Atlanta,8%9%0,zone:noam%status:down
```

3. 创建一个Hive表来展示此数据:

```
$ HADOOP_USER_NAME=hdfs hive
```

```
hive> CREATE TABLE table_complextypes( index int, name string, int
          ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
          COLLECTION ITEMS TERMINATED BY '%'
          MAP KEYS TERMINATED BY ':'
          STORED AS TEXTFILE;
```

注意:

- `FIELDS TERMINATED BY` 将逗号标识为字段分隔符
- `COLLECTION ITEMS TERMINATED BY` 子句将百分号作为集合项(数组项, 映射键/值对)的终止符
- `MAP KEYS TERMINATED BY` 将冒号标识为映射键的终止符

4. 将 `pxf_hive_complex.txt` 示例数据文件加载到您刚刚创建的 `table_complextypes` 表中:

```
hive> LOAD DATA LOCAL INPATH '/tmp/pxf_hive_complex.txt' INTO TABL
```

5. 在Hive表 `table_complextypes` 上执行查询以验证数据是否已成功加载:

```
hive> SELECT * FROM table_complextypes;
```

```
3   Prague  [1,2,3] {"zone":"euro","status":"up"}
89  Rome    [4,5,6] {"zone":"euro"}
400 Bangalore   [7,8,9] {"zone":"apac","status":"pending"}
...
```

6. 使用 `Hive` 配置文件创建可读MPP数据库外部表，该表引用名为 `table_complextypes` 的Hive表:

```
postgres=# CREATE EXTERNAL TABLE complextypes_hiveprofile(index ir
             LOCATION ('pxf://table_complextypes?PROFILE=Hive')
           FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

注意整数数组和映射复杂类型已映射为MPP数据库文本数据类型

7. 查询外部表:

```
postgres=# SELECT * FROM complextypes_hiveprofile;
```

```
 index |    name    | intarray |               propmap
-------+------------+----------+-------------------------------------
     3 | Prague     | [1,2,3]  | {"zone":"euro","status":"up"}
    89 | Rome       | [4,5,6]  | {"zone":"euro"}
   400 | Bangalore  | [7,8,9]  | {"zone":"apac","status":"pending"
   183 | Beijing    | [0,1,2]  | {"zone":"apac"}
    94 | Sacramento | [3,4,5]  | {"zone":"noam","status":"down"}
   101 | Paris      | [6,7,8]  | {"zone":"euro","status":"up"}
    56 | Frankfurt  | [9,0,1]  | {"zone":"euro"}
   202 | Jakarta    | [2,3,4]  | {"zone":"apac","status":"up"}
   313 | Sydney     | [5,6,7]  | {"zone":"apac","status":"pending"
    76 | Atlanta    | [8,9,0]  | {"zone":"noam","status":"down"}
(10 rows)
```

`intarray` 和 `propmap` 都被序列化为文本字符串

示例: 使用HiveORC配置文件处理复杂数据类型

在以下示例中，您将创建和填充以ORC格式存储的Hive表。 您将使用 `HiveORC` 配置文件查询此Hive表中的复杂数据类型。

1. 用ORC存储格式创建Hive表:

```
$ HADOOP_USER_NAME=hdfs hive
```

```
hive> CREATE TABLE table_complextypes_ORC( index int, name string,
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
        COLLECTION ITEMS TERMINATED BY '%'
        MAP KEYS TERMINATED BY ':'
      STORED AS ORC;
```

2. 将您在此前示例中创建的 `table_complextypes` 表的数据写入到 `table_complextypes_ORC` 表中:

```
hive> INSERT INTO TABLE table_complextypes_ORC SELECT * FROM table
```

A copy of the sample data set is now stored in ORC format in `table_complextypes_ORC` .

3. 在 `table_complextypes_ORC` 上执行查询以验证数据是否成功加载:

```
hive> SELECT * FROM table_complextypes_ORC;
```

```
OK
3       Prague        [1,2,3]    {"zone":"euro","status":"up"}
89      Rome          [4,5,6]    {"zone":"euro"}
400     Bangalore     [7,8,9]    {"zone":"apac","status":"pending"}
...
```

4. 启动 `psql` 子系统:

```
$ psql -d postgres
```

5. 使用 PXF `HiveORC` 配置文件创建可读MPP数据库外部表，该表引用您在步骤 1 中创建的名为 `table_complextypes_ORC` 的Hive表。 `FORMAT` 子句必须指定为 `'CUSTOM'`。 `HiveORC` `CUSTOM` 格式仅支持内置的 `'pxfwritable_import'` `formatter`。

```
postgres=> CREATE EXTERNAL TABLE complextypes_hiveorc(index int, r
           LOCATION ('pxf://default.table_complextypes_ORC?PROFILE
             FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

注意整数数组和映射复杂类型已映射为MPP数据库文本数据类型

6. 查询外部表:

```
postgres=> SELECT * FROM complextypes_hiveorc;
```

```
 index |    name     | intarray |              propmap
-------+-------------+----------+-----------------------------------
     3 | Prague      | [1,2,3]  | {"zone":"euro","status":"up"}
    89 | Rome        | [4,5,6]  | {"zone":"euro"}
   400 | Bangalore   | [7,8,9]  | {"zone":"apac","status":"pending"
 ...
```

`intarray` 和 `propmap` 都被序列化为文本字符串

分区过滤下推

PXF Hive连接器支持Hive分区修剪和Hive分区目录结构。这样可以在包含Hive表的选定HDFS文件上排除分区。要使用分区筛选功能来减少网络流量和I/O，请使用 `WHERE` 子句在PXF外部表上运行查询，该子句引用已分区的Hive表中的特定分区列。

下面介绍了对Hive字符串和整数类型的PXF Hive连接器分区过滤支持：

- 字符串类型支持关系运算符 `=` ， `<<` ， `<=` ， `>` ， `>=` 和 `<>` 。
- 整数类型支持关系运算符 `=` 和 `<>` (要对Hive整数类型使用分区过滤，必须按照前提条件中所述更新Hive配置)。
- 与上述关系运算符一起使用时，支持逻辑运算符 `AND` 和 `OR` 。
- 不支持 `LIKE` 字符串运算符。

要利用PXF分区过滤下推功能，Hive和PXF分区字段名称必须相同。否则，PXF将忽略分区过滤，并且过滤将在MPP数据库端执行，从而影响性能。

PXF Hive连接器仅在分区列上过滤，而不在其他表属性上过滤。此外，仅以上确定的那些数据类型和运算符支持过滤器下推。

默认情况下，PXF过滤器下推处于启用状态。您可以按照如下所述配置PXF过滤器下推关于过滤器下推。

示例: 使用Hive配置文件访问同构分区的数据

在此示例中，您将使用 `Hive` 配置文件来查询名为 `sales_part` 的Hive表，该表使用 `delivery_state` 和 `delivery_city` 字段分区。 然后，您创建一个MPP数据库外

部表以查询 `sales_part` 。 该过程包括一些演示过滤器下推的特定示例。

1. 创建一个名为 `sales_part` 的Hive表，包含两个分区字段， `delivery_state` 和 `delivery_city` ：

```
hive> CREATE TABLE sales_part (name string, type string, supplier_
        PARTITIONED BY (delivery_state string, delivery_city strin
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

2. 加载数据至Hive表并添加一些分区：

```
hive> INSERT INTO TABLE sales_part
        PARTITION(delivery_state = 'CALIFORNIA', delivery_city = '
        VALUES ('block', 'widget', 33, 15.17);
hive> INSERT INTO TABLE sales_part
        PARTITION(delivery_state = 'CALIFORNIA', delivery_city = '
        VALUES ('cube', 'widget', 11, 1.17);
hive> INSERT INTO TABLE sales_part
        PARTITION(delivery_state = 'NEVADA', delivery_city = 'Reno
        VALUES ('dowel', 'widget', 51, 31.82);
hive> INSERT INTO TABLE sales_part
        PARTITION(delivery_state = 'NEVADA', delivery_city = 'Las
        VALUES ('px49', 'pipe', 52, 99.82);
```

3. 查询 `sales_part` 表：

```
hive> SELECT * FROM sales_part;
```

Hive分区表上的 `SELECT *` 语句显示记录末尾的分区字段。

4. 检查 `sales_part` 表Hive/HDFS的目录结构：

```
$ sudo -u hdfs hdfs dfs -ls -R /apps/hive/warehouse/sales_part
/apps/hive/warehouse/sales_part/delivery_state=CALIFORNIA/delivery
/apps/hive/warehouse/sales_part/delivery_state=CALIFORNIA/delivery
/apps/hive/warehouse/sales_part/delivery_state=NEVADA/delivery_cit
/apps/hive/warehouse/sales_part/delivery_state=NEVADA/delivery_cit
```

5. 创建一个PXF外部表来读取Hive分区表 `sales_part`。 要利用分区过滤器的下推功能，请在 `CREATE EXTERNAL TABLE` 属性列表的末尾定义与Hive分区字段相同的字段。

```
$ psql -d postgres
```

```
postgres=# CREATE EXTERNAL TABLE pxf_sales_part(
             item_name TEXT, item_type TEXT,
             supplier_key INTEGER, item_price DOUBLE PRECISION,
             delivery_state TEXT, delivery_city TEXT)
           LOCATION ('pxf://sales_part?Profile=Hive')
           FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

6. 查询外部表:

```
postgres=# SELECT * FROM pxf_sales_part;
```

7. 在 `pxf_sales_part` 上执行另一个查询(不下推)，以此返回 `delivery_city` 等于 `Sacramento` 和 `item_name` 等于 `cube` 的记录：

```
postgres=# SELECT * FROM pxf_sales_part WHERE delivery_city = 'Sac
```

这个查询过滤了 `delivery_city` 分区 `Sacramento`。 由于 `item_name` 字段不是分区列，因此该列上的过滤不会被下推。 在 `Sacramento` 分区传输完成后，在MPP数据库端执行此过滤操作。

8. 查询(使用过滤下推)所有 `delivery_state` 等于 `CALIFORNIA` 的记录:

```
postgres=# SET gp_external_enable_filter_pushdown=on;
postgres=# SELECT * FROM pxf_sales_part WHERE delivery_state = 'CA
```

该查询读取位于 `CALIFORNIA` `delivery_state` 分区中的所有数据，无论城市是哪个。

示例: 使用Hive配置文件访问异构分区的数据

您可以使用PXF `Hive` 配置文件访问任何Hive文件存储类型。 使用 `Hive` 配置文件，您可以在单个Hive表中访问分区以不同文件格式存储的异构格式数据。

本示例中，您将创建一个分区的Hive外部表。该表由之前示例创建的 `sales_info` (文本格式) 和 `sales_info_rcfile` (RC 格式) Hive表相关的HDFS数据文件组成。 您将按照年份对数据进行分区，将 `sales_info` 的数据分配给2013年， 而 `sales_info_rcfile` 的数据分配给2016年(这里忽略表包含相同数据的事实)。 然后，您将使用PXF `Hive` 配置文件查询这个分区的Hive外部表。

1. 创建一个名为 `hive_multiformpart` 的Hive外部表，该表以名为 `year` 的字符串字段进行分区：

```
$ HADOOP_USER_NAME=hdfs hive
```

```
hive> CREATE EXTERNAL TABLE hive_multiformpart( location string, m
        PARTITIONED BY( year string )
        ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

2. 描述 `sales_info` 和 `sales_info_rcfile` 表，注意每个表HDFS文件的 `location` 信息：

```
hive> DESCRIBE EXTENDED sales_info;
hive> DESCRIBE EXTENDED sales_info_rcfile;
```

3. 在 `hive_multiformpart` 表中为 `sales_info` 以及 `sales_info_rcfile` 表关联的HDFS文件位置创建分区：

```
hive> ALTER TABLE hive_multiformpart ADD PARTITION (year = '2013')
hive> ALTER TABLE hive_multiformpart ADD PARTITION (year = '2016')
```

4. 明确的标识与 `sales_info_rcfile` 表关联的分区的文件格式：

```
hive> ALTER TABLE hive_multiformpart PARTITION (year='2016') SET F
```

无需指定与 `sales_info` 表相关联分区的文件格式，因为 `TEXTFILE` 格式是默认格式。

5. 查询 `hive_multiformpart` 表：

```
hive> SELECT * from hive_multiformpart;
...
Bangalore   Jul 271 8320.55 2016
Beijing Dec 100 4248.41 2016
Prague  Jan 101 4875.33 2013
Rome    Mar 87  1557.39 2013
...
hive> SELECT * from hive_multiformpart WHERE year='2013';
hive> SELECT * from hive_multiformpart WHERE year='2016';
```

6. 查询 `hive_multiformpart` 表的分区定义并退出 `hive`：

```
hive> SHOW PARTITIONS hive_multiformpart;
year=2013
year=2016
hive> quit;
```

7. 启动 `psql` 子系统:

```
$ psql -d postgres
```

8. 使用PXF `Hive` 配置文件创建一个可读的MPP数据库外部表，该表引用此前步骤您在Hive中创建的 `hive_multiformpart` 外部表。

```
postgres=# CREATE EXTERNAL TABLE pxf_multiformpart(location text,
             LOCATION ('pxf://default.hive_multiformpart?PROFILE=H
           FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

9. 查询PXF外部表：

```
postgres=# SELECT * FROM pxf_multiformpart;
```

```
   location    | month | num_orders | total_sales | year
---------------+-------+------------+-------------+--------
 ....
 Prague        | Dec   |        333 |     9894.77 | 2013
 Bangalore     | Jul   |        271 |     8320.55 | 2013
 Beijing       | Dec   |        100 |     4248.41 | 2013
 Prague        | Jan   |        101 |     4875.33 | 2016
 Rome          | Mar   |         87 |     1557.39 | 2016
 Bangalore     | May   |        317 |     8936.99 | 2016
 ....
```

0. 执行第二个查询以计算2013年的订单总数：

```
postgres=# SELECT sum(num_orders) FROM pxf_multiformpart WHERE mon
 sum
-----
 433
```

使用PXF访问Hive默认分区

本主题描述了当Hive使用默认分区时，Hive和PXF在查询结果上的区别。 当Hive启用动态分区后，分区表可能会将数据存储在默认分区中。 当分区列的值与列的定义类型不匹配是时(例如，当任何分区列存在NULL值时)，Hive会创建一个默认分区。 在Hive中，任何在分区列上的过滤查询都会 排除 默认分区中存储的数据。

与Hive类似， PXF将表的分区列表示为附加在表末尾的列。但是，PXF会将默认分区中的任何列值都转换为NULL值。这意味着在同一个Hive查询中，在分区列上包含 `IS NULL` 过滤器的MPP数据库查询可以返回不同结果。

考虑使用以下语句创建的Hive分区表:

```
hive> CREATE TABLE sales (order_id bigint, order_amount float) PARTI
```

该表加载了包含以下内容的五行数据：

```
1.0    1900-01-01
2.2    1994-04-14
3.3    2011-03-31
4.5    NULL
5.0    2013-12-06
```

插入第4行会创建一个Hive默认分区，因为分区列 `xdate` 包含NULL值。

在Hive中，对分区列进行筛选的任何查询都将忽略默认分区中的数据。例如，以下查询不返回任何行：

```
hive> SELECT * FROM sales WHERE xdate IS null;
```

但是，如果将此Hive表映射到MPP数据库中的PXF外部表，则所有默认分区列的值都将被转换为实际的NULL值。在MPP数据库中，对PXF外部表执行相同的查询将返回第4行作为结果，因为过滤器匹配NULL值。

在Hive分区表上执行 `IS NULL` 查询时，请牢记此行为。

读取HBase表数据

In this topic:

- 先决条件
- HBase入门
- HBase Shell
- 查询外部HBase数据
- 数据类型映射
- 列映射
- 行键

Apache HBase是Hadoop上的分布式、版本化的非关系型数据库。

PXF HBase连接器读取存储在HBase表中的数据。HBase连接器支持过滤器下推。

本节介绍如何使用PXF HBase连接器

先决条件

使用HBase表数据前，请确保您已满足：

- 将 `$GPHOME/pxf/lib/pxf-hbase-*.jar` 复制到HBase集群中的每个节点，并且该 PXF JAR文件位于 `$HBASE_CLASSPATH` 中。 PXF HBase 连接器需要此配置才能支持过滤器下推。
- 满足PXF Hadoop 先决条件.

HBase入门

本主题假设您对以下HBase概念有基本的了解：

- HBase 列包含两个组件：列簇和列限定词。 这些组件由冒号 `:` 分隔，<column-family>:<column-qualifier>
- HBase 行由一个行键和一个或多个列值组成。 行键是表行的唯一标识符
- HBase 表是由具有一个或多个列的数据行组成的多维映射。 创建HBase表时，可以指定一组完整的列簇
- HBase 单元由行(列簇, 列限定词, 列值)和时间戳组成。 给定单元格中的列值和时间戳表示该值的版本

有关HBase的详细信息，请参阅Apache HBase Reference Guide。

HBase Shell

HBase shell 是一个类似 `psql` 的子系统。 要启动HBase shell：

```
$ hbase shell
<hbase output>
hbase(main):001:0>
```

HBase默认的命名空间为 `default` 。

示例: 创建一个HBase表

创建一个示例HBase表。

1. 在 `default` 命名空间中创建一个名为 `order_info` 的HBase表。 `order_info` 具有两个列簇: `product` 和 `shipping_info` ：

    ```
    hbase(main):> create 'order_info', 'product', 'shipping_info'
    ```

2. `order_info` `product` 列簇具有名为 `name` 和 `location` 的列标识符。 `shipping_info` 列簇具有名为 `state` 和 `zipcode` 的列标识符。 将一些数据添加到 `order_info` 表中：

    ```
    put 'order_info', '1', 'product:name', 'tennis racquet'
    put 'order_info', '1', 'product:location', 'out of stock'
    put 'order_info', '1', 'shipping_info:state', 'CA'
    put 'order_info', '1', 'shipping_info:zipcode', '12345'
    put 'order_info', '2', 'product:name', 'soccer ball'
    put 'order_info', '2', 'product:location', 'on floor'
    put 'order_info', '2', 'shipping_info:state', 'CO'
    put 'order_info', '2', 'shipping_info:zipcode', '56789'
    put 'order_info', '3', 'product:name', 'snorkel set'
    put 'order_info', '3', 'product:location', 'warehouse'
    put 'order_info', '3', 'shipping_info:state', 'OH'
    put 'order_info', '3', 'shipping_info:zipcode', '34567'
    ```

    在本主题后面的示例中，您将通过PXF直接访问 `orders_info` HBase 表。

3. 显示 `order_info` 表的内容：

```
hbase(main):> scan 'order_info'
ROW     COLUMN+CELL
 1      column=product:location, timestamp=1499074825516, value=ou
 1      column=product:name, timestamp=1499074825491, value=tennis
 1      column=shipping_info:state, timestamp=1499074825531, value
 1      column=shipping_info:zipcode, timestamp=1499074825548, val
 2      column=product:location, timestamp=1499074825573, value=or
 ...
3 row(s) in 0.0400 seconds
```

查询外部HBase数据

PXF HBase 连接器支持一个名为的 `HBase` 的配置文件。

使用以下语法创建引用HBase表的MPP数据库外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<hbase-table-name>?PROFILE=HBase')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

CREATE EXTERNAL TABLE 命令中HBase连接器使用的特定关键字和值见下表描述。

关键字	值
<hbase-table-name>	HBase表的名称
PROFILE	`PROFILE` 关键字必须指定为 `HBase`

关键字	值
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
FORMAT	`FORMAT` 子句必须指定为 `'CUSTOM' (FORMATTER='pxfwritable_import')`

数据类型映射

HBase是基于字节的; 它将所有数据类型存储为字节数组。 要在MPP数据库中表示HBase数据，请为MPP数据库列选择与HBase列标识符值的底层内容匹配的数据类型。

注意: PXF不支持复杂HBase对象

列映射

您可以创建一个引用HBase表定义中全部或部分列标识符的MPP数据库外部表。 PXF支持MPP数据库表列和HBase表列标识符之间的直接或间接映射。

直接映射

当您使用直接映射将MPP数据库外部表的列名映射到HBase标识符时，您可以将列簇限定的HBase标识符名称指定为带引号的值。 PXF HBase连接器在读取表数据时将这些列名按原样传递给HBase。

例如, 创建一个访问以下数据的MPP数据库外部表：

- 列簇 `product` 中的 `name` 标识符

- 列簇 `shipping_info` 中的 `zipcode` 标识符

从此前您在示例: 创建一个HBase表 创建的 `order_info` HBase 表中访问数据，使用以下 `CREATE EXTERNAL TABLE` 语法：

```
CREATE EXTERNAL TABLE orderinfo_hbase ("product:name" varchar, "shipp
    LOCATION ('pxf://order_info?PROFILE=HBase')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

通过查找表间接映射

当您使用间接映射将MPP数据库外部表列名映射到HBase标识符时，您可以在HBase中创建的查找表中指定映射。 查找表将 <column-family>:<column-qualifier> 映射到在创建MPP数据库外部表时指定的列名别名。

您必须将HBase PXF查找表命名为 `pxflookup` 。 并且您必须使用一个名为 `mapping` 的单列簇定义此表。例如:

```
hbase(main):> create 'pxflookup', 'mapping'
```

尽管直接映射既快速又直观， 使用间接映射可以让您为HBase <column-family>:<column-qualifier> 名称创建一个较短的，基于字符的别名。由于以下原因，这可以更好的使HBase标识符与MPP数据库保持一致:

- HBase 标识符可能非常长。 MPP数据库列名大小限制为63各字符。
- HBase 标识符名称可以包含二进制或不可打印的字符。 MPP数据库列名称是基于字符的。

填充 `pxflookup` HBase表时, 向表中添加行, 以便:

- 行键指定为HBase表名

- `mapping` 列簇标识符定义为MPP数据库列名, 值标识为需要创建别名的HBase `<column-family>:<column-qualifier>`

例如，要在 `order_info` 表上使用间接映射，将以下条目添加到 `pxflookup` 表：

```
hbase(main):> put 'pxflookup', 'order_info', 'mapping:pname', 'produc
hbase(main):> put 'pxflookup', 'order_info', 'mapping:zip', 'shipping
```

使用以下 `CREATE EXTERNAL TABLE` 语法创建MPP数据库外部表

```
CREATE EXTERNAL TABLE orderinfo_map (pname varchar, zip int)
    LOCATION ('pxf://order_info?PROFILE=HBase')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

行键

HBase 表行键是表行的唯一标识符。 PXF 以特殊方式处理行键。

要在MPP数据库外部表查询中使用行键，请使用名为 `recordkey` 的PXF保留列定义外部表。 `recordkey` 列名指示PXF返回每一行的HBase表记录键。

使用MPP数据库数据类型 `bytea` 定义 `recordkey`

例如:

```
CREATE EXTERNAL TABLE <table_name> (recordkey bytea, ... )
    LOCATION ('pxf://<hbase_table_name>?PROFILE=HBase')
    FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

在您创建外部表之后，您可以在 `WHERE` 子句中使用 `recordkey` 在一系列键值上过滤HBase表

注意: 若要在 `recordkey` 上启用过滤器下推，请将字段定义为 `text`

使用PXF访问Azure, Google Cloud Storage, Minio和S3对象存储

In this topic:

- 先决条件
- 连接器、数据格式和配置文件
- 示例CREATE EXTERNAL TABLE命令

PXF安装了Azure Blob Storage, Azure Data Lake, Google Cloud Storage, Minio和S3对象存储的连接器

先决条件

在使用PXF处理对象存储数据之前，请确保：

- 您已经配置并初始化了PXF，并且PXF正在每台主机上运行。有关其他信息，请参阅配置PXF
- 您已经配置了计划使用的PXF对象存储连接器。有关说明，请参阅配置Azure、Google云端存储、Minio和S3对象存储的连接器。
- MPP数据库segment主机与外部对象存储系统之间的时间是同步的。

连接器、数据格式和配置文件

PXF对象存储连接器提供内置配置文件支持以下数据格式：

- Text
- Avro
- JSON
- Parquet
- AvroSequenceFile
- SequenceFile

与Azure、Google Cloud Storage、Minio和S3的PXF连接器提供了以下配置文件以读取、写入(在许多情况下)这些受支持的数据格式：

数据格式	Azure Blob Storage	Azure Data Lake	Google Cloud Storage	S3 或 Minio
分隔的单行纯文本	wasbs:text	adl:text	gs:text	s3:text
分隔的带引号的换行符文本	wasbs:text:multi	adl:text:multi	gs:text:multi	s3:text:multi
Avro	wasbs:avro	adl:avro	gs:avro	s3:avro
JSON	wasbs:json	adl:json	gs:json	s3:json
Parquet	wasbs:parquet	adl:parquet	gs:parquet	s3:parquet
AvroSequenceFile	wasbs:AvroSequenceFile	adl:AvroSequenceFile	gs:AvroSequenceFile	s3:AvroSequenceFile
SequenceFile	wasbs:SequenceFile	adl:SequenceFile	gs:SequenceFile	s3:SequenceFile

在 `CREATE EXTERNAL TABLE` 命令中指定 `pxf` 协议以创建引用特定对象存储中的文件或目录的MPP数据库外部表时，可以提供配置文件名称。

示例CREATE EXTERNAL TABLE命令

以下命令创建了一个引用S3上的文本文件的外部表。指定名为 `s3:text` 的配置文件和名为 `s3srvcfg` 的服务配置：

```
CREATE EXTERNAL TABLE pxf_s3_text(location text, month text, num_orders int, total_sale
  LOCATION ('pxf://S3_BUCKET/pxf_examples/pxf_s3_simple.txt?PROFILE=s3:text&SERVE
FORMAT 'TEXT' (delimiter=E',');
```

以下命令创建了一个引用Azure Blob Storage上文本文件的外部表。 指定名为 `wasbs:text` 的配置文件和名为 `wasbssrvcfg` 的服务配置。 您将提供Azure Blob Storage 容器标识和您的 Azure Blob Storage 账户。

```
CREATE EXTERNAL TABLE pxf_wasbs_text(location text, month text, num_orders int, total_
  LOCATION ('pxf://AZURE_CONTAINER@YOUR_AZURE_BLOB_STORAGE_ACCOUNT_NA
FORMAT 'TEXT';
```

以下命令创建了一个引用Azure Data Lake上文本文件的外部表。 指定名为 `adl:text` 的配置文件和名为 `adlsrvcfg` 的服务配置。 您将提供您的Azure Data Lake 账户。

```
CREATE EXTERNAL TABLE pxf_adl_text(location text, month text, num_orders int, total_sale
  LOCATION ('pxf://YOUR_ADL_ACCOUNT_NAME.azuredatalakestore.net/path/to/file?PROFI
FORMAT 'TEXT';
```

以下命令创建了一个引用Google Cloud Storage上JSON文件的外部表。 指定名为 `gs:json` 的配置文件和名为 `gcssrvcfg` 的服务配置：

```
CREATE EXTERNAL TABLE pxf_gsc_json(location text, month text, num_orders int, total_sa
  LOCATION ('pxf://dir/subdir/file.json?PROFILE=gs:json&SERVER=gcssrvcfg')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

关于访问S3对象存储

In this topic:

- 用DDL覆盖S3服务器配置
- 使用Amazon S3 Select服务

PXF安装了与S3对象库的连接器。PXF通过此连接器支持以下其他运行时特性：

- 通过在 `CREATE EXTERNAL TABLE` 命令DDL中提供它们来覆盖服务器配置中指定的S3凭据。
- 使用Amazon S3 Select服务从S3读取某些CSV和Parquet数据。

用DDL覆盖S3服务器配置

如果您访问兼容S3的对象库，可以通过 `CREATE EXTERNAL TABLE` `LOCATION` 子句中的以下自定义选项直接指定S3访问ID和密钥，从而覆盖S3服务配置：

自定义选项	值描述
accesskey	AWS账户访问密钥ID
secretkey	与AWS访问密钥ID关联的密钥

例如：

```
CREATE EXTERNAL TABLE pxf_ext_tbl(name text, orders int)
  LOCATION ('pxf://S3_BUCKET/dir/file.txt?PROFILE=s3:text&SERVER=s3srvcfg&accesske
FORMAT 'TEXT' (delimiter=E',');
```

已这种方式提供的凭据在外部表定义中可见。不要在生产环境中使用这种传递凭据的方法。

PXF目前不支持以这种方式覆盖Azure，Google Cloud Storage和Minio服务器凭据。

有关PXF用于获取MPP数据库用户的配置属性设置的优先级规则的详细信息，请参考配置属性优先级。

使用Amazon S3 Select服务

请参阅使用S3 Select从S3读取CSV和Parquet数据，了解有关PXF如何使用Amazon S3 Select服务读取S3上存储的CSV和Parquet文件的特定信息。

在对象存储中读取和写入文本数据

In this topic:

- 先决条件
- 读取文本数据
  - 示例: 从S3读取文本数据
- 读取含有双引号引起来的换行符的文本数据
  - 示例: 从S3中读取多行文本数据
- 写入文本数据
  - 示例: 将文本数据写入S3

PXF 对象存储连接器支持普通分隔和逗号分隔的值文本数据。 本节介绍如何使用PXF访问对象存储中的文本数据，包括如何创建、查询以及将数据写入引用对象存储中文件的外部表。

注意: 从对象存储中访问文本数据和访问HDFS中的文本数据非常相似。

先决条件

在尝试从对象存储中读取数据或将数据写入对象存储之前，请确保已满足PXF对象存储先决条件

读取文本数据

当您从对象存储中读取每行都是一条记录的分隔的纯文本或.csv数据时，请使用 `<objstore>:text` 配置文件。 PXF 支持以下 `<objstore>` 配置文件前缀：

对象存储	配置前缀
Azure Blob Storage	wasbs
Azure Data Lake	adl

对象存储	配置前缀
Google Cloud Storage	gs
Minio	s3
S3	s3

以下语法创建一个引用对象存储中示例文本文件的MPP数据库可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-file>?PROFILE=<objstore>:text[&SERVER=<serv
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
```

CREATE EXTERNAL TABLE 命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-file>	S3 对象存储中文件或目录的绝对路径
PROFILE=<objstore>:text	`PROFILE` 关键字必须定义为特定的对象存储。 例如: `s3:text`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
FORMAT	当 <path-to-file> 引用分隔的纯文本数据时，请使用 `FORMAT 'TEXT'` 。 当 <path-to-file> 引用逗号分隔的值数据时，请使用 `FORMAT 'CSV'`

关键字	值
delimiter	数据中的分隔符。对于 FORMAT 'CSV' ，默认的<delim_value>是逗号 , 。 当分隔符为转义序列时，在<delim_value>前面加上 E 。示例： (delimiter=E'\t') , (delimiter ':') 。

如果要访问S3对象存储库：

- 您可以通过 CREATE EXTERNAL TABLE 命令中的自定义选项提供S3凭据，如使用DDL覆盖S3服务器配置中所述。
- 如果您正在从S3中读取CSV格式的数据，则可以指示PXF使用S3 Select Amazon服务检索数据。有关用于此目的的PXF自定义选项的更多信息，请参考使用Amazon S3选择服务。

示例: 从S3读取文本数据

执行以下步骤创建示例文本文件，将文件复制到S3, 然后使用 s3:text 配置文件创建两个PXF 外部表来查询数据。

要运行此示例，您必须：

- 在系统上安装了AWS CLI工具
- 知道您的AWS访问ID和密钥
- 拥有对S3存储桶的写权限

1. 在S3上为PXF示例文件创建一个文件夹。 例如, 假设您对名为 BUCKET 的S3存储桶具有写权限：

```
$ aws s3 mb s3://BUCKET/pxf_examples
```

2. 在本地创建一个名为 pxf_s3_simple.txt 的分隔的纯文本数据文件:

```
$ echo 'Prague,Jan,101,4875.33
Rome,Mar,87,1557.39
Bangalore,May,317,8936.99
Beijing,Jul,411,11600.67' > /tmp/pxf_s3_simple.txt
```

请注意使用逗号 `,` 分隔四个字段

3. 将数据文件复制到您在步骤1创建的S3 目录中：

```
$ aws s3 cp /tmp/pxf_s3_simple.txt s3://BUCKET/pxf_examples/
```

4. 验证文件现在位于S3中:

```
$ aws s3 ls s3://BUCKET/pxf_examples/pxf_s3_simple.txt
```

5. 启动 `psql` 子系统:

```
$ psql -d postgres
```

6. 使用 PXF `s3:text` 配置文件创建MPP数据库外部表，该表引用您刚刚创建并添加到S3中的 `pxf_s3_simple.txt` 文件。 例如, 假设您的服务名为 `s3srvcfg` ：

```
postgres=# CREATE EXTERNAL TABLE pxf_s3_textsimple(location text,
            LOCATION ('pxf://BUCKET/pxf_examples/pxf_s3_simple.txt
          FORMAT 'TEXT' (delimiter=E',');
```

7. 查询外部表:

```
postgres=# SELECT * FROM pxf_s3_textsimple;
```

```
   location    | month | num_orders | total_sales
---------------+-------+------------+-------------
 Prague        | Jan   |        101 |     4875.33
 Rome          | Mar   |         87 |     1557.39
 Bangalore     | May   |        317 |     8936.99
 Beijing       | Jul   |        411 |    11600.67
(4 rows)
```

8. 创建另一个引用 `pxf_s3_simple.txt` 的外部表，这次指定 `CSV` `FORMAT`：

```
postgres=# CREATE EXTERNAL TABLE pxf_s3_textsimple_csv(location te
            LOCATION ('pxf://BUCKET/pxf_examples/pxf_s3_simple.txt
          FORMAT 'CSV';
postgres=# SELECT * FROM pxf_s3_textsimple_csv;
```

当您为逗号分隔值数据指定 `FORMAT 'CSV'` 时，不需要提供 `delimiter` 分隔符选项，因为逗号是默认的分隔符。

读取含有双引号引起来的换行符的文本数据

使用 `<objstore>:text:multi` 配置文件读取数据中含有引号引起来的换行符，单行或多行的分隔文本数据。以下语法创建一个引用对象存储中此类文本文件的MPP数据库可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-file>?PROFILE=<objstore>:text:multi[&SERVEI
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-file>	S3数据存储中的目录或文件的绝对路径
PROFILE=<objstore>:text:multi	`PROFILE` 关键字必须指定为特定的对象存储。 例如：`s3:text:multi`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
FORMAT	当<path-to-hdfs-file>引用分隔的纯文本数据时，请使用 `FORMAT` `'TEXT'` 。 当<path-to-hdfs-file>引用逗号分隔值数据时，请使用 `FORMAT` `'CSV'` 。
delimiter	数据中的分隔符。对于 `FORMAT` `'CSV'` ，默认的<delim_value>是逗号 `,` 。 当分隔符为转义序列时，在<delim_value>前面加上 `E` 。示例： `(delimiter=E'\t')` ， `(delimiter ':')` 。

如果要访问S3对象存储，则可以通过 `CREATE EXTERNAL TABLE` 命令中的自定义选项提供S3凭据，如使用DDL覆盖S3服务器配置中所述。

示例: 从S3中读取多行文本数据

执行以下步骤创建一个样例文本文件、 将文件复制到S3，并使用 PXF `s3:text:multi` 配置文件创建一个MPP数据库可读外部表来查询数据。

要运行此示例，您必须：

- 在系统上安装了AWS CLI工具

- 知道您的AWS访问ID和密钥
- 拥有对S3存储桶的写权限

1. 创建第二个带分隔符的纯文本文件：

```
$ vi /tmp/pxf_s3_multi.txt
```

2. 将以下数据复制/黏贴到 `pxf_s3_multi.txt` 中：

```
"4627 Star Rd.
San Francisco, CA  94107":Sept:2017
"113 Moon St.
San Diego, CA  92093":Jan:2018
"51 Belt Ct.
Denver, CO  90123":Dec:2016
"93114 Radial Rd.
Chicago, IL  60605":Jul:2017
"7301 Brookview Ave.
Columbus, OH  43213":Dec:2018
```

注意使用冒号 `:` 分隔三个字段。 另外还要注意第一个字段(地址)周围的引号。 此字段包含嵌入式换行符，用于将街道地址与城市和州分开。

3. 将文本文件复制到S3：

```
$ aws s3 cp /tmp/pxf_s3_multi.txt s3://BUCKET/pxf_examples/
```

4. 使用 `s3:text:multi` 配置文件创建一个引用S3文件 `pxf_s3_multi.txt` 的外部表，确保定义 `:` (冒号) 作为字段分隔符。 例如，假设您的服务名为 `s3srvcfg` ：

```
postgres=# CREATE EXTERNAL TABLE pxf_s3_textmulti(address text, mo
             LOCATION ('pxf://BUCKET/pxf_examples/pxf_s3_multi.txt?
           FORMAT 'CSV' (delimiter ':');
```

注意指定 `delimiter` 的替代语法。

5. 查询 `pxf_s3_textmulti` 表:

```
postgres=# SELECT * FROM pxf_s3_textmulti;
```

```
         address          | month | year
--------------------------+-------+------
 4627 Star Rd.            | Sept  | 2017
 San Francisco, CA  94107
 113 Moon St.             | Jan   | 2018
 San Diego, CA  92093
 51 Belt Ct.              | Dec   | 2016
 Denver, CO  90123
 93114 Radial Rd.         | Jul   | 2017
 Chicago, IL  60605
 7301 Brookview Ave.      | Dec   | 2018
 Columbus, OH  43213
(5 rows)
```

写入文本数据

“<objstore>:text”配置文件支持将单行纯文本文件写入对象存储中。 使用PXF创建可写外部表时，请指定目录名称。当您将记录写入可写外部表中，您插入的数据块将写入指定目录中的一个或多个文件中。

注意： 使用可写配置文件创建的外部表仅支持 `INSERT` 操作。 如果要查询插入的数据，则比如单独创建一个引用该目录的可读外部表。

使用以下语法创建一个引用对象存储目录的MPP数据库可写外部表：

```
CREATE WRITABLE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-dir>
    ?PROFILE=<objstore>:text[&SERVER=<server_name>][&<custom-option>=
FORMAT '[TEXT|CSV]' (delimiter[=|<space>][E]'<delim_value>');
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE 命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-dir>	S3数据存储中目录的绝对路径
PROFILE=<objstore>:text	`PROFILE` 关键字必须指定为对象存储。例如： `s3:text`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的; 如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	<custom-option> 选项见下方描述
FORMAT	将分隔的纯文本写入<path-to-dir>时，使用 `FORMAT` `'TEXT'` 。 将逗号分隔值文本写入<path-to-dir>时，使用 `FORMAT` `'CSV'` 。
delimiter	数据中的分隔符。 数据中的分隔符。对于 `FORMAT` `'CSV'` ，默认的<delim_value> 是逗号 `,` 。 当分隔符为转义序列时，在<delim_value> 前面加上 `E` 。示例： `(delimiter=E'\t')` ， `(delimiter ':')` 。

关键字	值
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用与MPP表相同的分布策略或<字段名>。 这样做可以避免数据加载操作中segment节点间额外的数据移动。.

使用 `<objstore>:text` 配置文件创建可写外部表时，可以选择使用记录或块压缩。 PXF `<objstore>:text` 配置文件支持以下压缩编解码器：

- org.apache.hadoop.io.compress.DefaultCodec
- org.apache.hadoop.io.compress.GzipCodec
- org.apache.hadoop.io.compress.BZip2Codec

您可以通过 `CREATE EXTERNAL TABLE` `LOCATION` 子句中的自定义选项指定压缩编码器。 `<objstore>:text` 配置文件支持以下自定义写入选项：

选项	值描述

COMPRESSION_CODEC	压缩编解码器Java类名。 如果未提供此选项，MPP数据库不会执行压缩编码。支持的压缩编解码器包括：
	`org.apache.hadoop.io.compress.DefaultCodec`
	`org.apache.hadoop.io.compress.BZip2Codec`
	`org.apache.hadoop.io.compress.GzipCodec`

COMPRESSION_TYPE	采用的压缩类型; 支持的值为 `RECORD` (默认) 或 `BLOCK`
THREAD-SAFE	确定表查询是否可以在多线程模式下运行的布尔值。 默认为 `TRUE`。 将此选项设置为 `FALSE` 以处理单个线程中所有非线程安全操作 (例如，压缩)的请求。

如果要访问S3对象存储，则可以通过 `CREATE EXTERNAL TABLE` 命令中的自定义选项提供S3凭据，如使用DDL覆盖S3服务器配置中所述。

示例: 将文本数据写入S3

这个实力引用在 示例: 从S3读取文本数据 中介绍的数据模式。

列名	数据类型
location	text
month	text
number_of_orders	int
total_sales	float8

此示例还可以选择使用您在该练习中创建的名为 `pxf_s3_textsimple` 的MPP数据库外部表。

步骤

执行以下步骤，使用与上述相同的数据模式创建MPP数据库可写外部表， 其中之一将使用压缩。你将使用 PXF `s3:text` 配置文件将数据写入S3。 您还将创建一个单独的可读外部表，以读取您写入S3的数据。

1. 使用上述的数据模式创建MPP数据库可写外部表。写入S3 目录 `BUCKET/pxf_examples/pxfwrite_s3_textsimple1` 。 创建表是指定逗号 `,` 作为分隔符。例如，假设您的服务名称为 `s3srvcfg` ：

   ```
   postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_s3_writetbl_1(locati
                 LOCATION ('pxf://BUCKET/pxf_examples/pxfwrite_s3_texts
               FORMAT 'TEXT' (delimiter=',');
   ```

   将 `FORMAT` 子句 `delimiter` 值指定为单个ASCII逗号字符 `,`

2. 通过在 `pxf_s3_writetbl_1` 上调用SQL `INSERT` 命令，将一些单独的记录写入 `pxfwrite_s3_textsimple1` S3目录：

```
postgres=# INSERT INTO pxf_s3_writetbl_1 VALUES ( 'Frankfurt', 'Ma
postgres=# INSERT INTO pxf_s3_writetbl_1 VALUES ( 'Cleveland', 'Oc
```

3. (可选) 将您在 示例: 从S3读取文本数据 创建的 `pxf_s3_textsimple` 表中的数据插入到 `pxf_s3_writetbl_1` ：

```
postgres=# INSERT INTO pxf_s3_writetbl_1 SELECT * FROM pxf_s3_text
```

4. MPP数据库不支持直接查询可写外部表。要查询刚刚添加到S3的数据，必须创建一个引用该S3目录的MPP数据库可读外部表：

```
postgres=# CREATE EXTERNAL TABLE pxf_s3_textsimple_r1(location tex
           LOCATION ('pxf://BUCKET/pxf_examples/pxfwrite_s3_texts
           FORMAT 'CSV';
```

创建可读外部表时，请指定 `'CSV'` `FORMAT` ，因为您创建可写外部表时使用逗号 `,` 作为分隔字符，即 `'CSV'` `FORMAT` 默认分隔符。

5. 查询可读外部表：

```
postgres=# SELECT * FROM pxf_s3_textsimple_r1 ORDER BY total_sales
```

```
 location  | month | num_orders | total_sales
-----------+-------+------------+-------------
 Rome      | Mar   |         87 |     1557.39
 Frankfurt | Mar   |        777 |     3956.98
 Prague    | Jan   |        101 |     4875.33
 Bangalore | May   |        317 |     8936.99
 Beijing   | Jul   |        411 |    11600.67
 Cleveland | Oct   |       3812 |    96645.37
(6 rows)
```

`pxf_s3_textsimple_r1` 表包含了您单独插入的记录以及 `pxf_s3_textsimple` 表的完整内容(如果执行了可选步骤)

6. 创建第二个MPP数据库可写外部表，这次使用Gzip 压缩并使用冒号 `:` 作为分隔符：

```
postgres=# CREATE WRITABLE EXTERNAL TABLE pxf_s3_writetbl_2 (locat
             LOCATION ('pxf://BUCKET/pxf_examples/pxfwrite_s3_texts
           FORMAT 'TEXT' (delimiter=':');
```

7. 通过直接写入 `pxf_s3_writetbl_2` 表将一些记录写入到 `pxfwrite_s3_textsimple2` S3目录：

```
gpadmin=# INSERT INTO pxf_s3_writetbl_2 VALUES ( 'Frankfurt', 'Mar
gpadmin=# INSERT INTO pxf_s3_writetbl_2 VALUES ( 'Cleveland', 'Oct
```

8. 要从新创建的名为 `pxfwrite_s3_textsimple2` 的S3目录查询数据，您可以如上所述创建一个MPP数据库可读外部表，该表引用此S3目录并指定 `FORMAT 'CSV' (delimiter=':')` 。

从对象存储中读取**Avro**数据

In this topic:

- 先决条件
- 使用Avro数据
- 创建外部表
- 示例

PXF 对象存储连接器支持读取Avro格式数据。 本节描述如何使用PXF访问对象存储中的Avro数据， 包括创建和查询引用对象存储中Avro文件的外部表。

注意: 从对象存储访问Avro格式数据与访问HDFS中的Avro格式数据非常相似。 本主题标识了读取对象存储所需的特定信息，并在通用信息的位置链接到 PXF HDFS Avro 文档。

先决条件

在您尝试从对象存储中读取数据前，确保已满足PXF 对象存储先决条件 。

使用Avro数据

有关Apache Avro数据序列化框架的说明，请参考PXF HDFS Avro文档中的使用Avro数据

创建外部表

使用 `<objstore>:avro` 配置文件从对象存储中读取Avro格式文件。PXF 支持以下 `<objstore>` 配置前缀：

对象存储	配置前缀
Azure Blob Storage	wasbs
Azure Data Lake	adl
Google Cloud Storage	gs
Minio	s3

对象存储	配置前缀
S3	s3

以下语法创建一个引用Avro格式文件的MPP数据库可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-file>?PROFILE=<objstore>:avro[&SERVER=<serv
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-file>	对象存储中文件或目录的绝对路径
PROFILE=<objstore>:avro	`PROFILE` 必须指定为特定的对象存储。例如， `s3:avro`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	特定于Avro的自定义选项描述见 PXF HDFS Avro 文档
FORMAT ‘CUSTOM’	在 `<objstore>:avro` 配置文件中使用 `FORMAT` `'CUSTOM'` 。 `CUSTOM` `FORMAT` 要求您指定为 `(FORMATTER='pxfwritable_import')` 。

如果要访问S3对象存储，则可以通过[CREATE EXTERNAL TABLE]命令中的自定义选项提供S3凭据，如使用DDL覆盖S3服务器配置中所述。

示例

有关Avro示例，请参阅PXF HDFS Avro 文档中的 示例: 读取Avro数据。在对象存储中运行示例，必要的修改如下：

- 将文件复制到对象存储而不是HDFS。 例如，将文件复制到S3：

```
$ aws s3 cp /tmp/pxf_avro.avro s3://BUCKET/pxf_examples/
```

- 使用上述的 `CREATE EXTERNAL TABLE` 语法和 `LOCATION` 关键字及设置。例如， 假设您的服务名为 `s3srvcfg` ：

```
CREATE EXTERNAL TABLE pxf_s3_avro(id bigint, username text, follow
  LOCATION ('pxf://BUCKET/pxf_examples/pxf_avro.avro?PROFILE=s3:av
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

从对象存储中读取JSON数据

In this topic:

- 先决条件
- 使用JSON数据
- 创建外部表
- 示例

PXF 对象存储连接器支持读取JSON格式数据。 本节描述如何使用PXF访问对象存储中的JSON数据，包括如何创建和查询引用对象存储中JSON文件的外部表。

注意: 从对象存储访问JSON数据和访问HDFS中的JSON数据非常相似。 本主题标识了读取对象存储中JSON数据所需的特定信息，并在通用信息的位置链接到 PXF HDFS JSON 文档 。

先决条件

在您尝试从对象存储中读取数据前，确保已满足PXF 对象存储先决条件 。

使用JSON数据

有关基于JSON文本的数据交换格式的说明，请参阅PXF HDFS JSON文档中的使用JSON数据。

创建外部表

使用 `<objstore>:json` 配置文件读取对象存储中的JSON格式文件。 PXF 支持以下 `<objstore>` 配置前缀:

对象存储	*配置前缀*
Azure Blob Storage	*wasbs*
Azure Data Lake	*adl*
Google Cloud Storage	*gs*
Minio	*s3*
S3	*s3*

以下语法创建一个引用JSON格式文件的MPP数据库可读外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-file>?PROFILE=<objstore>:json[&SERVER=<serv
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-file>	对象存储中文件或目录的绝对路径
PROFILE=<objstore>:json	`PROFILE` 必须指定为特定的对象存储。 例如： `s3:json` 。
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	JSON 支持的名为 `IDENTIFIER` 的自定义选项描述见 PXF HDFS JSON 文档 。
FORMAT ‘CUSTOM’	在 `<objstore>:json` 配置文件中使用 `FORMAT` `'CUSTOM'` 。 `CUSTOM` `FORMAT` 要求您指定为 `(FORMATTER='pxfwritable_import')` 。

如果要访问S3对象存储，则可以如使用DDL覆盖S3服务器配置中所述直接在 `CREATE EXTERNAL TABLE` 命令的自定义选项中提供S3凭据。

示例

有关JSON示例，请参阅PXF HDFS JSON 文档中的 将样本JSON数据加载到

HDFS 以及 示例：读取单行记录的JSON文件 。 在对象存储中运行示例，必要的修改如下：

- 将文件复制到对象存储而不是HDFS。 例如，将文件复制到S3：

```
$ aws s3 cp /tmp/singleline.json s3://BUCKET/pxf_examples/
$ aws s3 cp /tmp/multiline.json s3://BUCKET/pxf_examples/
```

- 使用上述的 `CREATE EXTERNAL TABLE` 语法和 `LOCATION` 关键字及设置。例如， 假设您的服务名为 `s3srvcfg` ：

```
CREATE EXTERNAL TABLE singleline_json_s3(
  created_at TEXT,
  id_str TEXT,
  "user.id" INTEGER,
  "user.location" TEXT,
  "coordinates.values[0]" INTEGER,
  "coordinates.values[1]" INTEGER
)
  LOCATION('pxf://BUCKET/pxf_examples/singleline.json?PROFILE=s3:j
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

在对象存储中读写Parquet数据

In this topic:

- 先决条件
- 数据类型映射
- 创建外部表
- 示例

PXF 对象存储连接器支持读写Parquet格式数据。 本节描述了如何使用PXF访问对象存储中的Parquet格式数据， 包括创建和查询引用对象存储中Parquet文件的外部表。

PXF Parquet写入支持是一个Beta功能。

注意: 从对象存储访问Parquet格式数据与访问HDFS中的Parquet格式数据非常相似。 本主题标识了读取对象存储所需的特定信息，并在通用信息的位置链接到 PXF HDFS Parquet 文档 。

先决条件

在您尝试从对象存储中读取数据前，确保已满足PXF 对象存储先决条件 。

数据类型映射

有关MPP数据库和Parquet数据类型之间映射的说明，请参阅 数据类型映射 。

创建外部表

PXF `<objstore>:parquet` 配置文件支持读取和写入Parquet格式数据。 PXF 支持以下 `<objstore>` 配置前缀：

对象存储	配置前缀
Azure Blob Storage	wasbs
Azure Data Lake	adl
Google Cloud Storage	gs

对象存储	配置前缀
Minio	s3
S3	s3

使用以下语法创建引用S3目录的MPP数据库外部表。 当您将记录插入可写外部表中时，您插入的数据块将写入指定目录中一个或多个文件。

```
CREATE [WRITABLE] EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-dir>
    ?PROFILE=<objstore>:parquet[&SERVER=<server_name>][&<custom-optic
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import'|'pxfwritable_export'
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-dir>	对象存储中文件或目录的绝对路径
PROFILE=<objstore>:parquet	`PROFILE` 必须指定为特定的对象存储。例如， `s3:parquet`
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。可选的; 如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	特定于Parquet的自定义选项描述见 PXF HDFS Parquet 文档

关键字	值
FORMAT 'CUSTOM'	使用 FORMAT '时指定 (FORMATTER='pxfwritable_export') (写入) 或 (FORMATTER='pxfwritable_import') (读取)
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用与MPP表相同的分布策略或<字段名>。 这样做可以避免数据加载操作中segment节点间额外的数据移动。

如果要访问S3 对象存储，则可以如 覆盖S3服务配置 中所述直接在 CREATE EXTERNAL TABLE 命令中提供S3凭据。 如果要访问S3对象存储库： - 您可以通过 CREATE EXTERNAL TABLE 命令中的自定义选项提供S3凭据，如用DDL覆盖S3服务器配置中所述。 - 如果您正在从S3读取Parquet数据，则可以指示PXF使用S3 Select Amazon服务检索数据。 有关用于此目的的PXF自定义选项的更多信息，请参考使用Amazon S3选择服务。

示例

有关Parquet 读取/写入的示例，请参阅PXF HDFS Parquet 文档中的 示例 。 在对象存储中运行示例，必要的修改如下：

- 使用上述的 CREATE EXTERNAL TABLE 语法和 LOCATION 关键字及设置。例如， 假设您的服务名为 s3srvcfg ：

```
CREATE WRITABLE EXTERNAL TABLE pxf_tbl_parquet_s3 (location text,
  LOCATION ('pxf://BUCKET/pxf_examples/pxf_parquet?PROFILE=s3:parq
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

- 使用上述的 CREATE EXTERNAL TABLE 语法和 LOCATION 关键字及设置。例如， 假设您的服务名为 s3srvcfg ：

```
CREATE EXTERNAL TABLE read_pxf_parquet_s3(location text, month tex
  LOCATION ('pxf://BUCKET/pxf_examples/pxf_parquet?PROFILE=s3:parq
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

在对象存储中读写SequenceFile数据

In this topic:

- 先决条件
- 创建外部表
- 示例

PXF 对象存储连接器支持SequenceFile格式二进制数据。 本节描述如何使用PXF读写SequenceFile数据，包括如何创建、写入以及查询引用对象存储文件的外部表。

注意: 从对象存储访问SequenceFile格式数据与访问HDFS中的SequenceFile格式数据非常相似。 本主题标识了读取对象存储所需的特定信息，并在通用信息的位置链接到PXF HDFS SequenceFile 文档

先决条件

在您尝试从对象存储中读取数据前，确保已满足PXF 对象存储先决条件 。

创建外部表

PXF `<objstore>:SequenceFile` 配置文件支持以SequenceFile格式读取和写入数据。 PXF 支持以下 `<objstore>` 配置文件前缀：

对象存储	配置前缀
Azure Blob Storage	wasbs
Azure Data Lake	adl
Google Cloud Storage	gs
Minio	s3
S3	s3

使用以下语法创建引用S3目录的MPP数据库外部表。 当您将记录插入可写外部表中时，您插入的数据块将写入指定目录中一个或多个文件。

```
CREATE [WRITABLE] EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-dir>
    ?PROFILE=<objstore>:SequenceFile[&SERVER=<server_name>][&<custom-
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import'|'pxfwritable_export'
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<path-to-dir>	对象存储中文件或目录的绝对路径
PROFILE=<objstore>:SequenceFile	`PROFILE` 必须指定为特定的对象存储。例如， `s3:SequenceFile`

关键字	值
SERVER =<server_name>	PXF用于访问数据的命名服务器配置。可选的;如果未指定，PXF将使用 `default` 服务器。
<custom-option>=<value>	特定于SequenceFile的自定义选项描述见 PXF HDFS SequenceFile 文档
FORMAT ‘CUSTOM’	使用 `FORMAT` ’ `CUSTOM` '时指定 `(FORMATTER='pxfwritable_export')` (写入) 或 `(FORMATTER='pxfwritable_import')` (读取)
DISTRIBUTED BY	如果您计划将现有MPP数据库表中的数据加载到可写外部表，考虑在可写外部表上使用与MPP表相同的分布策略或<字段名>。这样做可以避免数据加载操作中segment节点间额外的数据移动。

如果要访问S3对象存储，则可以如使用DDL覆盖S3服务器配置中所述直接在 `CREATE EXTERNAL TABLE` 命令的自定义选项中提供S3凭据。

示例

有关SequenceFile读取/写入的示例，请参阅PXF HDFS SequenceFile 文档中的 示例: 将二进制数据写入HDFS。在对象存储中运行示例，必要的修改如下：

- 使用上述的 `CREATE EXTERNAL TABLE` 语法和 `LOCATION` 关键字及设置。例如，假设您的服务名为 `s3srvcfg` ：

```
CREATE WRITABLE EXTERNAL TABLE pxf_tbl_seqfile_s3(location text, m
  LOCATION ('pxf://BUCKET/pxf_examples/pxf_seqfile?PROFILE=s3:Sequ
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

- 使用上述的 `CREATE EXTERNAL TABLE` 语法和 `LOCATION` 关键字及设置。例如，假设您的服务名为 `s3srvcfg` ：

```
CREATE EXTERNAL TABLE read_pxf_tbl_seqfile_s3(location text, month
  LOCATION ('pxf://BUCKET/pxf_examples/pxf_seqfile?PROFILE=s3:Sequ
 FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

将多行文本文件读入单个表行

In this topic:

- 先决条件
- 创建外部表
- 示例

PXF对象存储连接器支持将多行文本文件读取为单个表行。 本节介绍如何使用PXF读取对象存储中的多行文本和JSON数据文件，包括如何创建引用存储中多个文件的外部表。

PXF支持以这种方式仅读取文本和JSON文件。

注意: 从对象存储访问多行文件与访问HDFS中的多行文件非常相似。 本主题标识读取这些文件所需的特定于对象存储的信息。 有关更多信息，请参考PXF HDFS文档。

先决条件

在尝试从驻留在对象存储中的多个文件中读取数据之前，请确保已满足PXF对象存储的先决条件。

创建外部表

使用 `<objstore>:hdfs:multi` 配置文件将对象存储中的多个文件读入单个表行。 PXF支持以下 `<objstore>` 配置文件前缀：

对象存储	配置文件前缀
Azure Blob Storage	wasbs
Azure Data Lake	adl
Google Cloud Storage	gs
Minio	s3
S3	s3

以下语法创建了MPP数据库可读的外部表，该表引用对象存储中的一个或多个文本文件：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> text|json | LIKE <other_table> )
  LOCATION ('pxf://<path-to-files>?PROFILE=<objstore>:text:multi[&SE
  FORMAT 'CSV');
```

下表描述了CREATE EXTERNAL TABLE命令中使用的特定关键字和值。

关键字	值
<path-to-files>	对象存储中目录或文件的绝对路径。
PROFILE=<objstore>:text:multi	`PROFILE` 关键字必须标识特定的对象存储。 例如，`s3:text:multi` 。

关键字	值
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 `default` 服务器。
FILE_AS_ROW=true	指示PXF将每个文件读入单个表行的必需选项。
FORMAT	`FORMAT` 必须指定 `'CSV'` 。

如果要访问S3对象存储，则可以通过 `CREATE EXTERNAL TABLE` 命令中的自定义选项提供S3凭据，如使用DDL覆盖S3服务器配置中所述。

示例

有关示例，请参阅PXF HDFS文档中的示例：将HDFS文本文件读入单个表行。 您必须对使用对象库运行示例进行的修改包括：

- 将文件复制到对象存储而不是HDFS。 例如，要将文件复制到S3：

```
$ aws s3 cp /tmp/file1.txt s3://BUCKET/pxf_examples/tdir
$ aws s3 cp /tmp/file2.txt s3://BUCKET/pxf_examples/tdir
$ aws s3 cp /tmp/file3.txt s3://BUCKET/pxf_examples/tdir
```

- 使用上述的 `CREATE EXTERNAL TABLE` 语法和 `LOCATION` 关键字和设置。 例如，如果您的服务器名称是 `s3srvcfg` ：

```
CREATE EXTERNAL TABLE pxf_readfileasrow_s3( c1 text )
  LOCATION('pxf://BUCKET/pxf_examples/tdir?PROFILE=s3:text:multi&S
FORMAT 'CSV'
```

使用S3 Select从S3读取CSV和Parquet数据

In this topic:

- 启用PXF以使用S3 Select
- 使用S3 Select读取Parquet数据
  - 指定Parquet列的压缩类型
  - 创建外部表
- 使用S3 Select读取CSV文件
  - 处理CSV文件头
  - 指定CSV文件压缩类型
  - 创建外部表

PXF S3连接器支持使用Amazon S3 Select服务从S3读取某些CSV和Parquet格式的数据。 S3 Select提供了对Amazon S3中存储的数据的直接就地查询功能。

启用它后，PXF使用S3 Select过滤S3对象的内容以检索您请求的数据子集。 这样通常可以减少传输到MPP数据库的数据量和查询时间。

可以将PXF S3连接器与S3选择一起使用以读取：

- `gzip` 或 `bzip2` 压缩的CSV文件
- 带有 `gzip` 或 `snappy` 压缩列的 `Parquet` 文件

数据必须是 `UTF-8` 编码的，并且可能是服务器端加密的。

当使用S3 Select时，PXF支持列投影以及 `AND` 和 `OR` 或 `NOT` 运算符的谓词下推。

使用Amazon S3 Select服务可能会增加数据访问和检索的成本。 在启用PXF使用S3 Select服务之前，请务必考虑相关开销。

启用PXF以使用S3 Select

在访问S3对象存储库时，`S3_SELECT`外部表自定义选项将控制PXF对S3 Select的使用。 设置`S3_SELECT`选项时，可以提供以下值：

S3-SELECT值	描述
OFF	PXF不使用S3 Select。 默认值。
ON	PXF始终使用S3 Select。
AUTO	PXF将在有利于访问或性能的情况下使用S3 Select。

默认情况下，PXF不使用S3 Select（`S3_SELECT = OFF`）。 您可以使PXF始终使用S3 Select，或者仅在PXF确定它可能对性能有利时才使用S3 Select。 例如，当`S3_SELECT = AUTO`时，当外部表上的查询使用列投影或谓词下推时，或者当引用的CSV文件具有标题行时，PXF会自动使用S3 Select。

使用S3 Select读取Parquet数据

PXF支持从S3读取Parquet数据，如在对象存储中读取和写入Parquet数据中所述。 如果您希望PXF在读取Parquet数据时使用S3 Select，则将`S3_SELECT`自定义选项和值添加到`CREATE EXTERNAL TABLE`的`LOCATION` URI中。

指定Parquet列的压缩类型

如果Parquet文件中的列是`gzip`或`snappy`压缩的，请在`LOCATION` URI中使用`COMPRESSION_CODEC`自定义选项来标识压缩编解码器别名。 例如：

```
&COMPRESSION_CODEC=gzip
```

或，

```
&COMPRESSION_CODEC=snappy
```

创建外部表

使用以下语法在您希望PXF使用S3 Select服务访问的S3上引用一个Parquet文件的情况下，创建MPP数据库外部表：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
  LOCATION ('pxf://<path-to-file>?PROFILE=s3:parquet[&SERVER=<server_
FORMAT 'CSV';
```

当您启用PXF以使用S3 Select来访问S3上Parquet文件的外部表时，必须指定 `FORMAT 'CSV'` 。

例如，使用以下命令让PXF使用S3在最佳时机访问S3上的Parquet文件：

```
CREATE EXTERNAL TABLE parquet_on_s3 ( LIKE table1 )
  LOCATION ('pxf://bucket/file.parquet?PROFILE=s3:parquet&SERVER=s3si
FORMAT 'CSV';
```

使用S3 Select读取CSV文件

PXF支持从S3读取CSV数据，如在对象存储中读取和写入文本数据中所述。 如果您希望PXF在读取CSV数据时使用S3 Select，则可以将 `S3_SELECT` 自定义选项和值添加到 `CREATE EXTERNAL TABLE` `LOCATION` URI中。 您也可以指定定界符，文件头和压缩自定义选项。

处理CSV文件头

当您启用PXF以使用S3 Select访问CSV格式的文件时，可以在 `LOCATION` URI中使用 `FILE_HEADER` 自定义选项来标识CSV文件是否具有标题行，如果有的话，还可以确定PXF如何处理标题 `FILE_HEADER` 选项采用以下值：

FILE_HEADER值	描述
NONE	该文件没有标题行； 默认值。
IGNORE	该文件具有标题行； 忽略标题。
USE	该文件具有标题行； 读取标题。

默认的 `FILE_HEADER` 值为 `NONE` 。 您还可以指示PXF忽略或读取文件头。 例如，要让PXF忽略标题，请将以下内容添加到 `CREATE EXTERNAL TABLE` `LOCATION` URI中：

```
&FILE_HEADER=IGNORE
```

仅当S3连接器使用Amazon S3 Select服务访问S3上的文件时，PXF可以读取带标题行的CSV文件。 PXF不支持从任何其他外部数据存储中读取包含头行的CSV文件。

指定CSV文件压缩类型

如果CSV文件是 `gzip` 或 `bzip2` 压缩文件，请使用 `LOCATION` URI中的 `COMPRESSION_CODEC` 自定义选项来标识压缩编解码器别名。 例如：

```
&COMPRESSION_CODEC=gzip
```

或，

```
&COMPRESSION_CODEC=bzip2
```

创建外部表

使用以下语法创建MPP数据库外部表，该表引用您希望PXF使用S3 Select服务访问的S3上的CSV文件：

```
CREATE EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<path-to-file>
    ?PROFILE=s3:text[&SERVER=<server_name>]&S3_SELECT=ON|AUTO[&FILE_H
FORMAT 'CSV' [(delimiter '<delim_char>')];
```

例如，使用以下命令使PXF始终使用S3 Select来访问S3上的 `gzip` 压缩文件，其中字段分隔符是竖线（'|'）字符，并且您想读取标题行：

```
CREATE EXTERNAL TABLE gzippedcsv_on_s3 ( LIKE table2 )
  LOCATION ('pxf://bucket/file.csv.gz?PROFILE=s3:text&SERVER=s3srvcfg
FORMAT 'CSV' (delimiter '|');
```

使用PXF(JDBC)访问SQL数据库

In this topic:

- 先决条件
- 数据类型支持
- 访问外部SQL数据库
  - JDBC自定义选项
  - 示例: 读取和写入PostgreSQL表
- 关于使用命名查询
  - 示例：读取PostgreSQL查询的结果
- 用DDL覆盖JDBC服务器配置

您的某些数据可能已经存储在外部SQL数据库中。 PXF通过PXF JDBC连接器提供对此数据的访问。 JDBC 连接器是一个JDBC客户端。 它可以从SQL数据库(包括MySQL, ORACLE, PostgreSQL, Apache Ignite和Hive)读取或向SQL数据库写入数据。

本节描述如何使用PXF JDBC 连接器访问外部SQL数据库中的数据，包括如何创建引用外部数据库表的PXF外部表，向该表查询数据或将数据插入该表中。

写入外部SQL数据库时，JDBC 连接器不保证一致性。 请注意，如果 `INSERT` 操作失败，部分数据可能会写入外部数据库表中。 如果您需要写操作的一致性，请考虑写入到外部数据库的临时表，并仅在验证写操作后才加载到目标表。

先决条件

在您使用PXF JDBC连接器访问外部数据库前，请确保：

- 您已配置并初始化PXF, 并且PXF正在每台segment主机上运行。更多详情，请参阅配置 PXF。
- 您可以确定PXF 用户配置目录(`$PXF_CONF`)。

- 所有MPP数据库segment主机和外部SQL数据库库之间都可以连接。
- 您已经配置了外部SQL数据库，以便从所有MPP数据库segment主机进行访问。
- 您已经注册了所有JDBC驱动程序的JAR依赖。
- (推荐)您已经按照 配置PXF JDBC 连接器 中的描述创建了一个或多个命名的 PXF JDBC连接服务配置。

数据类型支持

PXF JDBC 连接器支持以下数据类型:

- INTEGER, BIGINT, SMALLINT
- REAL, FLOAT8
- NUMERIC
- BOOLEAN
- VARCHAR, BPCHAR, TEXT
- DATE
- TIMESTAMP
- BYTEA

PXF JDBC 连接器不支持上面未列出的任何数据类型。

注意: JDBC连接器不支持读取或写入以字节数组(`byte[]`)存储的Hive数据。

访问外部SQL数据库

PXF JDBC连接器支持一个名为 Jdbc 的配置文件。您可以使用此概要文件从外部SQL数据库表读取数据或将数据写入外部SQL数据库表。您还可以使用连接器在外部SQL数据库中运行静态的命名查询并读取结果。

使用以下语法创建引用外部SQL数据库表的MPP数据库外部表，并使用JDBC连接器读取或写入数据： 要访问远程SQL数据库中的数据，您可以创建一个引用该远程数据库表的可读或可写的MPP数据库外部表。MPP数据库外部表和远程数据库表或查询结果元组必须具有相同的定义；列名称和类型必须匹配。

使用以下语法创建引用远程SQL数据库表或来自远程数据库的查询结果的MPP数据库外部表：

```
CREATE [READABLE | WRITABLE] EXTERNAL TABLE <table_name>
    ( <column_name> <data_type> [, ...] | LIKE <other_table> )
LOCATION ('pxf://<external-table-name>|query:<query_name>?PROFILE=Jdbc[&SERVER=<
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import'|'pxfwritable_export');
```

CREATE EXTERNAL TABLE命令中使用的特定关键字和值见下表中描述。

关键字	值
<external-table-name>	外部表的全名。 取决于外部SQL数据库，可能包括模式名称和表名称。
query:<query_name>	要在远程SQL数据库中执行的查询的名称。
PROFILE	PROFILE 关键字必须指定为 Jdbc .
SERVER=<server_name>	PXF用于访问数据的命名服务器配置。 可选的; 如果未指定，PXF将使用 default 服务器。

关键字	值
<custom-option>=<value>	<custom-option> 是特定于配置文件的。 `Jdbc` 配置文件的选项将在下一部分讨论。
FORMAT ‘CUSTOM’	JDBC `CUSTOM` `FORMAT` 支持用于读取操作的内置 `'pxfwritable_import'` `FORMATTER` 函数和用于写入操作的内置 `'pxfwritable_export'` 函数

注意: 在创建PXF外部表时，不能在 `FORMAT` 规范中使用 `HEADER` 选项。

JDBC自定义选项

您可以在 `LOCATION` URI中包含JDBC连接器自定义选项，并在每个选项前加上 `&` 符号。 Jdbc概要文件支持的 `CREATE EXTERNAL TABLE` <custom-option>包括：

选项名称	操作	描述
BATCH_SIZE	Write	整数，标识要批处理到外部SQL数据库的 `INSERT` 操作数量。 PXF始终会验证 `BATCH_SIZE` 选项，即使是在读取操作中提供。 默认开启批处理。默认值是100。

选项名称	操作	描述
FETCH_SIZE	Read	整数，标识从外部SQL数据库读取时要缓冲的行数。 读取行批处理默认情况下处于启用状态； 默认读取大小为1000。
QUERY_TIMEOUT	Read/Write	整数，用于标识JDBC驱动程序等待语句执行的时间（以秒为单位）。 默认等待时间是无限的。

选项名称	操作	描述
POOL_SIZE	Write	在 INSERT 操作上启动线程池，并标识线程池中的线程数。默认情况下，线程池是禁用的。
PARTITION_BY	Read	启用读取分区。 分区列<column-name>:<column-type>。 您只能指定一个分区列。 JDBC连接器支持 date ， int 和 enum <column-type>值。 如果您未标识 PARTITION_BY 列，则单个PXF实例将为读取请求提供服务。
RANGE	Read	当指定 PARTITION_BY 时是必需的。 查询范围； 用作提示以帮助创建分区。 RANGE 格式取决于分区列的数据类型。 当分区列为 enum 类型时， RANGE 必须指定值列表，即<value>:<value>[:<value>[…]]，每种形成它自己的片段。 如果分区列是 int 或 date 类型，则 RANGE 必须指定<start-value>:<end-value>，并表示从<start-value>到<end-value>（含）。 如果分区列是 date 类型，请使用 yyyy-MM-dd 日期格式。

选项名称	操作	描述
INTERVAL	Read	如果指定了 PARTITION_BY 且类型为 int 或 date ，则为必填项。 一个片段的间隔[:<interval-unit>]。 与 RANGE 一起使用，以提示创建分区。 在<interval-value>中指定片段的大小。 如果分区列是 date 类型，请使用<interval-unit>指定 year ， month 或 day 。 当 PARTITION_BY 列为 enum 类型时，PXF会忽略 INTERVAL 。
QUOTE_COLUMNS	Read	控制在构造外部数据库的SQL查询是 PXF 是否应引用列名。 指定为 true 强制PXF引用所有列名称; 如果指定任何其他值，PXF不会引用列名。 如果未指定 QUOTE_COLUMNS (默认)， 当查询中 任一字段满足以下条件， PXF自动引用 所有 列名： - 包含特殊字符, 或 - 混合大小写，并且外部数据库不支持未引用的混合大小写标识符。

批量写入操作(写)

当外部SQL数据库的JDBC驱动程序支持它时，批量 INSERT 操作可能会大大提升性能。

默认情况下启用批量写， 默认批处理大小为100。 要禁用批处理或修改批处理大小的值，请使用 BATCH_SIZE 设置创建PXF外部表：

- `BATCH_SIZE=0` 或 `BATCH_SIZE=1` - 关闭批处理
- `BATCH_SIZE=(n>1)` - 将 `BATCH_SIZE` 设置为 `n`

当外部数据库的JDBC驱动程序不支持批处理时，PXF JDBC 连接器的行为取决于 `BATCH_SIZE` 设置，如下所述：

- `BATCH_SIZE` 省略 - JDBC 连接器插入时不使用批处理。
- `BATCH_SIZE=(n>1)` - `INSERT` 操作失败并且连接器返回错误。

批量读取操作

默认情况下，PXF JDBC连接器自动批处理从外部数据库表中获取的行。 默认的行获取大小为1000。要修改默认的获取大小值，请在创建PXF外部表时指定 `FETCH_SIZE`。 例如：

```
FETCH_SIZE=5000
```

如果外部数据库JDBC驱动程序不支持读取时批处理，则必须通过设置 `FETCH_SIZE=0` 来显式禁用读取行批处理。

线程池(写)

当外部数据库的JDBC驱动程序支持线程化时，PXF JDBC连接器可以通过在多个线程中处理 `INSERT` 操作来进一步提升性能。

考虑将批处理和线程池一起使用。 当一起使用时，每个线程将接收并处理一批完整的数据。如果您使用线程池而不使用批处理，则线程池中的每个线程都恰好接收一个元组。

当线程池中的任一线程失败时，JDBC 连接器返回一个错误。请注意 `INSERT` 操作失败，部分数据可能会写入外部数据库表中。

要禁用或启动线程池并设置线程池大小，请使用 `POOL_SIZE` 设置创建PXF外部表，如下所述：

- `POOL_SIZE=(n<1)` - 线程池大小是系统中的CPU数量
- `POOL_SIZE=1` - 关闭线程池
- `POOL_SIZE=(n>1)` - 将 `POOL_SIZE` 设为 `n`

分区(读)

PXF JDBC 连接器支持运行在多个segment主机上的PXF实例同时对外部SQL表的读取访问。 此功能被称为分区。 默认情况下，未启用读取分区。 要启用读取分区，请在创建PXF外部表时设置 `PARTITION_BY` ， `RANGE` 和 `INTERVAL` 自定义选项。

PXF使用您指定的 `RANGE` 和 `INTERVAL` 值以及 `PARTITON_BY` 列将外部表中的特定数据行分配给在MPP数据库segment主机上运行的PXF实例。 此列选择特定于PXF处理，与您可能已为外部SQL数据库中的表指定的分区列没有关系。

标识分区参数的示例JDBC <custom-option> 子字符串：

```
&PARTITION_BY=id:int&RANGE=1:100&INTERVAL=5
&PARTITION_BY=year:int&RANGE=2011:2013&INTERVAL=1
&PARTITION_BY=createdate:date&RANGE=2013-01-01:2016-01-01&INTERVAL=1
&PARTITION_BY=color:enum&RANGE=red:yellow:blue
```

启用分区时，PXF JDBC连接器将 `SELECT` 查询拆分为多个子查询，这些子查询检索数据的子集，每个子集称为一个片段。 JDBC连接器会自动向每个片段添加额外的查询约束（ `WHERE` 表达式），以确保从外部数据库中检索每个元组的数

据都恰好一次。

例如，当用户查询使用指定 `&PARTITION_BY=id:int&RANGE=1:5&INTERVAL=2` 的 `LOCATION` 子句创建的PXF外部表时，PXF会生成5个片段：根据分区设置两个，和最多三个隐式片段生成的碎片。 与每个片段相关的约束如下：

- 片段 1: WHERE (id < 1) - 隐式生成的片段，用于RANGE起始区间
- 片段 2: WHERE (id >= 1) AND (id < 3) - 分区设置指定的片段
- 片段 3: WHERE (id >= 3) AND (id < 5) - 分区设置指定的片段
- 片段 4: WHERE (id >= 5) - 隐式生成的片段，用于RANGE结束区间
- 片段 5: WHERE (id IS NULL) - 隐式生成的片段

PXF在MPP数据库segment之间分配片段。 在segment主机上运行的PXF实例为服务片段的主机上的每个segment生成一个线程。 如果片段的数量小于或等于在片段主机上配置的MPP segment的数量，则单个PXF实例可以为所有片段提供服务。 每个PXF实例将其结果发送回MPP数据库，在此收集它们并将其返回给用户。

当您指定 `PARTITION_BY` 选项时，根据与目标数据库的最佳JDBC连接数以及跨MPP数据库segment的最佳外部数据分配，调整 `INTERVAL` 值和单位。 `INTERVAL` 低边界由MPP数据库segment的数量驱动，而高边界由与目标数据库的可接受的JDBC连接数量驱动。 `INTERVAL` 设置会影响片段的数量，理想情况下不应设置得太高或太低。 使用多个值进行测试可以帮助您选择最佳设置。

示例: 读取和写入PostgreSQL表

在本例中，您将：

- 创建一个PostgreSQL数据库和表，并将数据写入表中
- 创建一个PostgreSQL用户并将表上的所有权限都赋予该用户

- 配置PXF JDBC 连接器以访问PostgreSQL数据库
- 创建一个引用PostgreSQL表的PXF可读外部表
- 读取PostgreSQL表中的数据
- 创建一个引用PostgreSQL表的PXF可写外部表
- 将数据写入PostgreSQL表
- 再次读取PostgreSQL表中的数据

创建一个PostgreSQL表

执行以下步骤在名为 `pgtestdb` 的数据库的 `public` 模式下创建一个名为 `forpxf_table1` 的PostgreSQL表，并向名为 `pxfuser1` 的用户赋予该表的所有权限：

1. 确定PostgreSQL服务器的主机名和端口。
2. 以 `postgres` 用户连接默认PostgreSQL数据库。 例如，假设您的PostgreSQL服务以默认端口运行在 `pserver` 主机上：

   ```
   $ psql -U postgres -h pserver
   ```

3. 创建一个名为 `pgtestdb` 的PostgreSQL数据库并连接到这个数据库：

   ```
   =# CREATE DATABASE pgtestdb;
   =# \connect pgtestdb;
   ```

4. 创建一个名为 `forpxf_table1` 的表并并向表中写入一些数据：

```
=# CREATE TABLE forpxf_table1(id int);
=# INSERT INTO forpxf_table1 VALUES (1);
=# INSERT INTO forpxf_table1 VALUES (2);
=# INSERT INTO forpxf_table1 VALUES (3);
```

5. 创建一个名为 `pxfuser1` 的PostgreSQL用户：

```
=# CREATE USER pxfuser1 WITH PASSWORD 'changeme';
```

6. 为用户 `pxfuser1` 分配 `forpxf_table1` 表的所有权限，并退出 `psql` 子系统：

```
=# GRANT ALL ON forpxf_table1 TO pxfuser1;
=# \q
```

   有了这些权限， `pxfuser1` 可以读取和写入 `forpxf_table1` 表。

7. 更新 PostgreSQL 配置以允许用户 `pxfuser1` 从每个MPP数据库segment主机访问 `pgtestdb` 。 此配置特定于您的PostgreSQL环境。 您将更新 `/var/lib/pgsql/pg_hba.conf` 文件，然后重启PostgreSQL服务。

配置JDBC连接器

您必须为PostgreSQL创建JDBC服务配置，将PostgreSQL驱动程序JAR文件下载到您的系统，将该JAR文件复制到PXF用户配置目录，同步PXF配置， 然后重启PXF。

此过程通常由MPP数据库管理员执行。

1. 登录到MPP数据库master节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 如示例配置步骤中所述为PostgreSQL创建JDBC服务配置，命名服务目录为 `pgsrvcfg` 。 `jdbc-site.xml` 文件的内容应类似于以下内容(将PostgreSQL主机系统替换为 `pgserverhost`)：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<property>
    <name>jdbc.driver</name>
    <value>org.postgresql.Driver</value>
</property>
<property>
    <name>jdbc.url</name>
    <value>jdbc:postgresql://pgserverhost:5432/pgtestdb</value>
</property>
<property>
    <name>jdbc.user</name>
    <value>pxfuser1</value>
</property>
<property>
    <name>jdbc.password</name>
    <value>changeme</value>
</property>
</configuration>
```

3. 同步PXF 配置到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

从PostgreSQL表中读取

执行以下步骤创建一个PXF外部表，该表引用您在上一节创建的 `forpxf_table1` PostgreSQL表， 并读取该表中的数据：

1. 指定 `Jdbc` 配置文件创建PXF外部表。例如：

```
gpadmin=# CREATE EXTERNAL TABLE pxf_tblfrompg(id int)
            LOCATION ('pxf://public.forpxf_table1?PROFILE=Jdbc&SER
            FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

2. 显示 `pxf_tblfrompg` 表的所有行：

```
gpadmin=# SELECT * FROM pxf_tblfrompg;
 id
----
  1
  2
  3
(3 rows)
```

写入PostgreSQL表

执行以下步骤，将一些数据写入Postgres `forpxf_table1` 表，然后从该表中读取它们。 您必须为写操作创建一个新的外部表。

1. 指定 `Jdbc` 配置文件创建一个可写的PXF外部表。 例如：

```
gpadmin=# CREATE WRITABLE EXTERNAL TABLE pxf_writeto_postgres(id i
            LOCATION ('pxf://public.forpxf_table1?PROFILE=Jdbc&SER
          FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

2. 将一些数据写入 `pxf_writeto_postgres` 表。 例如：

```
=# INSERT INTO pxf_writeto_postgres VALUES (111);
=# INSERT INTO pxf_writeto_postgres VALUES (222);
=# INSERT INTO pxf_writeto_postgres VALUES (333);
```

3. 使用您在上一节创建的可读外部表 `pxf_tblfrompg` 来查看PostgreSQL

`forpxf_table1` 表中的数据：

```
gpadmin=# SELECT * FROM pxf_tblfrompg ORDER BY id DESC;
 id
-----
 333
 222
 111
   3
   2
   1
(6 rows)
```

关于使用命名查询

PXF JDBC连接器允许您指定静态定义的查询以对远程SQL数据库运行。 考虑在以下情况下使用命名查询：

- 您需要联接所有都驻留在同一外部数据库中的几个表。
- 您想在数据源附近执行复杂的聚合。
- 您将在外部数据库中使用但不允许创建 `VIEW` 。
- 您宁愿消耗外部系统中的计算资源，以最大程度地减少MPP数据库资源的利用率。
- 您要运行HIVE查询并通过YARN控制资源利用率。

MPP数据库管理员定义了一个查询，并为您提供了创建外部表时要使用的查询名称。 在表 `CREATE EXTERNAL TABLE` `LOCATION` 子句中指定 `query:<query_name>` 代替表名，以指示PXF JDBC连接器在远程SQL数据库中运行名为 `<query_name>` 的静态查询。

PXF仅支持具有可读外部表的命名查询。 您必须为要运行的每个查询创建一个唯一的MPP数据库可读外部表。

外部表列的名称和类型必须与查询结果返回的列的名称，类型和顺序完全匹配。如果查询返回聚合或其他函数的结果，请确保使用 `AS` 限定符指定特定的列名。

例如，假设您正在使用具有以下定义的PostgreSQL表：

```
CREATE TABLE customers(id int, name text, city text, state text);
CREATE TABLE orders(customer_id int, amount int, month int, year int,
```

这个PostgreSQL查询中，管理员将其命名为 `order_rpt` ：

```
SELECT c.name, sum(o.amount) AS total, o.month
  FROM customers c JOIN orders o ON c.id = o.customer_id
  WHERE c.state = 'CO'
GROUP BY c.name, o.month
```

该查询返回类型为 `(name text, total int, month int)` 的元组。 如果为名为 `pgserver` 的PXF JDBC服务器定义了 `order_rpt` 查询，则可以创建MPP数据库外部表来读取这些查询结果，如下所示：

```
CREATE EXTERNAL TABLE orderrpt_frompg(name text, total int, month int
  LOCATION ('pxf://query:order_rpt?PROFILE=Jdbc&SERVER=pgserver&PARTI
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

该命令引用在 `pgserver` 服务器配置中定义的名为 `order_rpt` 的查询。 它还指定了JDBC读取分区选项，这些选项为PXF提供了用于在其服务器segment之间划分查询结果数据的信息。

PXF JDBC连接器自动将列投影和过滤器下推应用于引用命名查询的外部表。

示例：读取PostgreSQL查询的结果

在此示例中，您：

- 使用在示例：读取和写入PostgreSQL数据库中创建的PostgreSQL数据库 `pgtestdb` ，用户 `pxfuser1` 和PXF JDBC连接器服务器配置 `pgsrvcfg` 。
- 创建两个PostgreSQL表并将数据插入表中。
- 将表上的所有特权分配给 `pxfuser1` 。
- 定义一个在两个PostgreSQL表上执行复杂SQL语句的命名查询，并将该查询添加到 `pgsrvcfg` JDBC服务器配置中。
- 创建一个与查询结果元组匹配的PXF可读外部表定义，并指定读取分区选项。
- 使用PXF列投影和过滤器下推读取查询结果。

创建PostgreSQL表并分配权限

执行以下过程，以在名为 `pgtestdb` 的数据库的 `public` 模式中创建名为 `customers` 和 `orders` 的PostgreSQL表，并向用户 `pxfuser1` 授予这些表的所有特权：

1. 确定PostgreSQL服务器的主机名和端口。
2. 以postgres用户身份连接到 `pgtestdb` PostgreSQL数据库。 例如，如果您的PostgreSQL服务器正在名为 `pserver` 的主机的默认端口上运行：

   ```
   $ psql -U postgres -h pserver -d pgtestdb
   ```

3. 创建一个名为 `customers` 的表，并将一些数据插入该表：

```
CREATE TABLE customers(id int, name text, city text, state text);
INSERT INTO customers VALUES (111, 'Bill', 'Helena', 'MT');
INSERT INTO customers VALUES (222, 'Mary', 'Athens', 'OH');
INSERT INTO customers VALUES (333, 'Tom', 'Denver', 'CO');
INSERT INTO customers VALUES (444, 'Kate', 'Helena', 'MT');
INSERT INTO customers VALUES (555, 'Harry', 'Columbus', 'OH');
INSERT INTO customers VALUES (666, 'Kim', 'Denver', 'CO');
INSERT INTO customers VALUES (777, 'Erik', 'Missoula', 'MT');
INSERT INTO customers VALUES (888, 'Laura', 'Athens', 'OH');
INSERT INTO customers VALUES (999, 'Matt', 'Aurora', 'CO');
```

4. 创建一个名为 `orders` 的表，并将一些数据插入该表：

```
CREATE TABLE orders(customer_id int, amount int, month int, year i
INSERT INTO orders VALUES (111, 12, 12, 2018);
INSERT INTO orders VALUES (222, 234, 11, 2018);
INSERT INTO orders VALUES (333, 34, 7, 2018);
INSERT INTO orders VALUES (444, 456, 111, 2018);
INSERT INTO orders VALUES (555, 56, 11, 2018);
INSERT INTO orders VALUES (666, 678, 12, 2018);
INSERT INTO orders VALUES (777, 12, 9, 2018);
INSERT INTO orders VALUES (888, 120, 10, 2018);
INSERT INTO orders VALUES (999, 120, 11, 2018);
```

5. 为用户 `pxfuser1` 分配表 `customers` 和 `orders` 的所有特权，然后退出 `psql` 子系统：

```
GRANT ALL ON customers TO pxfuser1;
GRANT ALL ON orders TO pxfuser1;
\q
```

配置命名查询

在此过程中，您将创建一个命名查询文本文件，将其添加到 `pgsrvcfg` JDBC服务

器配置中，并将PXF配置同步到MPP数据库集群。

此过程通常由MPP数据库管理员执行。

1. 登录到MPP数据库主节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 导航到JDBC服务器配置目录 `pgsrvcfg` 。 例如：

```
gpadmin@gpmaster$ cd $PXF_CONF/servers/pgsrvcfg
```

3. 在文本编辑器中打开名为 `pg_order_report.sql` 的查询文本文件，然后将以下查询复制/粘贴到该文件中：

```
SELECT c.name, c.city, sum(o.amount) AS total, o.month
  FROM customers c JOIN orders o ON c.id = o.customer_id
  WHERE c.state = 'CO'
GROUP BY c.name, c.city, o.month
```

4. 保存文件并退出编辑器。

5. 将对PXF配置的这些更改同步到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

阅读查询结果

在MPP数据库群集上执行以下过程，以创建一个PXF外部表，该表引用您在上一节中创建的查询文件，然后读取查询结果数据：

1. 创建指定 Jdbc 配置文件的PXF外部表。 例如：

```
CREATE EXTERNAL TABLE pxf_queryres_frompg(name text, city text, to
  LOCATION ('pxf://query:pg_order_report?PROFILE=Jdbc&SERVER=pgsrv
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

通过这种分区方案，PXF将向远程SQL数据库发出4个查询，每季度一个查询。 每个查询将返回目标季度每个月的客户名称以及给定月份中每个客户每月的总订单总额。 然后，当您查询外部表时，MPP数据库将为您将数据合并到单个结果集中。

2. 显示查询结果的所有行：

```
SELECT * FROM pxf_queryres_frompg ORDER BY city, total;

 name |  city   | total | month
------+---------+-------+-------
 Matt | Aurora |   120 |    11
 Tom  | Denver |    34 |     7
 Kim  | Denver |   678 |    12
(3 rows)
```

3. 使用列投影来显示每个城市的订单总数：

```
SELECT city, sum(total) FROM pxf_queryres_frompg GROUP BY city;

  city   | sum
---------+-----
 Aurora | 120
 Denver | 712
(2 rows)
```

当您执行此查询时，PXF仅请求和检索 city 和 total 列的查询结果，从而减少了发送回MPP数据库的数据量。

4. 提供其他过滤器和聚合以过滤PostgreSQL中的 total ：

```
SELECT city, sum(total) FROM pxf_queryres_frompg
             WHERE total > 100
             GROUP BY city;

  city   | sum
--------+-----
 Denver | 678
 Aurora | 120
(2 rows)
```

在此示例中，PXF将向子查询添加 `WHERE` 过滤器。 该过滤器被推送到远程数据库系统并在其上执行，从而减少了PXF发送回MPP数据库的数据量。 但是， `GROUP BY` 聚合不会推送到远程，而是由MPP执行。

用DDL覆盖JDBC服务器配置

您可以通过直接在 `CREATE EXTERNAL TABLE` `LOCATION` 子句中指定自定义选项，来为特定的外部数据库表覆盖JDBC服务器配置中的某些属性：

自定义选项名称	jdbc-site.xml属性名
JDBC_DRIVER	jdbc.driver
DB_URL	jdbc.url
USER	jdbc.user
PASS	jdbc.password
BATCH_SIZE	jdbc.statement.batchSize
FETCH_SIZE	jdbc.statement.fetchSize
QUERY_TIMEOUT	jdbc.statement.queryTimeout

通过自定义选项指定的示例JDBC连接字符串：

```
&JDBC_DRIVER=org.postgresql.Driver&DB_URL=jdbc:postgresql://pgserverh
&JDBC_DRIVER=com.mysql.jdbc.Driver&DB_URL=jdbc:mysql://mysqlhost:3306
```

例如：

```
CREATE EXTERNAL TABLE pxf_pgtbl(name text, orders int)
  LOCATION ('pxf://public.forpxf_table1?PROFILE=Jdbc&JDBC_DRIVER=org.postgresql.Dri
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

您以这种方式提供的凭据在外部表定义中可见。 不要在生产环境中使用这种传递凭据的方法。

有关PXF用于获取MPP数据库用户的配置属性设置的优先级规则的详细信息，请参考配置属性优先级。

PXF故障排查

In this topic:

- PXF错误
- PXF日志
  - 服务级别日志记录
  - 客户端级别日志记录
- 解决PXF内存问题
  - 配置内存不足条件操作
  - 为PXF增加JVM内存
  - 资源受限的PXF segment主机的另一种选择
- 解决PXF JDBC连接器时区错误
- PXF片段元数据缓存

PXF错误

下表描述了使用PXF时可能会遇到的一些错误：

Error Message	Discussion
Protocol “pxf” does not exist	**Cause**: pxf扩展名未注册。 **Solution**: 按照PXF启用过程中的说明为Enable Procedure 为数据库创建(启用)PXF扩展
Invalid URI pxf://<path-to-data>: missing options section	**Cause**: `LOCATION` URI配置项不包括配置或其他需要的信息。 **Solution**:在URI中提供配置和必需的选项。

Error Message	Discussion
org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: hdfs://<namenode>:8020/<path-to-file>	**Cause**: 在<path-to-file>中指定的HDFS文件不存在。 **Solution**: 指定现有HDFS文件的路径
NoSuchObjectException(message:<schema>.<hivetable> table not found)	**Cause**: <schema>.<hivetable>指定的Hive表不存在。 **Solution**: 提供存在的Hive表的名称。
Failed to connect to <segment-host> port 5888: Connection refused (libchurl.c:944) (<segment-id> slice<N> <segment-host>:40000 pid=<process-id>) ...	**Cause**: 在<segment-host>主机上PXF未运行。 **Solution**: <segment-host>主机上重启PXF
ERROR: failed to acquire resources on one or more segments DETAIL: could not connect to server: Connection refused Is the server running on host “<segment-host>” and accepting TCP/IP connections on port 40000?(seg<N> <segment-host>:40000)	**Cause**: GPDB集群 <segment-host> 节点down
org.apache.hadoop.security.AccessControlException: Permission denied: user=, access=READ, inode=“”:::-rw——-	**Cause**: 执行PXF操作的MPP数据库用户无权访问基础Hadoop服务(HDFS或Hive)。参阅Configuring User Impersonation and Proxying.

PXF日志

启用更详细的日志记录可能有助于PXF故障排除工作。 PXF提供了两类消息日志记录：服务级别和客户端级别。

服务级别日志记录

PXF利用 log4j 进行服务级别的日志记录。PXF-service-related日志消息捕获由 $PXF_CONF/conf/pxf-log4j.properties 文件中的 log4j 控制。默认的PXF日志记录配置会将INFO和更严格的日志记录写入 $PXF_CONF/logs/pxf-service.log 。您可以配置日志记录级别和日志文件位置。

启用 DEBUG 级别时，PXF提供更详细的日志记录。要配置PXF DEBUG 日志记录并检查输出，请执行以下操作：

1. 登录gpdb集群的master主机

   ```
   $ ssh gpadmin@<gpmaster>
   ```

2. 使用编辑器打开 $PXF_CONF/conf/pxf-log4j.properties ，取消以下行的注释，保存文件，然后退出编辑器：

   ```
   #log4j.logger.org.mpp.pxf=DEBUG
   ```

3. 使用 pxf cluster sync 命令拷贝更新的文件 pxf-log4j.properties 到MPP数据库集群。例如：

   ```
   gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
   ```

4. 依照Restarting PXF章节描述，重启gpdb集群的每个segment节点的pxf

5. 现在启用了 DEBUG 级别的日志记录，您可以执行PXF操作。 确保记下时

间；这会将信息定向输出到 `$PXF_CONF/logs/pxf-service.log` 中的相关日志。

```
$ date
Wed Oct  4 09:30:06 MDT 2017
$ psql -d <dbname>
```

6. 创建和查询外部表。例如：

```
dbname=> CREATE EXTERNAL TABLE hdfstest(id int, newid int)
    LOCATION ('pxf://data/dir/hdfsfile?PROFILE=hdfs:text')
    FORMAT 'TEXT' (delimiter='E',');
dbname=> SELECT * FROM hdfstest;
<select output>
```

7. 最后，从 `pxf-service.log` 检查/收集日志消息。

Note: `DEBUG` 记录非常冗长，并且会对性能产生影响. 在收集了所需的信息之后，请关闭PXF服务的 `DEBUG` 日志记录。

客户端级别日志记录

数据库级客户端日志记录可以提供对内部PXF服务操作的了解。

在psql会话中将 `client_min_messages` 服务器配置参数设置为 `DEBUG2` ，可以在对PXF外部表进行操作期间启用MPP数据库及PXF调试消息日志记录。

```
$ psql -d <dbname>
```

```
dbname=# SET client_min_messages=DEBUG2;
dbname=# SELECT * FROM hdfstest;
...
DEBUG2:  churl http header: cell #19: X-GP-URL-HOST: seghost1  (seg0
CONTEXT:  External table hdfstest
DEBUG2:  churl http header: cell #20: X-GP-URL-PORT: 5888  (seg0 slic
CONTEXT:  External table hdfstest
DEBUG2:  churl http header: cell #21: X-GP-DATA-DIR: data/dir/hdfsfil
CONTEXT:  External table hdfstest
DEBUG2:  churl http header: cell #22: X-GP-OPTIONS-PROFILE: hdfs:text
CONTEXT:  External table hdfstest
...
```

检查/收集来自stdout的日志消息

Note: `DEBUG2` *数据库会话日志记录会影响性能。记住，在收集了所需的信息之后，请关闭* `DEBUG2` *日志记录。*

```
dbname=# SET client_min_messages=NOTICE;
```

解决PXF内存问题

因为单个PXF客户端服务(JVM)为segment主机上的多个segment实例提供服务，所以PXF堆大小可能是限制运行的瓶颈。在并发工作负载时针对大文件的查询，影响更加明显。您可能会遇到由于内存不足或Java垃圾收集器影响响应时间而导致查询挂起或失败的情况。要避免或纠正这些情况，请首先尝试增加Java最大堆大小或减少Tomcat最大线程数，这取决于最适合您系统配置的方式。您还可以选择将PXF配置为在检测到内存不足情况时执行特定的操作。

Note: *本主题中描述的配置更改需要在MPP数据库集群的每个节点上修改配置文件。 在主服务器上执行更新后，请确保将PXF配置同步到MPP数据库集群。*

配置内存不足条件操作

在内存不足（OOM）的情况下，PXF返回以下错误以响应查询：

```
java.lang.OutOfMemoryError: Java heap space
```

您可以将PXF JVM配置为在检测到OOM条件时启用/禁用以下操作：

- 自动杀死PXF服务器（默认情况下启用）。
- 转储Java堆（默认情况下禁用）。

自动杀死PXF服务器

默认情况下，对PXF进行了配置，以便当PXF JVM在segment主机上检测到内存不足情况时，它将自动运行一个脚本，该脚本将杀死主机上运行的PXF服务器。 `PXF_OOM_KILL` 配置属性控制此自动终止行为。

启用自动终止功能后，PXF JVM检测到OOM条件并终止segment主机上的PXF服务器：

- PXF将以下消息记录到segment主机上的 `$PXF_CONF/logs/catalina.out` 中：

  ```
  =====> <date> PXF Out of memory detected <======
  =====> <date> PXF shutdown scheduled <======
  ```

- 在PXF外部表上运行的任何查询都将失败，并显示以下错误，直到您在segment主机上重新启动PXF服务器为止：

  ```
  ... Failed to connect to <host> port 5888: Connection refused
  ```

当以这种方式关闭segment主机上的PXF服务器时，必须显式重新启动主机

上的PXF服务器。 有关 `pxf start` 命令的更多信息，请参见pxf参考页。

有关禁用/启用此PXF配置属性的说明，请参考下面的配置procedure。

转储Java堆

在内存不足的情况下，捕获Java堆转储以帮助确定导致资源耗尽的因素可能很有用。 您可以使用 `PXF_OOM_DUMP_PATH` 属性来配置PXF在检测到OOM条件时将堆转储写入文件。 默认情况下，PXF不会在OOM上转储Java堆。

如果选择在OOM上启用堆转储，则必须将 `PXF_OOM_DUMP_PATH` 设置为文件或目录的绝对路径：

- 如果指定目录，则PXF JVM将堆转储写入文件 `<directory>/java_pid<pid>.hprof` ，其中 `<pid>` 标识PXF服务器实例的进程ID。 每次JVM进行OOM操作时，PXF JVM都会将新文件写入目录。
- 如果指定文件，但该文件不存在，则PXF JVM在检测到OOM时会将堆转储写入文件。 如果文件已经存在，则JVM将不会转储堆。

确保 `gpadmin` 用户对转储文件或目录具有写权限。

Note: 堆转储文件通常很大。 如果在OOM上为PXF启用堆转储，并为 `PXF_OOM_DUMP_PATH` 指定一个目录，则多个OOM将在该目录中生成多个文件，并可能消耗大量磁盘空间。 如果为 `PXF_OOM_DUMP_PATH` 指定文件，则在文件名不变的情况下磁盘使用率是恒定的。 您必须重命名转储文件或配置其他 `PXF_OOM_DUMP_PATH` 才能生成后续的堆转储。

有关启用/禁用此PXF配置属性的说明，请参阅下面的配置步骤。

步骤

默认情况下，将启用OOM上PXF服务器的自动终止功能。 默认情况下，在OOM上禁用堆转储生成。 要配置这些属性之一或全部，请执行以下步骤：

1. 登录到您的MPP数据库主节点：

```
$ ssh gpadmin@<gpmaster>
```

2. 编辑 `$PXF_CONF/conf/pxf-env.sh` 文件。 例如：

```
gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
```

3. 如果您想在OOM上配置（即关闭或重新打开）PXF服务器自动关闭功能，请在 `pxf-env.sh` 文件中找到 `PXF_OOM_KILL` 属性。 如果该设置已被注释掉，请取消注释它，然后更新该值。 例如，要关闭此行为，请将值设置为 `false` ：

```
export PXF_OOM_KILL=false
```

4. 如果您要在PXF服务器达到OOM条件时配置（即打开或关闭）自动堆转储，请在 `pxf-env.sh` 文件中找到 `PXF_OOM_DUMP_PATH` 设置。

    1. 要打开此行为，请将 `PXF_OOM_DUMP_PATH` 属性值设置为您希望PXF JVM将Java堆转储到的文件系统位置。 例如，要转储到名为 `/home/gpadmin/pxfoom_segh1` 的文件：

    ```
    export PXF_OOM_DUMP_PATH=/home/pxfoom_segh1
    ```

    2. 要在打开堆转储后将其关闭，请注释掉 `PXF_OOM_DUMP_PATH` 属性设置：

    ```
    #export PXF_OOM_DUMP_PATH=/home/pxfoom_segh1
    ```

5. 保存 `pxf-enf.sh` 文件并退出编辑器。

6. 使用 `pxf cluster sync` 命令将更新的 `pxf-env.sh` 文件复制到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

7. 如重新启动PXF中所述，在每个MPP数据库segment主机上重新启动PXF。

为PXF增加JVM内存

在segment主机上运行的每个PXF代理都配置有默认的最大Java堆大小2GB和初始堆大小1GB。 如果MPP数据库群集中的segment主机具有足够的内存，请尝试将最大堆大小增加到3-4GB之间的值。 如果可以，将初始堆大小和最大堆大小设置为相同的值最佳。

执行以下过程来增加在MPP数据库集群中每个segment主机上运行的PXF代理服务的堆大小。

1. 登录gpdb集群的master主机

```
$ ssh gpadmin@<gpmaster>
```

2. 编辑 `$PXF_CONF/conf/pxf-env.sh` 文件。 例如：

```
gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
```

3. 在 `pxf-env.sh` 文件中找到 `PXF_JVM_OPTS` 设置，然后将 `-Xmx` 和/或 `-Xms` 选项更新为所需的值。 例如：

```
PXF_JVM_OPTS="-Xmx3g -Xms3g"
```

4. 保存文件并退出编辑器。

5. 使用 `pxf cluster sync` 命令将更新的 `pxf-env.sh` 文件复制到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

6. 依照Restarting PXF描述，重启每个segment主机上的pxf服务。

资源受限的PXF segment主机的另一种选择

如果增加最大堆大小不适合您的MPP数据库部署，请尝试减少PXF的的并发工作线程数。正在运行的线程数量的减少将防止任何PXF节点耗尽其内存，同时确保当前查询运行完毕(尽管速度稍慢)。 Tomcat的默认行为是将请求排队，直到线程空闲或队列耗尽为止。

PXF的Tomcat线程的默认最大数量为200。 `PXF_MAX_THREADS` 配置属性控制此设置。

PXF线程容量由配置文件以及是否压缩数据确定。 如果计划在外部Hive数据存储中的大量文件上运行大型工作负载，或者正在读取压缩的ORC或Parquet数据，请考虑指定较低的 `PXF_MAX_THREADS` 值。

Note: 请记住，线程数用完后，线程数的增加与内存消耗的增加相关。

执行以下过程，以设置在MPP数据库部署中每个segment主机上运行的PXF代理的Tomcat线程的最大数量。

1. 登录到gpdb集群的master节点

```
$ ssh gpadmin@<gpmaster>
```

2. 编辑 `$PXF_CONF/conf/pxf-env.sh` 。 例如:

```
gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
```

3. 在 `pxf-env.sh` 文件中找到 `PXF_MAX_THREADS` 设置。 取消注释设置，并将其更新为所需的值。 例如，要将Tomcat线程的最大数量设置为100：

```
export PXF_MAX_THREADS=100
```

4. 保存文件并退出编辑器。

5. 使用 `pxf cluster sync` 命令将更新的 `pxf-env.sh` 文件复制到MPP数据库集群。 例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

6. 依照Restarting PXF描述重启PXF服务

解决PXF JDBC连接器时区错误

您可以使用PXF JDBC连接器访问存储在外部SQL数据库中的数据。如果为PXF服务器设置的默认时区与为外部SQL数据库设置的时区不匹配，则取决于JDBC驱动程序，该程序可能会返回错误。

例如，如果使用PXF JDBC连接器访问时区冲突的Oracle数据库，则PXF会记录类似于以下内容的错误：

```
SEVERE: Servlet.service() for servlet [PXF REST Service] in context v
java.io.IOException: ORA-00604: error occurred at recursive SQL level
ORA-01882: timezone region not found
```

如果遇到此错误，可以在 `$PXF_CONF/conf/pxf-env.sh` 配置文件中的 `PXF_JVM_OPTS` 属性设置中为PXF服务器设置默认时区选项。 例如，要设置时区：

```
export PXF_JVM_OPTS="<current_settings> -Duser.timezone=America/Chica
```

您也可以使用 `PXF_JVM_OPTS` 属性来设置其他Java选项。

如前几节所述，您必须将更新的PXF配置同步到MPP数据库集群，然后在每个segment主机上重新启动PXF服务器。

PXF片段元数据缓存

PXF连接器Fragmenter使用来自外部数据源的元数据将数据拆分为可并行读取的片段列表（块，文件等）。 PXF在每个查询的基础上缓存片段元数据：访问片段元数据的第一个线程将信息存储在缓存中，其他线程重用此缓存的元数据。这种性质的缓存减少了具有大量片段的外部数据源对查询内存的需求。

默认情况下，PXF片段元数据缓存处于启用状态。 要关闭片段元数据缓存，或在关闭后重新启用片段元数据缓存，请执行以下过程：

1. 登录到您的MPP数据库主节点：

   ```
   $ ssh gpadmin@<gpmaster>
   ```

2. 编辑 `$PXF_CONF/conf/pxf-env.sh` 文件。 例如：

   ```
   gpadmin@gpmaster$ vi $PXF_CONF/conf/pxf-env.sh
   ```

3. 在 `pxf-env.sh` 文件中找到 `PXF_FRAGMENTER_CACHE` 设置。 如果该设置已被

注释掉，请取消注释它，然后更新该值。 例如，要关闭片段元数据缓存，请将值设置为 `false` ：

```
export PXF_FRAGMENTER_CACHE=false
```

4. 保存文件并退出编辑器。

5. 使用 `pxf cluster sync` 命令将更新的 `pxf-env.sh` 文件复制到MPP数据库集群。例如：

```
gpadmin@gpmaster$ $GPHOME/pxf/bin/pxf cluster sync
```

6. 如重新启动PXF中所述，在每个MPP数据库segment主机上重新启动PXF。

PXF实用程序参考手册

MPP平台扩展框架(PXF)包括以下实用程序参考页：

- pxf cluster
- pxf

pxf cluster

In this topic:

- 概要
- 描述
- Commands
- 举例
- See Also

在所有MPP数据库主机上管理PXF配置和PXF服务实例。

概要

```
pxf cluster <command>
```

where `<command>` is:

```
help
init
reset
start
status
stop
sync
```

描述

`pxf cluster` 工具命令在master,standby服务器以及所有MPP数据库segment主机上管理PXF。您可以使用该工具执行以下操作：

- 在MPP数据库集群中的所有主机上初始化PXF配置。

- 将所有主机上的PXF服务实例重置为其未初始化状态。
- 在所有segment主机上启动和停止PXF服务实例。
- 显示所有segment主机上的PXF服务实例的状态。
- 将PXF配置从MPP数据库master主机同步到standby和所有segment主机。

`pxf cluster` 需要一个正在运行的MPP数据库集群。 您必须在MPP数据库master主机上运行该程序。

（如果要在特定segment主机上管理PXF服务实例，请使用pxf工具。请参见 `pxf` 。）

Commands

help

显示 `pxf cluster` 帮助消息，然后退出

init

在master,standby节点和所有segment主机上初始化PXF服务实例。在MPP数据库集群中初始化PXF时，必须通过名为 `$PXF_CONF` 的环境变量来标识PXF用户配置目录。 如果在初始化PXF之前未设置 `$PXF_CONF` ，则PXF返回错误。

reset

在master，standby和所有segment主机上重置PXF服务实例。重置将删除PXF运行时文件和目录，并使PXF返回未初始化状态。在MPP数据库集群中重置PXF之前，必须停止在每个segment主机上运行的PXF服务实例。

start

在所有segment主机上启动PXF服务实例。

status

显示所有segment主机上的PXF服务实例的状态。

stop

在所有段主机上停止PXF服务实例。

sync

将PXF配置从master主机同步到standby和所有MPP数据库segment主机。如果您更新了PXF用户配置或添加了JAR文件，则在同步PXF配置后还必须重新启动PXF。

举例

在所有segment主机上停止PXF服务

```
$ $GPHOME/pxf/bin/pxf cluster stop
```

See Also

pxf

pxf

In this topic:

- 概要
- 描述
- 命令
- 选项
- 示例
- See Also

在本地MPP数据库主机上管理PXF配置和PXF服务实例。

概要

```
pxf <command> [<option>]
```

<command> 支持的选项:

```
cluster
help
init
reset
restart
start
status
stop
sync
version
```

描述

`pxf` 工具管理本地MPP数据库主机上的PXF配置和PXF服务实例。

您可以在master，standby或特定segment主机上初始化或重置PXF。您还可以将PXF配置从master服务器同步到这些主机。

您可以在特定的segment主机上启动，停止或重新启动PXF服务实例，或者显示在segment主机上运行的PXF服务实例的状态。

(Use the `pxf cluster` command to initialize or reset PXF on all hosts, synchronize the PXF configuration to the MPP Database cluster, or to start, stop, or display the status of the PXF service instance on all segment hosts in the cluster.)（使用 `pxf cluster` 命令在所有主机上初始化或重置PXF，将PXF配置同步到MPP数据库集群，或者启动，停止或显示集群中所有segment主机上的PXF服务实例的状态。）

命令

cluster
: 在所有MPP数据库主机上管理PXF配置和PXF服务实例。请参阅 `pxf群集` 。

help
: 显示pxf帮助消息，然后退出

init
: 在主机上初始化PXF服务实例。初始化PXF时，必须通过名为 `$PXF_CONF` 的环境变量来标识PXF用户配置目录。如果在初始化PXF之前未设置 `$PXF_CONF` ，则PXF会提示您在初始化过程中接受或拒绝默认用户配置目录 `$HOME/pxf` 。参阅 Options.

reset
: 重置主机上运行的PXF服务实例。重置将删除PXF运行时文件和目录，并

使PXF返回未初始化状态。在主机上重置PXF之前，必须停止在segment主机上运行的PXF服务实例。

restart

在segment主机上重新启动PXF服务

start

在segment主机上启动PXF服务

status

显示在segment主机上运行的PXF服务实例的状态

stop

停止在segment主机上运行的PXF服务实例

sync

将PXF配置从master服务器同步到特定的MPP数据库standby服务器或segment主机。您必须在master主机上运行 `pxf sync` 。请参阅选项。

version

显示PXF的版本信息并退出

选项

`pxf init` 命令支持以下选项:

-y

如果未设置环境变量，则不提示使用默认的 `$PXF_CONF` 目录位置。

`pxf reset` 命令采用以下选项：

-f | –force

重置PXF服务实例之前不要提示；无需用户干预即可重置。

在MPP数据库master主机上运行的 `pxf sync` 命令具有以下选项：

<gphost>
将PXF配置同步到的MPP数据库主机。 `<gphost>` 必须标识standby主机或segment主机。

示例

在segment主机本地启动PXF实例：

```
$ $GPHOME/pxf/bin/pxf start
```

See Also

`pxf cluster`