Arenadata™ Database

Версия - v5.28.11-arenadata15

Интеграция кластера ADB с JDBC-совместимыми источниками

2021 Arenadata™

Оглавление

1 Плагин PXF JDBC **3**

2 Предварительные условия **4**

3 Ограничения **5**

4 Синтаксис **6**

5 Запросы SELECT **8**

6 Запросы INSERT **9**
6.1 Пакетный режим 9

7 Пул потоков **10**

8 Предварительный запрос **11**

9 Партиционирование **12**
9.1 Синтаксис 12
9.2 Механизм работы 13
9.3 Пример партиционирования 13

10 Примеры **14**

11 Лучшие практики по использованию PXF JDBC коннектора **15**

В документе приведен принцип интеграции кластера Arenadata DB с JDBC-совместимыми источниками.

Документ может быть полезен администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий и информационной безопасности, осуществляющих внедрение кластера.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Плагин PXF JDBC

Плагин **PXF JDBC** позволяет получить доступ к внешним базам данных, реализующим **JDBC** API. Плагин поддерживает запросы чтения (*SELECT*) и записи (*INSERT*); он сам является клиентом **JDBC**, поэтому разворачивать **PXF** на хосте с внешней базой данных не требуется.

Глава 2

Предварительные условия

Для использования плагина **PXF JDBC** должны быть выполнены следующие условия:

- Плагин **PXF JDBC** установлен на всех узлах **PXF**;
- Драйвер **JDBC** для внешней базы данных установлен на всех узлах **PXF**;
- Все узлы **PXF** имеют возможность подключения к внешней базе данных.

Глава 3

Ограничения

Таблица **PXF** и таблица во внешней базе данных должны иметь одинаковую схему – столбцы должны иметь имена и типы, совпадающие с именами столбцов и типами внешней таблицы в **ADB**.

Плагин поддерживает следующие типы данных:

- *INTEGER*, *BIGINT*, *SMALLINT*;
- *REAL*, *FLOAT8*;
- *NUMERIC*;
- *BOOLEAN*;
- *VARCHAR*, *BPCHAR*, *TEXT*;
- *DATE*;
- *TIMESTAMP*;
- *BYTEA*.

Имя <*full_external_table_name*> не должно соответствовать регулярному выражению:

```
/.*/[0-9]*-[0-9]*_[0-9]*
```

То есть имя не должно начинаться с знака "/" или заканчиваться этим знаком, состоять из трех групп чисел произвольной длины, первые две из которых разделены знаком "-", а последние – "_". Например, следующее имя таблицы недопустимо: /public.table/1-2_3.

Important: Одна внешняя таблица ADB не может одновременно обрабатывать запросы *SELECT* и *INSERT*. Для каждого типа запросов требуется отдельная внешняя таблица

Глава 4

Синтаксис

Шаблон строки, передаваемой **ADB** при создании внешней таблицы:

```
CREATE [ READABLE | WRITABLE ] EXTERNAL TABLE <table_name> (
    { <column_name> <data_type> [, ...] | LIKE <other_table> }
)
LOCATION (
    'pxf://<full_external_table_name>?<pxf_parameters><jdbc_required_parameters><jdbc_login_
↪parameters><plugin_parameters>'
)
FORMAT 'CUSTOM' (FORMATTER={'pxfwritable_import' | 'pxfwritable_export'})
```

Параметры *<pxf_parameters>*:

```
{
PROFILE=JDBC
|
FRAGMENTER=org.apache.hawq.pxf.plugins.jdbc.JdbcPartitionFragmenter
&ACCESSOR=org.apache.hawq.pxf.plugins.jdbc.JdbcAccessor
&RESOLVER=org.apache.hawq.pxf.plugins.jdbc.JdbcResolver
}
```

Параметры *<jdbc_required_parameters>*:

```
&JDBC_DRIVER=<external_database_jdbc_driver>
&DB_URL=<external_database_url>
```

Параметры *<jdbc_login_parameters>* опциональны. Однако, если они есть, оба параметра должны быть указаны:

```
&USER=<external_database_login>
&PASS=<external_database_password>
```

Параметры *<plugin_parameters>* опциональны:

```
[
&BATCH_SIZE=<batch_size>
]
[
&POOL_SIZE=<pool_size>
]
[
```

```
&PARTITION_BY=<column>:<column_type>
&RANGE=<start_value>:<end_value>
[&INTERVAL=<value>[:<unit>]]
]
[
&PRE_SQL=<string>[&STOP_IF_PRE_FAILS=<string>]
]
```

Описание параметра *BATCH_SIZE* приведено в главе Пакетный режим.

Описание параметра *POOL_SIZE* приведено в главе Пул потоков.

Описание параметров *PARTITION_BY*, *RANGE* и *INTERVAL* приведено в главе Партицирование.

Описание параметров *PRE_SQL* и *STOP_IF_PRE_FAILS* приведено в главе Предварительный запрос.

Глава 5

Запросы SELECT

Плагин **PXF JDBC** позволяет выполнять запросы *SELECT* во внешних таблицах.

Для выполнения запросов *SELECT* необходимо в **PXF** создать таблицу *EXTERNAL READABLE TABLE* или просто *EXTERNAL TABLE* с форматом *FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import')*.

Параметры *BATCH_SIZE* и *POOL_SIZE* не используются в таких таблицах. Однако, если эти параметры присутствуют, их значения проверяются на корректность (параметр должен быть целым числом *integer*).

Глава 6

Запросы INSERT

Плагин **PXF JDBC** позволяет выполнять запросы *INSERT* во внешних таблицах.

Important: Плагин не гарантирует согласованность запросов *INSERT*. В этих целях рекомендуется использовать промежуточную (staging-) таблицу во внешней базе данных

Для выполнения запросов *INSERT* необходимо в **PXF** создать таблицу *EXTERNAL WRITABLE TABLE* с форматом *FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export')*.

Параметры *PARTITION_BY*, *RANGE* и *INTERVAL* в данных таблицах игнорируются.

6.1 Пакетный режим

Запросы *INSERT* могут быть пакетированы, что значительно увеличивает производительность в случае, если внешняя база данных поддерживает пакетный режим.

Для включения пакетной обработки необходимо создать внешнюю таблицу с параметром *BATCH_SIZE*, установленным в одно из значений:

- *integer* > *1* – используется пакет заданного размера;
- *integer* < *0* – используется безразмерный пакет (все записи отправляются одним огромным JDBC-запросом). Настройка может вызвать ошибки, так как каждая база данных имеет собственный лимит на размер запросов JDBC;
- *0* или *1* – пакетирование не используется.

Пакетная обработка должна поддерживаться драйвером **JDBC** внешней базы данных. В случае если драйвер не поддерживает пакетный режим, то плагин **PXF** выполняет запрос *INSERT* так, как если бы параметр *BATCH_SIZE* отсутствовал, а в журналы **PXF** помещается информационное сообщение.

По умолчанию пакетная обработка не используется (параметр *BATCH_SIZE* отсутствует).

Глава 7

Пул потоков

Запросы *INSERT* могут обрабатываться несколькими потоками, что значительно увеличивает производительность в случае, если внешняя база данных может эффективно работать с несколькими соединениями одновременно.

Вместе с пулом потоков рекомендуется использовать пакетирование. Тогда каждый поток получает данные из одной (целой) партии и обрабатывает ее. А если пул потоков используется без пакетной обработки, то каждый поток в пуле получает ровно одну запись; как правило, это занимает гораздо больше времени, чем обычный однопоточный *INSERT*.

Important: Если при выполнении операции любой поток из пула не смог выполнить какой-либо запрос, пользователь получает сообщение об ошибке. Тем не менее, как и в случае использования пакетной обработки, при возникновении ошибки часть записей может оказаться во внешней базе данных. Иными словами, операция вставки при использовании пула потоков не является атомарной даже в том случае, когда внешняя база данных поддерживает транзакции

Для включения пула потоков необходимо создать внешнюю таблицу с параметром *POOL_SIZE* с одним из следующих значений:

- *integer* $>$ *1* – пул потоков состоит из заданного количества потоков;
- *integer* $<$ *1* – количество потоков в пуле равно количеству процессоров в системе;
- *integer* $=$ *1* – не использовать пул потоков.

Если параметр не указан, пул потоков не используется.

Глава 8

Предварительный запрос

Перед выполнением любого запроса плагин **PXF JDBC** может выполнить произвольный SQL-код. Для этого следует передать его как обычную строку (включая символ “*;*” при необходимости) в качестве параметра *PRE_SQL* при создании внешней таблицы **ADB**.

Если исполнение “основного” запроса *INSERT* или *SELECT* не требуется, то в случае ошибки при выполнении произвольного кода в опции *PRE_SQL* можно установить параметр *STOP_IF_PRE_FAILS* равным любому значению. Тогда плагин **PXF JDBC** не исполняет “основной” запрос в случаях, когда произвольный код приводит к возникновению исключения во внешнем источнике данных. Если данный параметр не задан, плагин **PXF JDBC** продолжает выполнение “основного” запроса, даже если предварительный запрос завершается неудачно, и в журналы **PXF** помещается предупреждение.

Глава 9

Партиционирование

Плагин **PXF JDBC** поддерживает одновременный доступ к внешней таблице на чтение из нескольких узлов **PXF**. Такая функция называется партиционированием.

9.1 Синтаксис

Для активации партиционирования необходимо использовать следующие параметры *<plugin_parameters>*:

```
&PARTITION_BY=<column>:<column_type>
&RANGE=<start_value>:<end_value>
[&INTERVAL=<value>[:<unit>]]
```

Параметр *PARTITION_BY* указывает, какой столбец используется в качестве столбца партиционирования (можно использовать только один столбец):

- *<column>* – имя столбца, по которому производится партиционирование;
- *<column_type>* – тип данных столбца *<column>*. Поддерживаемые типы: *INT*, *DATE* и *ENUM*.

Параметр *RANGE* указывает диапазон данных для запроса:

- Если тип партиционирования – *ENUM*, параметр *RANGE* должен быть списком значений, каждое из которых формирует собственную партицию;
- Если тип партиционирования – *INT* или *DATE*, параметр *RANGE* должен быть конечным замкнутым слева диапазоном (*... >= start_value AND ... < end_value*);
- Для типа партиционирования *DATE* формат даты должен быть *yyyy-MM-dd*.

Параметр *INTERVAL* необходим для партиционирований типа *INT* и *DATE* и игнорируется в случае, если *<column_type>* является *ENUM*:

- *<value>* – размер каждой партиции (если данное значение не кратно параметру *RANGE*, последняя партиция будет меньшего размера);
- *<unit>* должен быть указан, если тип *<column_type>* является *DATE*, и допустимые значения – *year*, *month* и *day*. Параметр игнорируется для других типов данных столбца партиции.

Примеры выражений, задающих параметры партиционирования:

```
&PARTITION_BY=id:int&RANGE=42:142&INTERVAL=2
```

```
&PARTITION_BY=createdate:date&RANGE=2008-01-01:2010-01-01&INTERVAL=1:month
```

```
&PARTITION_BY=grade:enum&RANGE=excellent:good:general:bad
```

9.2 Механизм работы

При активированном партиционировании запрос *SELECT* разбивается на набор отдельных запросов, каждый из которых называется партицией (или фрагментом). Все фрагменты одновременно обрабатываются отдельными инстансами **PXF**. В случае превышения числа фрагментов над числом инстансов **PXF**, некоторые инстансы обрабатывают более одного фрагмента (по очереди); а в случае только одного инстанса **PXF**, все фрагменты обрабатываются им. Распределение фрагментов по инстансам происходит в процессе исполнения запроса.

Дополнительно к каждому фрагменту автоматически добавляются параметры запроса (с выражением *WHERE*) для гарантии, что каждый набор данных извлекается из внешнего источника ровно один раз.

9.3 Пример партиционирования

В качестве примера дана следующая таблица **MySQL**:

```
CREATE TABLE sales (
    id int primary key,
    cdate date,
    amt decimal(10,2),
    grade varchar(30)
)
```

И внешняя таблица **ADB**:

```
CREATE EXTERNAL TABLE sales(
    id integer,
    cdate date,
    amt float8,
    grade text
)
LOCATION ('pxf://sales?PROFILE=JDBC&JDBC_DRIVER=com.mysql.jdbc.Driver&DB_URL=jdbc:mysql://192.168.
↪200.6:3306/demodb&PARTITION_BY=cdate:date&RANGE=2008-01-01:2010-01-01&INTERVAL=1:year')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

На основе приведенных таблиц плагин **PXF JDBC** генерирует два фрагмента для запроса “SELECT * FROM sales”. Затем **HAWQ** назначает каждый из них отдельному узлу **PXF**. Каждый узел выполняет запрос *SELECT*, первый получает записи с значениями *cdate* за *2008* год, а второй – за *2009* год. Затем каждый узел **PXF** отправляет свои результаты обратно в **HAWQ**, где они “сцепляются” и возвращаются в **ADB**.

Глава 10

Примеры

В следующем примере показано получение доступа к таблице **MySQL** через **JDBC**.

Предполагая, что инстанс **MySQL** доступен на *192.168.200.6:3306*, создается таблица в **MySQL**:

```
use demodb;
create table myclass(
    id int(4) not null primary key,
    name varchar(20) not null,
    degree double(16,2)
);
```

Затем некоторые данные вставляются в таблицу **MySQL**:

```
insert into myclass values(1, 'tom', 90);
insert into myclass values(2, 'john', 94);
insert into myclass values(3, 'simon', 79);
```

JDBC-файлы драйвера **MySQL** (**JAR**) копируются в */usr/lib/pxf* на всех узлах кластера, а в */etc/pxf/conf/pxf-public.classpath* добавляется строка:

```
/usr/lib/pxf/mysql-connector-java-*.jar
```

После этого все сегменты **PXF** перезапускаются. В **ADB** создается внешняя таблица:

```
CREATE EXTERNAL TABLE myclass(
    id integer,
    name text,
    degree float8
)
LOCATION (
    'pxf://localhost:51200/demodb.myclass?PROFILE=JDBC&JDBC_DRIVER=com.mysql.jdbc.Driver&DB_
↪URL=jdbc:mysql://192.168.200.6:3306/demodb&USER=root&PASS=root'
)
FORMAT 'CUSTOM' (
    FORMATTER='pxfwritable_import'
);
```

Наконец, к последней таблице выполняется запрос, который возвращает ожидаемые результаты:

```
SELECT * FROM myclass;
SELECT id, name FROM myclass WHERE id = 2;
```

Глава 11

Лучшие практики по использованию PXF JDBC коннектора

1. При запросах *INSERT* рекомендуется всегда использовать промежуточные (staging-) таблицы во внешней базе данных;
2. Для большинства СУБД оптимальное значение параметра *BATCH_SIZE* варьируется от *2000* до *30000* строк.