Arenadata™ Database

Версия - v6.18.2_arenadata30

Интеграция кластера ADB и Kafka

2021 Arenadata™

Оглавление

1 ADB to Kafka (kafka-connector) 3
- 1.1 Совместимость . . . 3
- 1.2 Установка с помощью ADCM . . . 3
- 1.3 Установка из rpm-пакетов . . . 3

2 PXF плагин 4
- 2.1 Установка . . . 4
- 2.2 При необходимости установки/обновления пользовательского pxf коннектора . . . 4
- 2.3 Запись информации в топик Kafka . . . 5
- 2.4 Запись данных в топик Kafka при помощи профиль сервера `SERVER=<server_name>` . . . 6
- 2.5 Преобразование типов GPDB → AVRO . . . 7

3 Kafka to ADB (kadb-fdw) 8
- 3.1 Совместимость . . . 8
- 3.2 Установка с помощью ADCM . . . 8
- 3.3 SQL-интерфейс . . . 8
- 3.4 Примеры использования . . . 19
- 3.5 Замечания по применению . . . 21

В документе приведено краткое описание интеграции кластера Arenadata DB и кластера брокера сообщений Kafka.

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

Глава 1

ADB to Kafka (kafka-connector)

kafka-connector - это коннектор к Arenadata DB, позволяющий производить транзакционную загрузку данных из ADB в кластер брокера сообщений Kafka.

1.1 Совместимость

Коннектор совместим с Apache Kafka версий 1.0.0 и выше

1.2 Установка с помощью ADCM

Для установки коннектора с помощью **ADCM** требуется инсталлировать сервис *PXF* на все сегментные ноды кластера, а также в списке сервисов выбрать сервис *ADB to Kafka*. При этом необходимые пакеты и файлы автоматически устанавливаются на машины кластера.

1.3 Установка из rpm-пакетов

Установка из rpm-пакетов предполагает, что в кластере *ADB* установлен сервис *PXF*.

Для установки коннектора из rpm-пакетов необходимо:

1. Установить пакет *kafka-connector* на всех хостах кластера *ADB*, где установлен сервис *PXF*;
2. Добавить в файл */etc/pxf/conf/pxf-profiles-default.xml* на каждом хосте кластера следующую секцию:

```
<profile>
    <name>kafka</name>
    <description>A profile for export data into Apache Kafka</description>
    <plugins>
        <accessor>org.greenplum.pxf.plugins.kafka.KafkaAccessor</accessor>
        <resolver>org.greenplum.pxf.plugins.kafka.KafkaResolver</resolver>
    </plugins>
    <optionMappings>
        <mapping option="BOOTSTRAP_SERVERS" property="kafka.bootstrap.servers"/>
        <mapping option="BATCH_SIZE" property="kafka.batch.size"/>
    </optionMappings>
</profile>
```

3. Перезапустить сервис *PXF* на всех хостах кластера.

Глава 2

PXF плагин

PXF плагин позволяет передавать данные из *ADB* в *Kafka*.

2.1 Установка

Для установки плагина, необходимо иметь:

- Рабочий кластер ADB с установленным PXF.
- Рабочий кластер ADS или

2.2 При необходимости установки/обновления пользовательского pxf коннектора

Плагине имеется папка `pxf-kafka/build/libs``c jar-файлами (``pxf-kafka.jar`, `avro-1.9.2.jar`, `kafka-clients-2.5.0.jar`). Их необходимо скопировать на каждый сегмент хост *ADB* и удалить старые:

```
cp pxf-kafka.jar /usr/lib/pxf/lib/
chown pxf:pxf /usr/lib/pxf/lib/pxf-kafka.jar

rm /usr/lib/pxf/lib/shared/avro-1.7.7.jar
cp avro-1.9.2.jar /usr/lib/pxf/lib/shared/
chown pxf:pxf /usr/lib/pxf/lib/shared/avro-1.9.2.jar

cp kafka-clients-2.5.0.jar /usr/lib/pxf/lib/shared/
chown pxf:pxf /usr/lib/pxf/lib/shared/kafka-clients-2.5.0.jar
```

Необходимо скопировать текст в тэгах `<profiles>...</profiles>` из файла `pxf-kafka/env/pxf-profiles.xml` и добавить его в файл `/var/lib/pxf/conf/pxf-profiles.xml` на каждом сегмент хосте *ADB*. Пример файла *pxf-profiles.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>

<profiles>
    <profile>
        <name>kafka</name>
        <description>A profile for export data into Apache Kafka</description>
        <plugins>
            <accessor>org.greenplum.pxf.plugins.kafka.KafkaAccessor</accessor>
```

```
            <resolver>org.greenplum.pxf.plugins.kafka.KafkaResolver</resolver>
        </plugins>
        <optionMappings>
            <mapping option="BOOTSTRAP_SERVERS" property="kafka.bootstrap.servers"/>
            <mapping option="BATCH_SIZE" property="kafka.batch.size"/>
            <mapping option="TOPIC_AUTO_CREATE_FLAG" property="kafka.topic.auto.create"/>
            <mapping option="AVRO_DEFAULT_DECIMAL_PRECISION" property="avro.decimal.default.precision" />
            <mapping option="AVRO_DEFAULT_DECIMAL_SCALE" property="avro.decimal.default.scale" />
        </optionMappings>
    </profile>
</profiles>
```

После чего необходимо перезапустить сервис pxf на каждом сегмент хосте:

```
systemctl restart pxf
```

2.3 Запись информации в топик Kafka

Коннектор *PXF Kafka* с профилем `kafka` поддерживает запись данных в Kafka. Когда вы создаете внешнюю таблицу для записи при помощи коннектора *PXF Kafka*, вы указываете имя топика kafka. Когда вы вводите данные в эту таблицу, запись об этих данных появляется в указанном топике.

Используйте следующий синтаксис для создания внешней таблицы с профилем `kafka`:

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

Ключевые параметры используемые в команде CREATE EXTERNAL TABLE описаны в таблице ниже:

Профиль `kafka` поддерживает следующие пользовательские настройки:

Настройка	Описание значения	Обязателен
BOOTSTRAP_SERVERS	Список брокеров Kafka через запятую, каждый из которых является хостом или host:port строкой.	Да
BATCH_SIZE	Определяет сколько строк должно складываться в одно сообщение Avro.	Нет
TOPIC_AUTO_CREATE_FLAG	Указывает топикам создаваться автоматически, когда в них записывают данные. Топик будет создан с 1 партицией с фактором репликации =1. Значение по умолчанию: true.	Нет
AVRO_DEFAULT_DECIMAL_PRECISION	Максимальное количество знаков в числе. Должно быть положительным числом выше нуля. Значение по-умолчанию: 38.	Нет
AVRO_DEFAULT_DECIMAL_SCALE	Количество знаков после запятой для чисел. Например, число 123.45 имеет DECIMAL_PRECISION 5 и DECIMAL_SCALE 2. Должно быть неотрицательным числом меньше или равным предыдущему параметру. Значение по-умолчанию: 18.	Нет

2.3.1 Пример: запись данных в топик Kafka

```
DROP EXTERNAL TABLE IF EXISTS kafka_tbl;

CREATE WRITABLE EXTERNAL TABLE kafka_tbl (a TEXT, b TEXT, c TEXT)
  LOCATION ('pxf://data_from_gp?PROFILE=kafka&BOOTSTRAP_SERVERS=10.92.8.43:9092,10.92.8.38:9092&BATCH_
↪SIZE=10')
  FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

```
INSERT INTO kafka_tbl VALUES ('a', 'b,c', 'd'), ('x', 'y', 'z');
```

2.4 Запись данных в топик Kafka при помощи профиль сервера SERVER=<server_name>

Профиль сервера SERVER=<server_name> позволяет вам использовать конфигурационный файл для параметров, вместо использования их в секции LOCATION внешней таблицы. Чтобы настроить профиль сервера, необходимо:

1. Создать папку в разделе /var/lib/pxf/servers/ с подходящим названием. Например, следующая команда создаст папку для Kafka в среде разработки:

   ```
   mkdir /var/lib/pxf/servers/kafka_dev
   ```

2. Скопировать шаблон конфигурационного файла из pxf-kafka/env/kafka-site.xml в папку /var/lib/pxf/servers/kafka_dev. Пример конфигурационного файла kafka-site.xml:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <configuration>
       <property>
           <name>kafka.bootstrap.servers</name>
           <value>10.92.8.38:9092</value>
       </property>
       <property>
           <name>kafka.batch.size</name>
           <value>1</value>
       </property>
       <property>
           <name>kafka.topic.auto.create</name>
           <value>true</value>
       </property>
       <property>
           <name>avro.decimal.default.precision</name>
           <value>38</value>
       </property>
       <property>
           <name>avro.decimal.default.scale</name>
           <value>18</value>
       </property>
   </configuration>
   ```

3. Отредактировать файл kafka-site.xml и скопировать его на каждый сегмент хост в ту же папку. Все доступные параметры перечислены в таблице ниже (как можно заметить, параметры повторяют параметры профиля):

Настройка	Описание значения	Обязателен
kafka.bootstrap.servers	Список брокеров Kafka через запятую, каждый из которых является хостом или host:port строкой.	Да
kafka.batch.size	Определяет сколько строк должно складываться в одно сообщение Avro.	Нет
kafka.topic.autocreate	Позволяет топикам создаваться автоматически, когда в них записывают данные. Топик будет создан с 1 партицией с фактором репликации =1. Значение по умолчанию: true.	Нет
avro.decimal.defaultprecision	Максимальное количество знаков в числе. Должно быть положительным числом выше нуля. Значение по-умолчанию: 38.	Нет
avro.decimal.defaultscale	Количество знаков после запятой для чисел. Например, число 123.45 имеет DECIMAL_PRECISION 5 и DECIMAL_SCALE 2. Должно быть неотрицательным числом меньше или равным предыдущему параметру. Значение по-умолчанию: 18.	Нет

2.4.1 Пример: Запись данных в топик Kafka при помощи профиля сервера SERVER=<server_name>

```
DROP EXTERNAL TABLE IF EXISTS kafka_tbl_dev;

CREATE WRITABLE EXTERNAL TABLE kafka_tbl_dev (a TEXT, b TEXT, c TEXT)
  LOCATION ('pxf://data_from_gp?ACCESSOR=org.greenplum.pxf.plugins.kafka.KafkaAccessor&RESOLVER=org.
↪greenplum.pxf.plugins.kafka.KafkaResolver&kafka.bootstrap.servers=10.92.8.38:9092')
  FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');

INSERT INTO kafka_tbl_dev VALUES ('a', 'b,c', 'd'), ('x', 'y', 'z');
```

2.5 Преобразование типов GPDB → AVRO

Тип PXF	Примитивный тип AVRO	Логический тип AVRO
BOOLEAN	BOOLEAN	
TEXT	STRING	
VARCHAR	STRING	
TIMESTAMP	LONG	timestamp-micros
BIGINT	LONG	
TIME	LONG	time-micros
NUMERIC	DOUBLE	
FLOAT8	DOUBLE	
REAL	FLOAT	
SMALLINT	INT	
INTEGER	INT	
DATE	INT	date

Глава 3

Kafka to ADB (kadb-fdw)

kadb-fdw - это расширение для ADB/GPDB, позволяющее производить транзакционную загрузку данных из кластера брокера сообщений Kafka.

Особенности:

- AVRO десереализация;
- Хранение смещений Kafka вне кластера Kafka, на стороне потребителя;
- Поддержка транзакций ADB/GPDB;
- Поддержка Kerberos-аутентификации.

3.1 Совместимость

Коннектор совместим с Apache Kafka версий 1.0.0 и выше

3.2 Установка с помощью ADCM

Для установки коннектора с помощью **ADCM** требуется инсталлировать сервис *PXF* на все сегментные ноды кластера, а также в списке сервисов выбрать сервис *Kafka to ADB*. При этом необходимые пакеты и файлы автоматически устанавливаются на машины кластера.

Important: Процесс установки включает в себя выполнение следующей команды: **RESET client_min_messages**

3.3 SQL-интерфейс

3.3.1 Пример

```
-- Create a SERVER
DROP SERVER IF EXISTS ka_server;
CREATE SERVER ka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'localhost:9092'
);
```

```
-- Create a FOREIGN TABLE
DROP FOREIGN TABLE IF EXISTS ka_table;
CREATE FOREIGN TABLE ka_table(field1 INT, field2 TEXT)
SERVER ka_server
OPTIONS (
    format 'avro',  -- Data serialization format
    k_topic 'my_topic',  -- Kafka topic
    k_consumer_group 'my_consumer_group',  -- Kafka consumer group
    k_seg_batch '100',  -- Limit on the number of Kafka messages retrieved by each GPDB segment
    k_timeout_ms '1000',  -- Kafka response timeout
    k_initial_offset '42'  -- Initial Kafka offset (for new or unknown partitions)
);

-- Issue a SELECT query as usual
SELECT * FROM ka_table;
```

kadb_fdw предоставляет пользователю следующие интерфейсы взаимодействия через SQL:

- FOREIGN TABLE OPTIONS. См. CREATE FOREIGN TABLE, ALTER FOREIGN TABLE документацию. Дополнительные опции **kadb_fdw** представлены в разделе “Опции внешних таблиц”
- Таблица смещений
- Набор kadb функций

3.3.2 Таблица смещений

В ходе установки расширения создается служебная схема `kadb`, содержащая таблицу `kadb.offsets`. В данную таблицу помещаются пары соответствий партиция-смещение для любой когда-либо создававшейся в текущей базе внешней таблицы (FOREIGN TABLE).

Таблицы идентифицируются по значению OID, который можно узнать, выполнив следующую команду:

```
SELECT 'schema.table'::regclass::oid;
```

При выполнении SELECT-запроса к внешней таблице чтение сообщений из Kafka производится, начиная со смещения, указанного для данной внешней таблицы в таблице `kadb.offsets`.

Смещения можно изменять при помощи обычного SQL-запроса к таблице `kadb.offsets`.

Для новых партиций, записей о которых нет в таблице `kadb.offsets`, начальное смещение по умолчанию устанавливается равным 0 (значение может быть изменено в параметре *k_initial_offset*).

После успешного выполнения SELECT-запроса к внешней таблице смещение обновляется в соответствии со значениями, полученными от Kafka. Например, если последнее прочитанное сообщение из некоторой партиции имело смещение 84, значение смещения для данной партиции в таблице `kadb.offsets` будет равным 85.

3.3.3 Опции внешних таблиц

Для сервера (SERVER) и внешней таблицы (FOREIGN TABLE) возможно указание опций в виде пары ключ-значение.

Опции, определенные для сервера и внешней таблицы, не отличаются друг от друга. Иными словами, не имеет значения, для какого объекта они были указаны. Однако опции, определенные для внешней таблицы, более приоритетны, чем опции сервера (в случае, если для обоих объектов была указана одинаковая опция).

Поддерживаемые опции:

librdkafka

Параметры для librdkafka можно установить через FOREIGN TABLE OPTIONS.

ТДля того чтоб установить конфигурационный параметр для librdkafka, добавь OPTION с добавлением к имени знака решетка (#). Пример создания SERVER:

```
CREATE SERVER my_kafka
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    "#bootstrap.servers" 'localhost:9092',
    "#client.id" 'My GPDB instance'
);
```

Важно чтоб OPTION имена, содержащие знаки решетка (#) или точка (.) были обрамлены в двойные кавычки (“).

Кофигурационные параметры для librdkafka имеют приоритет над их алиасами (см. ниже).

Нижеперечисленные параметры для librdkafka запрещено устанавливать:

- enable.auto.commit
- enable.partition.eof
- plugin.library.paths
- interceptors
- все параметры заканчивающиеся на `_cb`

k_brokers

Обязательна.

Список брокеров Kafka, разделенный запятыми, где каждый элемент — это строка вида <host> или <host>:<port>.

k_topic

Обязательна.

Идентификатор топика Kafka.

k_consumer_group

Обязательна.

Идентификатор группы потребителей Kafka.

k_seg_batch

Обязательна.
Положительное целое число.

Максимальное количество сообщений Kafka, запрашиваемое каждым сегментом кластера ADB/GPDB.

При выполнении запроса с условием LIMIT сообщения в Kafka продолжают запрашиваться батчами. В результате смещения в *таблице смещений* устанавливается на основании *последнего* полученного сообщения для каждой партиции, даже если данные не содержались в результатах запроса.

При достижении конца партиции Kafka отправляет специальное сообщение. Таким образом, если значение `k_seg_batch` меньше, чем число партиций, определенное для одного сегмента ADB/GPDB и `k_seg_batch` этих партиций было полностью прочитано, может возникнуть ситуация, при которой не будет прочитано никаких полезных данных. По этой причине опция `k_seg_batch` *должна быть больше, чем максимальное число партиций определенных для одного сегмента.*

k_timeout_ms

Обязательна.
Неотрицательное целое число.

Время ожидания выполнения запроса к Kafka в миллисекундах. За это время Kafka-клиентом будут извлечены и представлены в качестве результата SELECT-запроса только сообщения, доступные в Kafka в этот период времени.

SELECT-запрос может выполняться быстрее, если в топике Kafka достаточно сообщений.

Реальное время выполнения SELECT-запроса может быть значительно больше. Для оценки максимального значения используйте следующее выражение:

Более всего на длительность запроса влияют партиции Kafka, в которых недостаточно сообщений: коннектор ожидает появления в этих партициях новых сообщений в течение *k_timeout_ms* миллесекунд.

На некоторых этапах выполнения SELECT-запроса его принудительная отмена может быть невозможна до окончания периода k_timeout_ms.

format

Обязательна.
Одно из предопределенных значений (без учета регистра).

Формат сереализованных данных:

- avro
- csv
- text

k_latency_ms

Неотрицательное целое число. По умолчанию 2000.

Таймаут запросов метаданных к Kafka.

Значение этого параметра следует установить в максимально ожидаемое время отклика (по всем сегментам ADB) любого запроса к метаданным к Kafka. Самый длительный запрос к метаданным производится от `kadb.offsets_to_committed(OID)` - эта функция может быть использована, чтоб определить разумное значение пераметра.

Если таймаут превышен, запрос к метаданным упадет с ошибкой. Несколько запросов к метаданным происходят, когда выполняется SELECT; таким образом, падение может произойти, если параметр установлен в сравнительно низкое значение.

k_initial_offset

Неотрицательное целое число. Значение по умолчанию: 0.

Смещение, которое следует использовать для партиций, записей о которых нет в *таблице смещений*.

k_automatic_offsets

Булевое значение (true, false). По умолчанию true.

Позволяет **kadb_fdw** выполнять следующее:

- **Немедленно перед каждым SELECT из FOREIGN TABLE:**
  - Добавить партицию, которая присутствует в Kafka, но отсутствует в таблице смещений, к набору партиций, откуда вычитывать данные;
  - Автоматически увеличивать стартовое смещение любой партиции к самомоу раннему смещению, доступному в Kafka. Если такое увеличение происходит, то формируется NOTICE.
- **Немедленно после каждого SELECT из FOREIGN TABLE:**
  - Обновить таблицу смещений, добавляя партиции (с помощью INSERT запроса), которые присутствуют в Kafka, и отсутствуют в таблице смещений.

Если выставлен в `false`, вызывается ERROR, когда наименьшее смещение сообщения, присутствующего в Kafka больше чем смещение в таблице смещений (для любой партиции).

Важно: присутствие партиций в Kafka, и их отсутствие в таблице смещений не видно пользователямесли используется CURSOR: INSERT новых записей в таблицу смещений происходит после SELECT, тогда как CURSOR постоянно в прогрессе.

После успешного SELECT, смещения в таблице смещений изменяются независимо от значения этого параметра, отражая количество сообщений, прочитанных из Kafka.

k_security_protocol

Обязательна, если используется Kerberos-аутентификация.

Протокол безопасности, который необходимо использовать для подключения к Kafka. В данный момент доступен только протокол sasl_plaintext.

avro_schema

AVRO-схема, которую необходимо использовать. Десериализованная AVRO схема представляет из себя JSON. Получаемые сообщения десериализуются двумя способами:

- Если присутствует опция `avro_schema`, используется указанная схема (сообщение так же должно быть представлено в OCF формате)
- В противном случае, схема извлекается из полученного сообщения в OCF формате.

Important: Пользовательская схема не может быть валидирована. Если реальная схема не соответствует указанной, десериализация завершается с ошибкой *ERROR: invalid memory alloc request size*. По этой причине опция `avro_schema` должна использоваться только в целях повышения производительности и только после тщательного изучения.

csv_quote

Одиночный символ, представимый одним байтом в текущей кодировке.
Значение по умолчанию: “”“.

Символ, используемый в качестве кавычек при парсинге CSV.

csv_delimeter

Одиночный символ, представимый одним байтом в текущей кодировке.
Значение по умолчанию: “,“.

Символ, используемый в качестве разделителя полей при парсинге CSV.

csv_null

Строка, используемая в качестве значения `NULL` в CSV.

По умолчанию пустая строка интерпретируется как NULL.

csv_ignore_header

Булево значение.
Значение по умолчанию: false.

Игнорировать первую строку каждого сообщения при парсинге.

csv_attribute_trim_whitespace

Булево значение.
Значение по умолчанию: true.

Удалять пробелы в начале и конце строки для каждого атрибута (поля) записи.

3.3.4 Набор kadb функций

Несколько функций предоставляются `kadb_fdw` для синхронизации смещений в Kafka с находящимися в таблицах смещений.

Все функции находятся в схеме `kadb`

Важно: * Функции не предоставляют транзакционную гарантию для Kafka. Это значит, что невозможно делать предположения об изменениях смещения в Kafka до или после того, как функция будет применена, даже если она объединена с `SELECT` (из `FOREIGN TABLE`) в единой SQL транзакции; * Некоторые функции **не атомарны**. Это значит, что они не создают снапшота всех партиций в определенный момент времени; вместо этого их результат получается из каждой партиции независимо, в (слегка) разные моменты времени.

kadb.commit_offsets(OID)

Параметры:

- OID внешней таблицы

Применяет смещение из таблицы смещений (для указанного `FOREIGN TABLE` OID) к Kafka.

Этот метод **атомарен**.

kadb.load_partitions(OID)

Параметры:

- OID внешней таблицы

Результат:

- `floid`: такой же как указанный OID внешней таблицы
- `prt`: идентификатор партиции
- `off`: k_initial_offset

Загружает список партиций, существующих в Kafka.

Этот метод **не атомарен**.

kadb.partitions_obtain(OID)

Параметры:

- OID внешней таблицы

Добавляет партиции, возвращенные командой `kadb.load_parittions(OID)` к таблице смещений. Добавляются только новые партиции; существующие остаются без изменений.

Этот метод **не атомарен**.

kadb.partitions_clean(OID)

Параметры:

- OID внешней таблицы

Удаляет все записи из таблицы смещений (для указанного OID внешней таблицы), которые *отсутствуют* в Kafka.

Этот метод **не атомарен**.

kadb.partitions_reset(OID)

Параметры:

- OID внешней таблицы

Удаляет все записи из таблицы смещений (для указанного OID внешней таблицы), и добавляет записи, возвращенные командой `kadb.load_parittions(OID)`.

Этот метод **не атомарен**.

kadb.load_offsets_at_timestamp(OID, BIGINT)

Параметры:

- OID внешней таблицы
- Временная метка: прошедшие миллисекунды с UNIX Epoch (UTC)

Результат:

- `floid`: такой же как указанный OID внешней таблицы

- `ort`: идентификатор партиции
- `off`: результат

Загружает самые ранние смещения, существующие в Kafka, чьи временные метки больше или равны указанной в команде (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений).

Этот метод **атомарен**.

kadb.offsets_to_timestamp(OID, BIGINT)

Параметры:

- OID внешней таблицы
- Временная метка: прошедшие миллисекунды с UNIX Epoch (UTC)

Изменяет смещения в таблице смещений (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений) на самые ранние смещения, существующие в Kafka, чьи временные метки больше или равны указанной в команде.

Этот метод **атомарен**.

kadb.load_offsets_earliest(OID)

Параметры:

- OID внешней таблицы

Результат:

- `floid`: такой же как указанный OID внешней таблицы
- `ort`: идентификатор партиции
- `off`: результат

Загружает самые ранние смещения, существующие в Kafka (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений).

Этот метод **не атомарен**.

kadb.offsets_to_earliest(OID)

Параметры:

- OID внешней таблицы

Меняет смещения в таблице смещений (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений) на самые ранние смещения, существующие в Kafka.

Этот метод **не атомарен**.

kadb.load_offsets_latest(OID)

Параметры:

- OID внешней таблицы

Результат:

- `floid`: такой же как указанный OID внешней таблицы
- `ort`: идентификатор партиции

- `off`: результат

Загружает самые поздние смещения, существующие в Kafka (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений).

Этот метод **не атомарен**.

kadb.offsets_to_latest(OID)

Параметры:

- **OID внешней таблицы**
  - Чтобы получить OID из названия таблицы можно использовать `'table_schema.table_name'::regclass::oid`

Меняет смещения в таблице смещений (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений) на самые поздние смещения, существующие в Kafka.

В результате, команды `SELECT` из указанной внешней таблицы возвращают только сообщения, введенные в Kafka после использования этой команды.

Этот метод **не атомарен**.

kadb.load_offsets_commited(OID)

Параметры:

- OID внешней таблицы

Результат:

- `floid`: такой же как указанный OID внешней таблицы
- `ort`: идентификатор партиции
- `off`: результат

Загружает самые поздние примененные смещения, существующие в Kafka (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений).

Этот метод **атомарен**.

kadb.offsets_to_commited(OID)

Параметры:

- OID внешней таблицы

Меняет смещения в таблице смещений (для указанного OID внешней таблицы, и только для партиций уже существующих в таблице смещений) на самые поздние примененные смещения, существующие в Kafka.

Этот метод **не атомарен**.

3.3.5 Десериализация

В настоящее время расширение *kafka-fdw* поддерживает сообщения Kafka, сериализованные при помощи одного из следующих форматов:

- AVRO OCF
- CSV
- text

Метод десериалиализации должен быть указан в опции *format*.

Вне зависимости от используемого метода десериализуется только сообщение Kafka, ключ сообщения игнорируется.

AVRO

kadb-fdw предоставляет поддержку формата сериализации AVRO OCF с некоторыми ограничениями.

Схема должна содержать только примитивные типы данных или объединения одного примитивного типа данных с типом `NULL`. Также поддерживается тип `fixed`, поскольку он расценивается как `bytes`.

Поддерживаются все логические типы данных, указанные в спецификации AVRO.

Определение внешней таблицы ADB/GPDB должно совпадать с настоящей схемой AVRO.

Поддерживаются следующие типы маппинга:

AVRO тип	PostgreSQL тип
`string`	`TEXT`, `BPCHAR`, `VARCHAR`
`string`	Пользовательский тип PostgreSQL (напр. `MONEY`). Преобразование такое же, как для пользовательских текстовых вводов.
`null`	Любой тип PostgeSQL в столбце с ненулевыми рамками.
`int`	`INTEGER`
`long`	`BIGINT`
`float`	`REAL`
`double`	`DOUBLE PRECISION`
`bytes`, `fixed`	`BOOLEAN`
`decimal`	`BYTEA`
`date`	`NUMERIC`
`time-millis`, `time-micros`	`DATE`
`timestamp-millis`	`TIME`
`timestamp-micros`	`TIMESTAMP(N)`, где `N` это 1, 2, или 3
`boolean`	`TIMESTAMP`, `TIMESTAMP(N)`, где `N` это 4 или более
`duration`	`INTERVAL`

Также **порядок** столбцов должен совпадать с порядком полей в схеме AVRO.

Пример схемы:

```
{
   "name": "doc",
   "type": "record",
   "fields": [
      {
         "name": "id",
         "type": "int"
      },
      {
         "name": "text",
         "type": ["string", "null"]
      },
      {
         "name": "issued_on",
         "type": "int",
      },
```

```
  ]
}
```

```
 {
  "name":"doc",
  "type":"record",
  "fields":[
    {
       "name":"d",
       "type":"int",
       "logicalType":"date"
    },
    {
       "name":"t_ms",
       "type":"int",
       "logicalType":"time-millis"
    },
    {
       "name":"t_us",
       "type":"long",
       "logicalType":"time-micros"
    },
    {
       "name":"ts_ms",
       "type":"long",
       "logicalType":"timestamp-millis"
    },
    {
       "name":"ts_us",
       "type":"long",
       "logicalType":"timestamp-micros"
    },
    {
       "name":"dur",
       "type":{
       "name":"dur_fixed",
       "type":"fixed",
       "size":12,
       "logicalType":"duration"
    }
 },
 {
    "name":"dec_1",
    "type":{
     "name":"dec_2_fixed",
     "type":"fixed",
     "size":6,
     "logicalType":"decimal"
  }
 },
 {
   "name":"dec_2",
   "type":{
     "name":"dec_2_fixed",
     "type":"bytes",
     "logicalType":"decimal",
     "precision":14,
     "scale":4
```

```
    }
  }
]
```

}

CSV

Поддержка CSV реализована при помощи libcsv. Таким образом, учитываются все соглашения о формате CSV, устанавливаемые этой библиотекой. Перечень соглашений представлен в данном документе. Спецификация CSV определена в RFC 4180.

Принимая во внимание эти рекомендации, *kadb-fdw* использует следующие правила парсинга CSV:

- Поля (атрибуты) разделяются специальным символом (*разделителем*);
- Строки (записи) разделяются символом новой строки;
- Поле может быть заключено в специальные символы (*кавычки*);
- Поля, содержащие символ разделителя, кавычки или новой строки, должны быть заключены в кавычки;
- Каждый символ *кавычки* должен быть экранирован путем его дублирования (добавления идентичного символа перед ним);
- Пустое поле всегда интерпретируется как значение `NULL`;
- Пустые строки игнорируются;
- Пробелы в начале и конце поля, не заключенного в кавычки, удаляются, если соответствующая *опция* выставлена в значение true.

Значения могут быть конвертированы в любой тип данных PostgreSQL. Используемые правила конвертации аналогичны правилам, применяемым для входных данных *psql*.

text

text - это сериализованный формат для представления данных в сыром виде в Kafka сообщении.

Когда испоьзуется этот формат, *kadb_fdw* действует следующим образом:

- Каждое сообщение предполагает представление одного атрибута (колонки) одной строки из `FOREIGN TABLE`
- Данные парсятся PostgreSQL как текстовые в сводобной форме (user-provided textual data).
- Такой формат требует, чтобы в `FOREIGN TABLE` содержался ровно один атрибут (колонку), которая может быть любого типа PostgreSQL.

Kafka сообщения с пустым содержимым (или нулевой длины) парсятся как `NULL` значения, и также учитываются.

3.4 Примеры использования

3.4.1 Таблица для данных в формате AVRO

```
DROP SERVER IF EXISTS ka_server CASCADE;
CREATE SERVER ka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'localhost:9092'
);
```

```
CREATE FOREIGN TABLE ka_table(
    i INT,
    t TEXT
)
SERVER ka_server
OPTIONS (
    format 'avro',
    k_topic 'my_topic',
    k_consumer_group 'my_consumer_group',
    k_seg_batch '5',
    k_timeout_ms '1000'
);
```

3.4.2 Таблица для данных в формате CSV

```
DROP SERVER IF EXISTS ka_kerberized_server CASCADE;
CREATE SERVER ka_kerberized_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'ke-kafka-sasl.ru-central1.internal:9092',
    "#security.protocol" 'sasl_plaintext',
    "#sasl.kerberos.keytab" '/root/adbkafka.service.keytab',
    "#sasl.kerberos.principal" 'adbkafka'
);

CREATE FOREIGN TABLE ka_table(
    i INT,
    t TEXT
)
SERVER ka_server
OPTIONS (
    format 'avro',
    k_topic 'my_topic',
    k_consumer_group 'my_consumer_group',
    k_seg_batch '5',
    k_timeout_ms '1000'
);
```

3.4.3 Таблица с Kerberos-аутентификацией

```
DROP SERVER IF EXISTS ka_kerberized_server CASCADE;
CREATE SERVER ka_kerberized_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'ke-kafka-sasl.ru-central1.internal:9092',
    "#security.protocol" 'sasl_plaintext',
    "#sasl.kerberos.keytab" '/root/adbkafka.service.keytab',
    "#sasl.kerberos.principal" 'adbkafka'
);

CREATE FOREIGN TABLE ka_table(
    i INT,
    t TEXT
)
SERVER ka_server
```

```
OPTIONS (
    format 'avro',
    k_topic 'my_topic',
    k_consumer_group 'my_consumer_group',
    k_seg_batch '5',
    k_timeout_ms '1000'
);
```

3.5 Замечания по применению

Данный раздел содержит заметки по применению `kadb_fdw`. Он предназначен, чтобы перечислить характерные поведения и ожидаемые результаты.

3.5.1 Одновременные SELECT

`kadb_fdw` использует таблицу смещения при каждом запросе `SELECT` из внешней таблицы. Это одиночная (`DISTRIBUTED REPLICATED`) таблица ADB/GPDB. `kadb_fdw` может послать запросы `INSERT` и `UPDATE` таблице смещений.

Таким образом, пределы одновременных операций влияют на `SELECT` из внешних таблиц `kadb_fdw`.

Обнаружение блокировок

Одновременные транзакции проходящие в GPDB обрабатываются детектором блокировок.

Когда детектор блокировок **отключен**, каждый `UPDATE` требует `ExclusiveLock`, который, по сути, блокирует всю таблицу для операции обновления. Для `kadb_fdw` это означает, что множественные одновременные “SELECT“(из *разных* внешних таблиц) невозможны. Такие операции выполняются последовательно.

Чтобы разрешить множественные одновременные `SELECT``(из *разных* внешних таблиц), **включите** детектор блокировок. При работающем детекторе каждый ``SELECT` требует только `RowExclusiveLock`, таким образом разрешая одновременные запросы `UPDATE` к внешним таблицам.

Одновременные `SELECT` из единой внешней таблицы невозможны. В некоторых обстоятельствах такие операции могут выполниться корректно, но это не гарантированно.

Чтобы включить детектор блокировок, установите параметр конфигурации GPDB `gp_enable_global_deadlock_detector` как `on`:

```
gpconfig -c gp_enable_global_deadlock_detector -v on
```

Распределение партиций

Каждый `SELECT` рассматривает только партиции существующие в таблице смещений. Ее содержание можно изменить до операции `SELECT` если `k_automatic_offsets` настроен, либо при помощи другой функции.

Партиции распределены между *сегментами GPDB* по следующим правилам:

1. Партиции распределены в равных пропорциях между всеми сегментами. Реальное количество партиций привязанных к сегменту различается не больше чем на 1 по всем сегментам.
2. Порядок партиций (возвращенный Kafka) и порядок привязанных к ним сегментов совпадают.

   Пример:

- Кластер состоит из трех сегментов
- Kafka возвращает пять партиций: `[0, 2, 3, 4, 1]`

Партиции распределяются по сегментам следующим образом:

1. `[0, 2]`
2. `[3, 4]`
3. `[1]`

Пользовательская отмена запроса

Любой запрос, включающий функции `librdkafka` или внешние таблицы может быть отменен пользователем до завершения. Запрос гарантированно будет завершен меньше, чем за одну секунду после того, как все сегменты GPDB получат сигнал об отмене запроса.

Заметьте, что активные подключения к Kafka не завершаются мгновенно после отмены запроса. В нормальных условиях подключения будут завершены в пределах `k_latency_ms`. Но, когда Kafka не отвечает (например, не прошла аутентификация Kerberos), подключения остаются активными бесконечно.

Чтобы гарантировать освобождение всех ресурсов, закройте сессию GPDB откуда поступил запрос.

Продолжительность SELECT

Когда запрос `SELECT` не остановлен, его максимальная длительность приблизительно:

```
[duration] = [k_latency_ms] * (1 + ceil([number of partitions] / [number of GPDB segments])) + [k_timeout_
↪ms]
```

`k_timeout_ms` и `k_latency_ms` значительно влияют на продолжительность `SELECT` и потому должны иметь правильные значения (особенно `k_timeout_ms`).