ЛР5. PostgreSQL. Секционирование

1. Секционирование

Секционирование (partitioning) — разделение хранимых объектов баз данных (таких как таблиц, индексов, материализованных представлений) на отдельные части с раздельными параметрами физического хранения. Используется в целях повышения управляемости, производительности и доступности для больших баз данных.

В различных СУБД возможности реализации несколько различаются. Можно выделить следующие типы секционирования:

- по интервалам (range partitioning),
- по списку значений (list partitioning),
- по хешу (hash partitioning),
- по ключам (key partitioning).

Основные преимущества секционирования:

- Секционирование позволяет хранить в одной таблице больше данных, чем может храниться на одном диске или разделе файловой системы.
- Данные, которые теряют свою полезность, часто можно легко удалить из секционированной таблицы, удалив секцию (или секции), содержащую только эти данные. И наоборот, процесс добавления новых данных в некоторых случаях можно значительно облегчить, добавив один или несколько новых разделов для хранения именно этих данных.
- Некоторые запросы могут быть значительно оптимизированы благодаря тому факту, что данные, удовлетворяющие заданному предложению WHERE, могут храниться только в одном или нескольких разделах, что автоматически исключает остальные разделы из поиска.

2. Особенности секционирования таблиц в PostgreSQL

PostgreSQL поддерживает два типа секционирования:

- декларативное секционирование;
- секционирование с использованием наследования.

При декларативном секционировании декларируется, что некоторая таблица разделяется на секции. Сама секционированная таблица является «виртуальной» и

как таковая не хранится. Хранилище используется её секциями, которые являются обычными таблицами, связанными с секционированной.

Сами секции могут представлять собой секционированные таблицы, таким образом реализуется вложенное секционирование. Хотя все секции должны иметь те же столбцы, что и секционированная родительская таблица, в каждой секции независимо от других могут быть определены свои индексы, ограничения и значения по умолчанию.

Преобразовать обычную таблицу в секционированную и наоборот нельзя. Однако в секционированную таблицу можно добавить в качестве секции существующую обычную или секционированную таблицу, а также можно удалить секцию из секционированной таблицы и превратить её в отдельную таблицу.

Возможности секционирования с использованием наследования:

- При декларативном секционировании все секции должны иметь в точности тот же набор столбцов, что и секционируемая таблица, тогда как обычное наследование таблиц допускает наличие в дочерних таблицах дополнительных столбцов, отсутствующих в родительской таблице.
- Механизм наследования таблиц поддерживает множественное наследование.
- С декларативным секционированием поддерживается только разбиение по спискам, по диапазонам и по хешу, тогда как с наследованием таблиц данные можно разделять по любому критерию, выбранному пользователем.

Но при секционировании с использованием наследования для добавления записей в нужные секции необходимо вручную создавать триггеры.

3. Пример декларативного секционирования в PostgreSQL

1. Создание таблицы как секционированной таблицы с предложением PARTITION BY, указав метод разбиения (в нашем случае RANGE) и список столбцов, которые будут образовывать ключ разбиения.

```
CREATE TABLE userslogs (
    username VARCHAR(20) NOT NULL,
    logdata TEXT NOT NULL,
    created DATE NOT NULL
)
PARTITION BY RANGE (created);
```

2. Создание секций. В определении каждой секции должны задаваться границы, соответствующие методу и ключу разбиения родительской таблицы. Указание

границ, при котором множество значений новой секции пересекается со множеством значений в одной или нескольких существующих секциях, будет ошибочным.

```
CREATE TABLE userslogs_y2013 PARTITION OF userslogs
     FOR VALUES FROM (MINVALUE) TO ('2014-01-01');
CREATE TABLE userslogs_y2014 PARTITION OF userslogs
     FOR VALUES FROM ('2014-01-01') TO ('2015-01-01');
CREATE TABLE userslogs_y2015 PARTITION OF userslogs
     FOR VALUES FROM ('2015-01-01') TO ('2016-01-01');
CREATE TABLE userslogs_y2016 PARTITION OF userslogs
     FOR VALUES FROM ('2016-01-01') TO (MAXVALUE);
```

3. Создание в секционируемой таблице индекса по ключевому столбцу (или столбцам), а также любые другие индексов, которые могут понадобиться. При этом автоматически будет создан соответствующий индекс в каждой секции, и все секции, которые будут создаваться или присоединяться позднее, тоже будут содержать такой индекс. Индексы или ограничения уникальности, созданные в секционированной таблице, являются «виртуальными», как и сама секционированная таблица: фактически данные находятся в дочерних индексах отдельных таблиц-секций.

```
CREATE INDEX ON userslogs (created);
```

4. Проверка, что параметр конфигурации `enable_partition_pruning` не выключен в postgresql.conf. Иначе запросы не будут оптимизироваться должным образом.

Для секционирования по хешу (`PARTITION BY HASH (field_name)`) в каждой секции необходимо определить параметры (модуль, остаток от деления) следующим образом:

```
CREATE TABLE part_0 PARTITION OF parent_table
     FOR VALUES WITH (MODULUS 3, REMAINDER 0);
CREATE TABLE part_1 PARTITION OF parent_table
     FOR VALUES WITH (MODULUS 3, REMAINDER 1);
…
```

4. Пример секционирования с использованием наследования в PostgreSQL

1. Создание «главной» таблицы, от которой будут наследоваться все «дочерние» таблицы. Главная таблица не будет содержать данные. Не нужно определять в ней никакие ограничения-проверки, если только есть определенное намерение применить их во всех дочерних таблицах. Также не имеет смысла определять в ней какие-либо индексы или ограничения уникальности.

```
CREATE TABLE userslogs (
    username VARCHAR(20) NOT NULL,
    logdata TEXT NOT NULL,
    created DATE NOT NULL
);
```

2. Создание нескольких «дочерних» таблиц - наследников главной. Обычно в такие таблицы не добавляются дополнительные столбцы, используются только унаследованные. Как и с декларативным секционированием, эти таблицы во всех отношениях будут обычными таблицами PostgreSQL (или внешними таблицами). В дочерние таблицы добавляются неперекрывающиеся ограничения, определяющие допустимые значения ключей для каждой из них.

```
CREATE TABLE userslogs_y2013( CHECK (created >= MINVALUE AND created < '2014-01-01') ) INHERITS (userslogs);
CREATE TABLE userslogs_y2014( CHECK (created >= '2014-01-01' AND created < '2015-01-01') ) INHERITS (userslogs);
CREATE TABLE userslogs_y2015( CHECK (created >= '2015-01-01' AND created < '2016-01-01') ) INHERITS (userslogs);
CREATE TABLE userslogs_y2016( CHECK (created >= '2016-01-01' AND created < MAXVALUE) ) INHERITS (userslogs);
```

3. Создание индекса по ключевому столбцу (или столбцам) для каждой дочерней таблицы, а также любых других индексов по усмотрению.

```
CREATE INDEX userslogs_y2013_ind ON userslogs_y2013 (created);
CREATE INDEX userslogs_y2014_ind ON userslogs_y2014 (created);
CREATE INDEX userslogs_y2015_ind ON userslogs_y2015 (created);
CREATE INDEX userslogs_y2016_ind ON userslogs_y2016 (created);
```

4. Создание триггера или правила на вставку в дочерние таблицы при заполнении родительской.

ЛР5. Вопросы для контроля

1. Секционирование. Типы секционирования
2. Виды секционирования в PostgreSQL.
3. Особенности декларативного секционирования.
4. Особенности секционирования с использованием наследования.
5. Отсечение секций.

ЛР5. Задание

1. Восстановить из логической резервной копии (созданной утилитой `pg_dump`) демонстрационную базу данных https://edu.postgrespro.ru/demo-big.zip. Диаграмма таблиц базы данных https://postgrespro.ru/docs/postgrespro/10/apjs02.html.
2. Выполнить декларативное секционирование таблицы `bookings.bookings`, чтобы оптимизировать поиск пассажиров по дате бронирования билета и коду бронирования. Секционирование таблицы выполнить по интервалам, по дате бронирования. Реализовать вложенное секционирование для одной из секций таблицы, вложенное разбиение выполнить по хешу, по коду бронирования.
3. Создать индекс по ключевому столбцу созданной таблицы.
4. Выполнить запрос поиска информации о бронировании билетов. Вывести информацию о забронированных билетах:
    4.1. В указанный день;
    4.2. В указанный день с указанием кода бронирования.

    Сравнить план выполнения запросов к исходной и секционированной таблицам.
5. Реализовать секционирование таблицы `ticket_flights` с использованием наследования, разбиение выполнить по каждому типу билета (эконом-класс, …).
6. Для каждой дочерней таблицы создать индекс по ключевому столбцу.
7. Создать триггер, вызывающий функцию, выполняющую вставку данных в дочерние таблицы перед вставкой в родительскую таблицу. Заполнить секционированную таблицу данными.
8. Выполнить обновление данных в секционированной таблице, изменив значение поля «`amount`» для конкретного перелета.
9. Выполнить запрос поиска информации о бронировании указанного типа билетов в указанную дату. Вывести информацию о забронированных билетах.