

LP5. PostgreSQL. Секционирование

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.

```
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.

```
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);  
...
```

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.

Теоретическая (тестовая) часть

Лекция 8. Секционирование. Сегментирование [Общая часть. MySQL. PostgreSQL]

Практическая часть

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 дате бронирования (способ секционирования – по интервалам для каждого года). Реализовать вложенное секционирование для одной из секций таблицы, вложенное разбиение выполнить по коду бронирования (способ секционирования – по хешу).

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 для вставки 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.

Бонусная часть (1 балл)

Написать Python-скрипт автоматической проверки задания на ЛР: проверка декларативного секционирования, проверка секционирования с использованием наследования (включая проверку операций вставки).