Секционирование больших таблиц в PostgreSQL

Фролков И.А., Постгрес Профессиональный

http://www.devconf.ru

Кто виноват

- Что делать, если таблица растет неограниченно?
- А почему это, собственно, плохо?
  - Затруднен или практически невозможен DDL – ни построить индекс, ни поменять тип колонки
  - Сборка мусора и transaction wraparound
    - Один процесс VACUUM на таблицу
  - Фактически ограничиваем размер БД.

Что делать? Секционирование!

- Одна большая таблица - плохо
- Много маленьких таблиц – лучше
  - Что значит “много” и насколько лучше?
- Что это дает?
  - Можно перестраивать индексы, не боясь занять все доступное место
  - Можно изменять таблицу по частям
  - Группировка однородных данных. Типичный пример – журнал с группировкой по дате

Секционирование в PostgreSQL

- Как это сделано - наследование
- Пример:
  - Создание таблицы и триггера

```
create table parttest(id int, val text);
create table parttest_lt_1000(check(id<1000)) inherits(parttest);
create table parttest_ge_1000(check(id>=1000)) inherits(parttest);
```

- Создание триггера before insert

```
CREATE TRIGGER ord_log_insert_trigger
  BEFORE INSERT
  ON ord.log
  FOR EACH ROW
  EXECUTE PROCEDURE public.parttest();
```

Секционирование в PostgreSQL

```
create or replace function parttest_insert() returns trigger as
$code$
begin
 if new.id<1000 then
   insert into parttest_lt_1000 select new.*;
 else
   insert into parttest_ge_1000 select new.*;
 end if;
 return null;
end;
$code$
language plpgsql
```

Секционирование в PostgreSQL

```
[local] postgres@work=# insert into parttest values(1,'test');
INSERT 0 0
Время: 15,713 мс
[local] postgres@work=#
[local] postgres@work=# select * from parttest;
Время: 0,705 мс
```

id	val
1	test

Ура! Работает! Но…

- При выполнении просматриваются все секции, если нельзя выбрать секцию при компиляции запроса.

```
explain analyze select * from generate_series(1,10) gs(n), parttest p where p.id=gs.n;
```

```
                                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Merge Join  (cost=248.94..444.51 rows=12705 width=40) (actual time=0.029..0.031 rows=2 loops=1)
   Merge Cond: (gs.n = p.id)
   ->  Sort  (cost=59.83..62.33 rows=1000 width=4) (actual time=0.018..0.019 rows=2 loops=1)
         Sort Key: gs.n
         Sort Method: quicksort  Memory: 25kB
         ->  Function Scan on generate_series gs  (cost=0.00..10.00 rows=1000 width=4) (actual time=0.010..0.011 rows=10 loops=1)
   ->  Sort  (cost=189.11..195.46 rows=2541 width=36) (actual time=0.008..0.008 rows=2 loops=1)
         Sort Key: p.id
         Sort Method: quicksort  Memory: 25kB
         ->  Append  (cost=0.00..45.40 rows=2541 width=36) (actual time=0.004..0.005 rows=2 loops=1)
               ->  Seq Scan on parttest p  (cost=0.00..0.00 rows=1 width=36) (actual time=0.000..0.000 rows=0 loops=1)
               ->  Seq Scan on parttest_lt_1000 p_1  (cost=0.00..22.70 rows=1270 width=36) (actual time=0.002..0.002 rows=2 loops=1)
               ->  Seq Scan on parttest_ge_1000 p_2  (cost=0.00..22.70 rows=1270 width=36) (actual time=0.001..0.001 rows=0 loops=1)
 Planning time: 0.156 ms
 Execution time: 0.069 ms
(15 строк)
```

Просматриваются ВСЕ секции

А что с индексами?

- А с индексами все плохо:
  - Индексы только на секцию и только
  - Нет индексов
    - Общих для всей таблицы (глобальных)
    - Секционированных по собственному правилу

Что в других СУБД?

- Oracle
  - Секции выбираются только необходимые
  - Индексы
    - Глобальные
    - Локальные
    - Секционированные по своему
  - Многоуровневое секционирование
  - Секций может быть много

MySQL

- Выбор секции в процессе
- Индексы — по факту только локальные
- Много секций не рекомендуют

SQL Server

- Индексы глобальные, локальные

"Creating and rebuilding nonaligned indexes on a table with more than 1,000 partitions is possible, but is not supported."

Sybase

- Локальные, глобальные индексы
- Про количество неясно

DB2

- Индексы глобальные, локальные
- Многоуровневое секционирование, **MDC**

PostgreSQL — что делать?

- Здесь и сейчас - LATERAL
- pg_pathman

LATERAL

- Можно ссылаться на колонки таблицы справа
- Может быть функция
- Функция может произвольно обращаться к данным
  - Достоинства — неограниченная гибкость
  - Недостатки — не самая впечатляющая производительность

Сюда пишется заголовок слайда, также до 3 строк, не больше!

- Это – контентная область слайда.
- Желательно создавать новые слайды, дублируя (copy/paste) этот слайд в качестве образца, чтобы сохранить общую стилистику презентации. Также рекомендуется для набора текста использовать шрифт Trebuchet.

LATERAL - пример

```
create or replace function ord.get_usr(p_ids int[]) returns
table(id int, email text)
as $code$
 begin
   return query execute
      format('select * from ord.usr2_1000_%s where id=$1',
             get_hash(hashint4(p_id),1000))
      using p_id;
 end;
$code$
```

pg_pathman

- Нормальное секционирование (Дмитрий Иванов, Ильдар Мусин)
- Секционирование по хешу, диапазону
- Пока только набор фунций
- Реализовано как extension

pg_pathman

- Таблица преобразуется с помощью функций

```
select create_hash_partitions(
    'ord.usr1',
    'id',
    100)
```

pg_pathman

- На основе inherits
- Секции выбираются во время выполнения

```
explain analyze
 select * from ord.usr1 u where u.id in(select gs.n from generate_series(1,4) as gs(n) );
"Nested Loop  (cost=0.29..746.70 rows=500000 width=30) (actual time=0.042..0.112 rows=4 loops=1)"
"  ->  Function Scan on generate_series gs  (cost=0.00..10.00 rows=1000 width=4) (actual
time=0.007..0.009 rows=4 loops=1)"
"  ->  Custom Scan (RuntimeAppend)  (cost=0.29..0.73 rows=1 width=30) (actual time=0.016..0.016 rows=1
loops=4)"
"        ->  Index Scan using usr1_70_pkey on usr1_70 u  (cost=0.29..0.73 rows=1 width=30) (actual
time=0.018..0.018 rows=1 loops=1)"
"              Index Cond: (id = gs.n)"
"        ->  Index Scan using usr1_26_pkey on usr1_26 u  (cost=0.29..0.73 rows=1 width=30) (actual
time=0.014..0.015 rows=1 loops=1)"
"              Index Cond: (id = gs.n)"
"        ->  Index Scan using usr1_27_pkey on usr1_27 u  (cost=0.29..0.72 rows=1 width=30) (actual
time=0.015..0.015 rows=1 loops=1)"
"              Index Cond: (id = gs.n)"
"        ->  Index Scan using usr1_63_pkey on usr1_63 u  (cost=0.29..0.72 rows=1 width=30) (actual
time=0.014..0.014 rows=1 loops=1)"
"              Index Cond: (id = gs.n)"
"Planning time: 4.527 ms"
"Execution time: 0.144 ms"
```

Производительность

- Таблица вида usr(id int, email text), id in [1...1000000]
- Запрос

```
select * from usrN u where u.id in(select gs.n from
generate_series(1,100) as gs(n) );
```

pg_pathman

-

pg_pathman				
Секций	0	10	100	1000
tps	4358	3170	1761	648

inherits			
Секций	10	100	1000
tps	1040	104	10

pg_pathman

- Что дальше?
  - Увеличение производительности
  - Декларативный синтаксис — к сожалению, невозможно реализовать как расширение
  - Поддержка разных типов индексов

Вопросы?