Временные таблицы в Postgresql — проблемы и методы решения

И.А.Фролков

www.postgrespro.ru

Зачем нужны временные таблицы?

- Сессионные данные
  - В Oracle — переменные пакетов. Надо очень быстро и дешево
- Промежуточные результаты. Также обычно надо дешево и быстро.
- Взаимодействие процедур. Опять же дешево, быстро и независимо друг от друга.
  - Таблица как параметр

Временные таблицы в Postgres

- Такая же таблица, как и остальные
- Отсюда:
  - Строчки в pg_class, pg_attribute, pg_depends
  - Как минимум один файл на каждую таблицу
  - Характерный паттерн работы с временными таблицами ведет к переполнению кеша ОС мусором.

Postgres PROFESSIONAL

Временные таблицы в Postgres

- Не видит автовакуум, отсюда:
  - Ручной ANALYZE
  - Недоступность для VACUUM
  - Ручной VACUUM тоже не работает в процедурах, надо вызывать отдельно.
    - Следствие — таблица может неисправимо замусориваться.

Отступление — что такое VACUUM

- При обновлении старая строка помечается как удаленная и добавляется новая строка.
- Отсюда — мусорные строки в таблице
- Отсюда — мусорные строки в индексе
- Видимость строк для различных транзакций
- Index-only scan & visibility map

Антипаттерн для временных таблиц

- Создание и удаление действительно большого количества
- Активное удаление/обновление строк
- Да, это встречается. pg_attributes в 12Г - реальность

Postgres PROFESSIONAL

Все ли так плохо?

- И да и нет
- Временные таблицы
  - ПЛОХОЕ решение для высоконагруженных страниц в вебе
  - Удовлетворительное решение для внутренних сервисов — админки и т. п.
  - НОРМАЛЬНОЕ решение для скриптов, различных периодических задач и т.п.

Postgres PROFESSIONAL

Что делать?

- Не использовать временные таблицы :-)
- А если надо? Альтернативы:
  - Массивы
  - pg_variables

Массивы

- Отличное решение, но...
  - Существуют только в период выполнения транзакции. Можно сложить во временную таблицу, но ее будет необходимо обслуживать
  - Массивы иммутабельны.

Postgres PROFESSIONAL

pg_variables

- Расширение, разработанное Postgres Professional
- Нет недостатков временных таблиц
- Простой и ясный api
  - pgv_set_int('package','name',value)
  - pgv_get_int('package','name')
  - pgv_insert('package','name',row(value1,value2…))
  - pgv_select('package','name')

pg_variables

- Пример — тестовая база, datafiller

```
CREATE TABLE ord.goods( -- df: mult=1000.0
    id SERIAL primary key,
    name TEXT NOT NULL, -- df: lenmin=3 lenmax=30 chars='a-f ' sub=uniform
    price numeric, -- df: float=gauss alpha=100.0 beta=30
    in_stock_qty int -- df: size=1000
);
create table ord.usr( -- df: mult=100
   id serial primary key,
   email text -- df: pattern='[a-z]{3,16}\.[a-z]{3,8}@((gmail|yahoo|
mail)\.com|(mail|yandex|inbox)\.ru)'
);
create table ord.discount( -- df: mult=100
  goods_id int not null references ord.goods(id),
  usr_id int not null references ord.usr(id),
  pct numeric not null, -- df: alpha=0.01 beta=0.07
  from_date date not null, -- df: start=2010-01-01 end=2016-04-01
  duration integer not null -- df: offset=1 size=361 step=30
)
```

pg_variables

```
create or replace function get_mailru_discounts_pgvariables()
 returns table(usr_cnt int, discounts_cnt int) as
$code$
begin
  if exists(select * from pgv_list()
              where package='package' and name='set') then
    perform pgv_remove('package','set');
  end if;
  perform pgv_insert('package', 'set', row(id))
     from ord.usr u where u.email like 'ab%@mail.ru';
  get diagnostics usr_cnt = row_count;
  select count(*) into discounts_cnt
    from ord.discount d, pgv_select('package','set') u(id int)
   where d.usr_id=u.id;
  return next;
end;
$code$
language plpgsql;
```

Postgres PROFESSIONAL

Результат

Вариант/транзакций/с	100	1000	5000	10000	20000
plain	10170	11349	11537	11560	11639
temptable	3364	3380	561	678	378
pg_variables	11852	15944	16634	16748	16719

Postgres PROFESSIONAL

pg_variables

- Недостатки
  - Нестандартный
  - Неясное будущее
- Достоинства
  - Быстро
  - Данные доступны в течение сессии
  - Свободен от недостатков временных таблиц

Postgres PROFESSIONAL

Перспективы

- В настоящее время готовится патч с новой версией временных таблиц, которые не используют системный каталог
- Но его надо еще смотреть и тестировать — не раньше 9.7

Postgres PROFESSIONAL

Спасибо за внимание!

Контакты:

i.frolkov@postgrespro.ru

+79168019789