



## Защита проекта

Тема: Разработка интернет-магазина книг и практическое  
тестирование разных сценариев нагрузки



Калиниченко Майя Евгеньевна

Должность: Старший инженер

Компания: ПАО «Ростелеком»



# Введение

Целью данного проекта является изучение особенностей работы rgbench при тестировании работы БД нестандартными скриптами нагрузки, разработанными под особенности конкретной БД PostgreSQL.

Для демонстрации работы проекта в Yandex Cloud была создана виртуальная машина с ОС Ubuntu 22.04 (2 ядра, RAM 4 Гб, SSD диск 16 Гб). На ней была развёрнута СУБД PostgreSQL 15 версии.

# Цели проекта

1

Закрепление основных навыков полученных на текущем курсе

2

Создание минимальной БД в PostgreSQL

3

Заполнение таблиц БД тестовыми данными

4

Изучение особенностей настройки pgbench под тестирование конкретной БД

5

Проведение тестирования созданной БД pgbench собственными тест-скриптами

# Что планировалось

1

Создать тестовую БД в PostgreSQL

2

Заполнить БД тестовыми данными

3

Подготовить скрипты для тестирования производительности

4

Прогнать pgbench с подготовленными скриптами

Выделение  
ресурсов



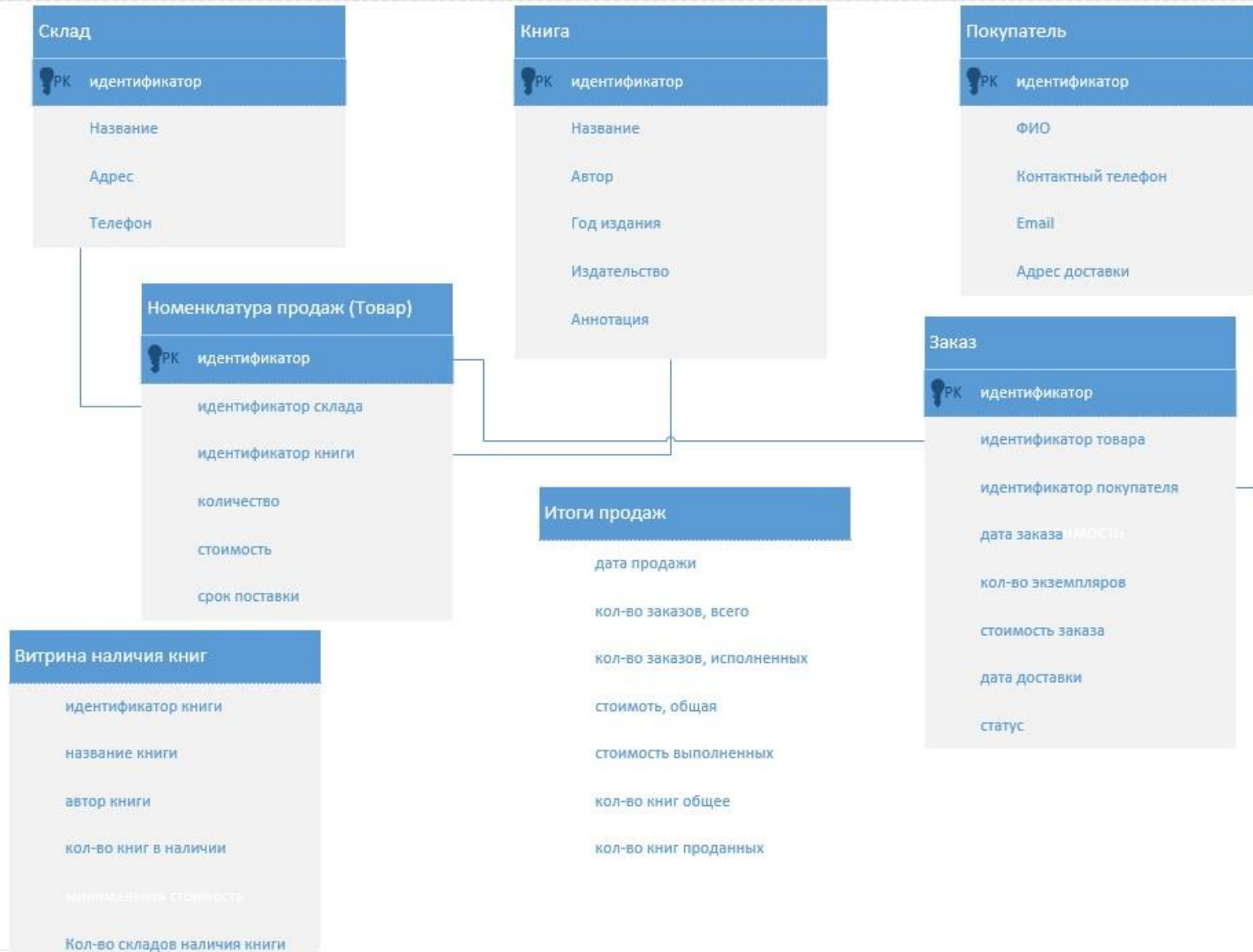
# Что получилось

## Репозиторий с текущей версией проекта

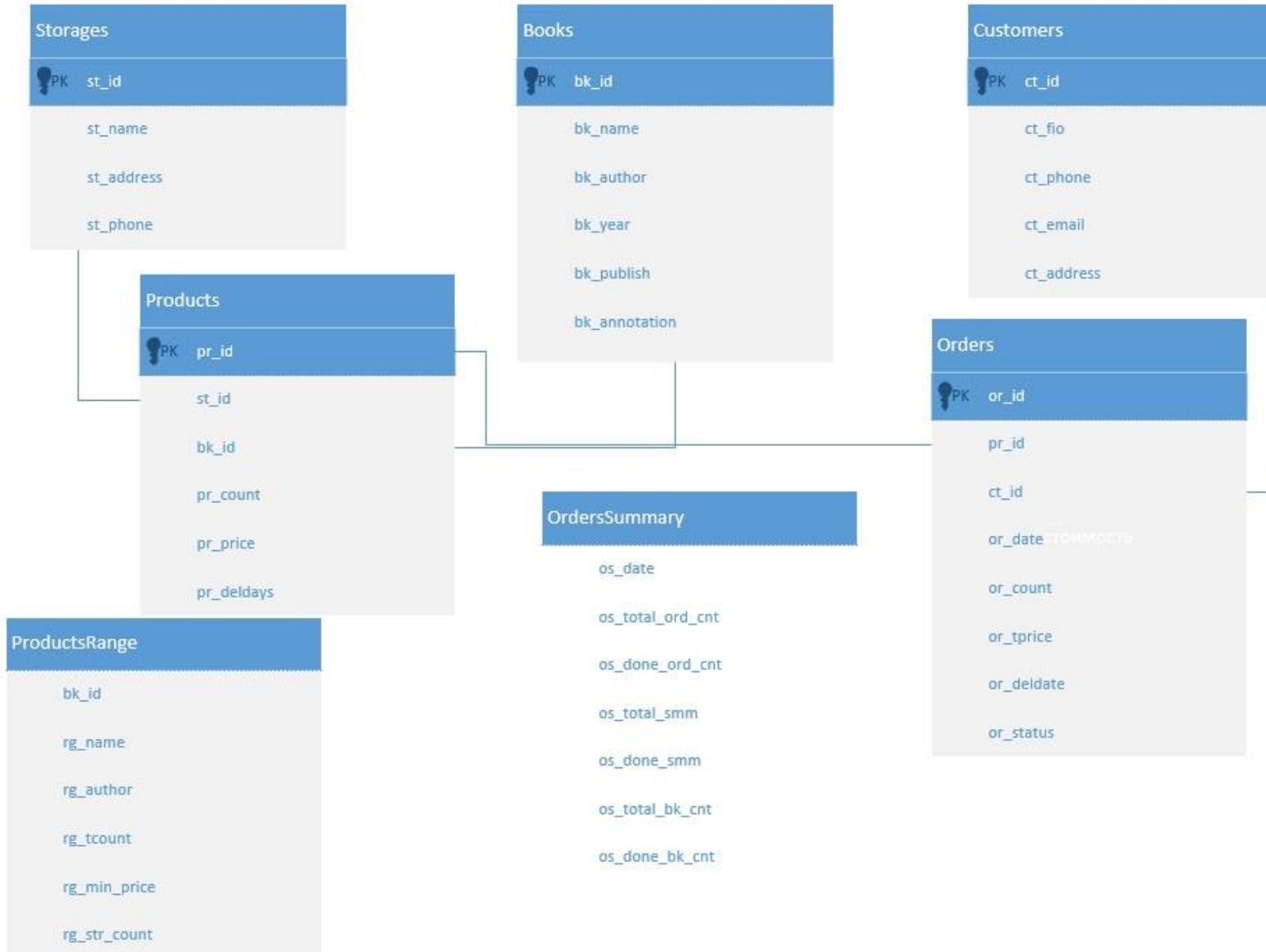
[https://github.com/mkalinichenko2023/otus\\_project/blob/main/README.md](https://github.com/mkalinichenko2023/otus_project/blob/main/README.md)

- 1 Разработана схема БД по теме «Интернет-магазин книг»
- 2 В Yandex Cloud была создана виртуальная машина, установлен PostgreSQL 15. В БД MyBookShop скриптами созданы основные объекты и функции.
- 3 Таблицы были заполнены тестовыми данными с использованием загрузки готовых списков, а также функции генерации случайных данных.
- 4 Подготовлены тестовые скрипты для проверки производительности системы.
- 5 Выполнено тестирование производительности системы rgbench с использованием подготовленных скриптов.

# Схемы (архитектура, БД)



# Схемы (архитектура, БД)



# Скрипты создания таблиц схемы

--создать базу, схему и таблицы

```
create database mybookshop;
```

```
\c mybookshop;
```

```
create schema bookshop;
```

```
set search_path = bookshop, pg_catalog;
```

--Книги

```
create table books(bk_id      integer,  
                  bk_name     varchar(500) not null,  
                  bk_author   varchar(500),  
                  bk_year     varchar(30),  
                  bk_publish   varchar(500),  
                  bk_annotation varchar(4000),  
                  constraint Books_pkey primary key(bk_id));
```

```
comment on table books is 'Книги';
```

```
comment on column books.bk_id is 'Идентификатор';
```

```
comment on column books.bk_name is 'Название';
```

```
comment on column books.bk_author is 'Автор';
```

```
comment on column books.bk_year is 'Год издания';
```

```
comment on column books.bk_publish is 'Издательство';
```

```
comment on column books.bk_annotation is 'Аннотация';
```

```
create sequence books_bk_id_seq start with 1 increment by 1 minvalue 1 maxvalue 9999999 cache 1 owned by books.bk_id;
```



# Скрипты создания таблиц схемы

--Триггерные функции к таблице Книги

create or replace function fOnUpdBooks()

returns trigger as \$BODY\$

declare

begin

if Old.bk\_name <> New.bk\_name OR Old.bk\_author <> New.bk\_author OR Old.bk\_year <> New.bk\_year OR Old.bk\_publish <> New.bk\_publish then

RAISE EXCEPTION 'Обновления для столбцов "bk\_name","bk\_author","bk\_year" и "bk\_publish" запрещены!';

end if;

return New;

end; \$BODY\$

language plpgsql;

create or replace trigger trOnUpdBooks

before update on books for each row execute function fOnUpdBooks();

create or replace function fBefInsertBooks()

returns trigger as \$BODY\$

declare

begin

if New.bk\_id is null then

New.bk\_id:= nextval('books\_bk\_id\_seq');

end if;

return New;

end; \$BODY\$

language plpgsql;

create or replace trigger trBefInsBooks

before insert on books for each row execute function fBefInsertBooks();

# Скрипты создания таблиц схемы

--Склады

```
create table Storages(st_id      integer,
                      st_name    varchar(500) not null,
                      st_address  varchar(1000),
                      st_phone    varchar(100),
                      constraint Storages_pkey primary key(st_id));
comment on table Storages is 'Склады';
comment on column Storages.st_id is 'Идентификатор';
comment on column Storages.st_name is 'Название';
comment on column Storages.st_address is 'Адрес';
comment on column Storages.st_phone is 'Контактный телефон';
create sequence Storages_st_id_seq start with 1 increment by 1
minvalue 1 maxvalue 9999 cache 1 owned by Storages.st_id;
```

--Покупатели

```
create table Customers(ct_id      integer,
                      ct_fio      varchar(1000) not null,
                      ct_phone    varchar(100),
                      ct_email    varchar(100),
                      ct_address  varchar(1000),
                      constraint Customers_pkey primary key(ct_id));
comment on table Customers is 'Покупатели';
comment on column Customers.ct_id is 'Идентификатор';
comment on column Customers.ct_fio is 'ФИО';
comment on column Customers.ct_phone is 'Контактный телефон';
comment on column Customers.ct_email is 'Эл.почта';
comment on column Customers.ct_address is 'Адрес';
create sequence Customers_ct_id_seq start with 1 increment by 1 minvalue 1 maxvalue 9999999 cache 1 owned by Customers.ct_id;
```

# Скрипты создания таблиц схемы

```
create or replace function fBefInsertCustomers()
returns trigger as $BODY$
declare
begin
    if New.ct_id is null then
        New.ct_id:= nextval('customers_ct_id_seq');
    end if;
return New;
end; $BODY$ language plpgsql;
create or replace trigger trBefInsCustomers before insert on Customers for each row execute function fBefInsertCustomers();
```

--Номенклатура продаж

```
create table Products(pr_id      integer,
                      st_id      integer,
                      bk_id      integer,
                      pr_count   integer not null check (pr_count>=0),
                      pr_price   numeric(9,2) not null check (pr_price>=0),
                      pr_deldays integer not null check (pr_deldays>0),
                      constraint Products_pkey primary key(pr_id),
                      constraint Products_Storages_fkey foreign key (st_id) references Storages(st_id),
                      constraint Products_Books_fkey foreign key (bk_id) references Books(bk_id));
comment on table Products is 'Номенклатура продаж';
comment on column Products.pr_id is 'Идентификатор';
comment on column Products.st_id is 'Идентификатор склада';
comment on column Products.bk_id is 'Идентификатор книги';
comment on column Products.pr_count is 'Количество';
comment on column Products.pr_price is 'Стоимость';
comment on column Products.pr_deldays is 'Дни доставки';
create sequence Products_pr_id_seq start with 1 increment by 1 minvalue 1 maxvalue 9999999999 cache 1 owned by Products.pr_id;
```

# Скрипты создания таблиц схемы

```
create or replace function fOnUpdProducts()
returns trigger as $BODY$
declare
begin
    if Old.st_id <> New.st_id OR Old.bk_id <> New.bk_id then
        RAISE EXCEPTION 'Обновления для столбцов "st_id" и "bk_id" запрещены!';
    end if;
return New;
end; $BODY$
language plpgsql;
```

```
create or replace trigger trOnUpdProducts
before update on Products for each row
execute function fOnUpdProducts();
```

--Заказы

```
create table Orders(or_id      integer,
                    ct_id      integer,
                    pr_id      integer,
                    or_date     timestamp not null,
                    or_count    integer not null check (or_count>0),
                    or_tprice   numeric(9,2) not null check (or_tprice>=0),
                    or_deldate  timestamp not null check (or_deldate>or_date),
                    or_status   varchar(20),
                    constraint Orders_pkey primary key(or_id),
                    constraint Orders_Customers_fkey foreign key (ct_id) references Customers(ct_id),
                    constraint Orders_Products_fkey foreign key (pr_id) references Products(pr_id),
                    constraint Orders_or_status_check check (or_status in ('Create','In delivery','Cancelled','Done')));
```



# Скрипты создания таблиц схемы

```
comment on table Orders is 'Заказы';
comment on column Orders.or_id is 'Идентификатор';
comment on column Orders.ct_id is 'Идентификатор покупателя';
comment on column Orders.pr_id is 'Идентификатор номенклатуры продаж';
comment on column Orders.or_date is 'Дата и время заказа';
comment on column Orders.or_count is 'Кол-во экземпляров';
comment on column Orders.or_tprice is 'Итоговая стоимость';
comment on column Orders.or_deldate is 'Дата доставки';
comment on column Orders.or_status is 'Статус заказа';
create sequence Orders_or_id_seq start with 1 increment by 1
minvalue 1 maxvalue 9999999999 cache 1 owned by Orders.or_id;
```

```
create or replace function fOnUpdOrders()
returns trigger as $BODY$
begin
    if Old.ct_id <> New.ct_id or Old.pr_id <> New.pr_id or Old.or_count <> New.or_count then
        RAISE EXCEPTION 'Обновления для столбцов "ct_id", "pr_id" и "or_count" запрещены!';
    end if;
return New;
end; $BODY$ language plpgsql;
```

```
create or replace function fOnRemOrders()
returns trigger as $BODY$
begin
    RAISE EXCEPTION 'Удаление заказа запрещено! Переведите в состояние "Cancelled".';
return New;
end; $BODY$ language plpgsql;
```

```
create or replace trigger trOnUpdOrders before update on Orders for each row execute function fOnUpdOrders();
create or replace trigger trOnRemOrders before delete on Orders for each row execute function fOnRemOrders();
```

# Скрипты создания таблиц схемы

--Витрина наличия книг

```
create table ProductsRange(bk_id      integer,
                           rg_name     varchar(500) not null,
                           rg_author   varchar(500),
                           rg_tcount   integer not null check (rg_tcount>=0),
                           rg_min_price numeric(9,2) not null check (rg_min_price>=0),
                           rg_str_count integer);
comment on table ProductsRange is 'Витрина наличия книг';
comment on column ProductsRange.bk_id is 'Идентификатор книги';
comment on column ProductsRange.rg_name is 'Название книги';
comment on column ProductsRange.rg_author is 'Автор книги';
comment on column ProductsRange.rg_tcount is 'Всего книг в наличии';
comment on column ProductsRange.rg_min_price is 'Минимальная стоимость книги';
comment on column ProductsRange.rg_str_count is 'Кол-во складов';
```

--Отчет по продажам

```
create table OrdersSummary(os_date     date not null,
                           os_total_ord_cnt integer,
                           os_done_ord_cnt integer,
                           os_total_smm  numeric(9,2),
                           os_done_smm  numeric(9,2),
                           os_total_bk_cnt integer,
                           os_done_bk_cnt integer);
comment on table OrdersSummary is 'Итоги продаж';
comment on column OrdersSummary.os_date is 'Дата продажи';
comment on column OrdersSummary.os_total_ord_cnt is 'Кол-во заказов общее';
comment on column OrdersSummary.os_done_ord_cnt is 'Кол-во заказов выполненных';
comment on column OrdersSummary.os_total_smm is 'Стоимость заказов общая';
comment on column OrdersSummary.os_done_smm is 'Стоимость заказов выполненных';
comment on column OrdersSummary.os_total_bk_cnt is 'Кол-во книг по всем заказам';
comment on column OrdersSummary.os_done_bk_cnt is 'Кол-во книг по выполненным заказам';
```

# Скрипты создания таблиц схемы

В результате выполнения скриптов было создано 7 таблиц в схеме bookshop БД MyBookShop.

```
mkalinichenko@mkalinichenko-pr: ~  
mybookshop=# \dt  
List of relations  
Schema | Name | Type | Owner  
-----+-----+-----+-----  
bookshop | books | table | postgres  
bookshop | customers | table | postgres  
bookshop | orders | table | postgres  
bookshop | orderssummary | table | postgres  
bookshop | products | table | postgres  
bookshop | productsrange | table | postgres  
bookshop | storages | table | postgres  
(7 rows)  
mybookshop=#
```

# Создание триггерных функций для поддержания витрин в актуальном состоянии

--Триггеры на "витрину наличия книг" ProductsRange

create or replace function fOnAddProducts()

returns trigger

as

\$BODY\$

declare

book\_name varchar(500);

book\_author varchar(500);

row\_cnt integer;

begin

select bk\_name,bk\_author into book\_name,book\_author from bookshop.Books b where b.bk\_id= New.bk\_id;

select count(\*) into row\_cnt from bookshop.ProductsRange pp where pp.bk\_id=New.bk\_id;

if row\_cnt=0 then

insert into bookshop.ProductsRange(bk\_id,rg\_name,rg\_author,rg\_tcount,rg\_min\_price,rg\_str\_count)

values(New.bk\_id,book\_name,book\_author,New.pr\_count,New.pr\_price,1);

else

update bookshop.ProductsRange

set rg\_tcount= rg\_tcount+New.pr\_count, rg\_min\_price=least(rg\_min\_price,New.pr\_price), rg\_str\_count= rg\_str\_count+1

where bk\_id=New.bk\_id;

end if;

return New;

end;

\$BODY\$

language plpgsql;



# Создание триггерных функций для поддержания витрин в актуальном состоянии

```
create or replace function fOnModifProducts()  
returns trigger as $BODY$  
declare  
    book_count integer;  
    book_price numeric(9,2);  
    stor_cnt integer;  
begin  
    select sum(p.pr_count),min(pr_price),count(pr_id) into book_count,book_price,stor_cnt from bookshop.Products p where p.bk_id=Old.bk_id;  
    update bookshop.ProductsRange set rg_tcount=book_count, rg_min_price=book_price, rg_str_count=stor_cnt where bk_id=Old.bk_id;  
return New;  
end; $BODY$  
language plpgsql;
```

```
create or replace function fOnRemProducts()  
returns trigger as $BODY$  
declare  
    book_count integer;  
    book_price numeric(9,2);  
    stor_cnt integer;  
begin  
    select sum(p.pr_count),min(pr_price),count(pr_id) into book_count,book_price,stor_cnt from bookshop.Products p where p.bk_id=Old.bk_id;  
    if stor_cnt=0 then  
        delete from bookshop.ProductsRange pp where pp.bk_id=Old.bk_id;  
    else  
        update bookshop.ProductsRange set rg_tcount=book_count, rg_min_price=book_price, rg_str_count=stor_cnt where bk_id=Old.bk_id;  
    end if;  
return New;  
end; $BODY$  
language plpgsql;
```

# Создание триггерных функций для поддержания витрин в актуальном состоянии

```
create trigger trOnInsertProducts after insert on Products for each row execute function fOnAddProducts();
create trigger tronDeleteProducts after delete on Products for each row execute function fOnRemProducts();
create trigger trOnUpdateProducts after update on Products for each row execute function fOnModifProducts();
```

-Триггеры на "отчет по продажам" OrdersSummary

```
create or replace function fOnAddOrders()
returns trigger as $BODY$
declare
    row_cnt integer;
begin
    if New.or_status<>'Cancelled' then
        update bookshop.Products set pr_count= greatest(pr_count-New.or_count,0) where pr_id=New.pr_id;
    end if;
    select count(*) into row_cnt from bookshop.OrdersSummary os where os.os_date= New.or_date::date;
    if row_cnt=0 then
        insert into bookshop.OrdersSummary(os_date,os_total_ord_cnt,os_total_smm,os_total_bk_cnt, os_done_ord_cnt,os_done_smm,os_done_bk_cnt)
        values(New.or_date::date,1,New.or_tprice,New.or_count, 0,0,0);
    else
        update bookshop.OrdersSummary
        set os_total_ord_cnt= os_total_ord_cnt+1, os_total_smm= os_total_smm+New.or_tprice, os_total_bk_cnt= os_total_bk_cnt+New.or_count
        where os_date=New.or_date::date;
    end if;
    if New.or_status='Done' then
        update bookshop.OrdersSummary
        set os_done_ord_cnt= os_done_ord_cnt+1, os_done_smm= os_done_smm+New.or_tprice, os_done_bk_cnt= os_done_bk_cnt+New.or_count
        where os_date=New.or_date::date;
    end if;
    return New;
end; $BODY$ language plpgsql;
```

# Создание триггерных функций для поддержания витрин в актуальном состоянии

```
create or replace function fOnModifOrders()
```

```
returns trigger as
```

```
$BODY$
```

```
declare
```

```
begin
```

```
if New.or_status='Cancelled' and Old.or_status<>'Cancelled' then
```

```
    update bookshop.Products set pr_count= pr_count+Old.or_count where pr_id=Old.pr_id;
```

```
end if;
```

```
delete from bookshop.OrdersSummary os where os.os_date= Old.or_date::date;
```

```
insert into bookshop.OrdersSummary(os_date,os_total_ord_cnt,os_total_smm,os_total_bk_cnt, os_done_ord_cnt,os_done_smm,os_done_bk_cnt)
```

```
select Old.or_date::date,count(or_id),sum(or_tprice),sum(or_count),
```

```
    sum(case when or_status='Done' then 1 else 0 end),
```

```
    sum(case when or_status='Done' then or_tprice else 0 end),
```

```
    sum(case when or_status='Done' then or_count else 0 end)
```

```
from bookshop.Orders where or_date::date=Old.or_date::date and or_status<>'Cancelled';
```

```
return New;
```

```
end; $BODY$
```

```
language plpgsql;
```

```
create trigger trOnInsertOrders
```

```
after insert on Orders for each row execute function fOnAddOrders();
```

```
create trigger trOnUpdateOrders
```

```
after update on Orders for each row execute function fOnModifOrders();
```

# Скрипты заполнения тестовых данных

--заполнение данными

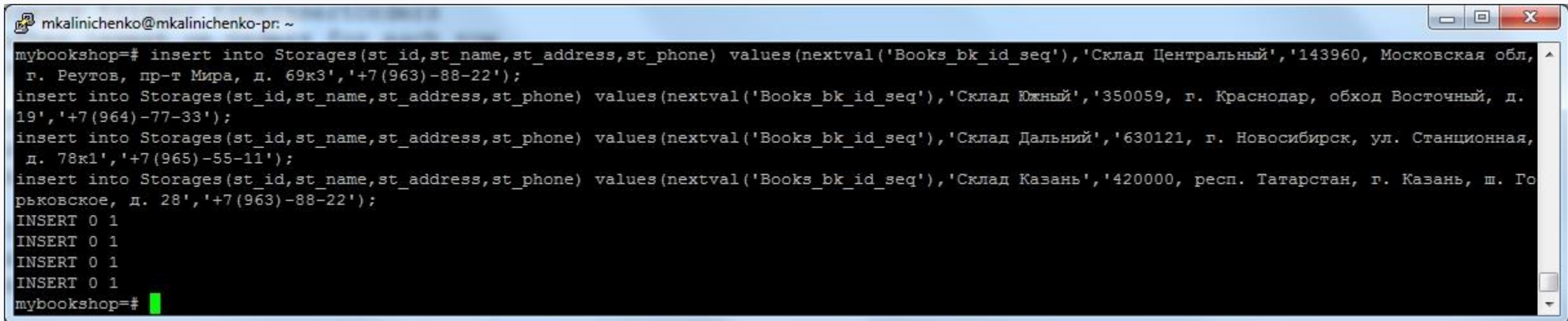
--Склады - небольшая таблица, сделала ручную вставку значений по смыслу

```
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Центральный','143960, Московская обл, г. Реутов, пр-т Мира, д. 69к3','+7(963)-88-22');
```

```
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Южный','350059, г. Краснодар, обход Восточный, д. 19','+7(964)-77-33');
```

```
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Дальний','630121, г. Новосибирск, ул. Станционная, д. 78к1','+7(965)-55-11');
```

```
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Казань','420000, респ. Татарстан, г. Казань, ш. Горьковское, д. 28','+7(963)-88-22');
```



```
mkalinichenko@mkalinichenko-pr: ~  
mybookshop=# insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Центральный','143960, Московская обл,  
г. Реутов, пр-т Мира, д. 69к3','+7(963)-88-22');  
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Южный','350059, г. Краснодар, обход Восточный, д.  
19','+7(964)-77-33');  
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Дальний','630121, г. Новосибирск, ул. Станционная,  
д. 78к1','+7(965)-55-11');  
insert into Storages(st_id,st_name,st_address,st_phone) values(nextval('Books_bk_id_seq'),'Склад Казань','420000, респ. Татарстан, г. Казань, ш. Го  
рьковское, д. 28','+7(963)-88-22');  
INSERT 0 1  
INSERT 0 1  
INSERT 0 1  
INSERT 0 1  
mybookshop=#
```



# Скрипты заполнения тестовых данных

--Книги - большая таблица, скачала из интернет готовый список книг и сделала загрузку в таблицу

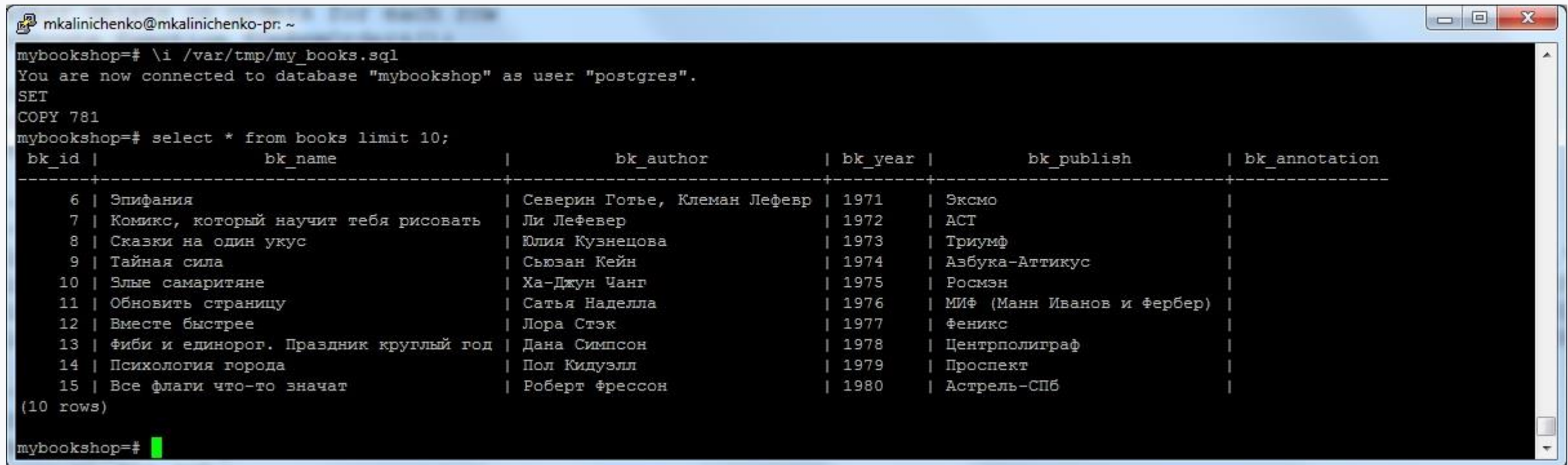
```
cd /var/tmp/
```

```
nano my_books.sql
```

```
sudo -u postgres psql
```

```
\i /var/tmp/my_books.sql
```

```
select * from books limit 10;
```



The screenshot shows a terminal window with the following content:

```
mkalinichenko@mkalinichenko-pr: ~  
mybookshop=# \i /var/tmp/my_books.sql  
You are now connected to database "mybookshop" as user "postgres".  
SET  
COPY 781  
mybookshop=# select * from books limit 10;  
 bk_id |          bk_name          |          bk_author          | bk_year |          bk_publish          |          bk_annotation          |  
-----+-----+-----+-----+-----+-----+  
  6 | Эпифания | Северин Готье, Клеман Лефевр | 1971 | Эксмо |  
  7 | Комикс, который научит тебя рисовать | Ли Лефевр | 1972 | АСТ |  
  8 | Сказки на один укус | Юлия Кузнецова | 1973 | Триумф |  
  9 | Тайная сила | Сьюзан Кейн | 1974 | Азбука-Аттикус |  
 10 | Злые самаритяне | Ха-Джун Чанг | 1975 | Росмэн |  
 11 | Обновить страницу | Сатъя Наделла | 1976 | МИФ (Манн Иванов и Фербер) |  
 12 | Вместе быстрее | Лора Стэк | 1977 | Феникс |  
 13 | Фиби и единорог. Праздник круглый год | Дана Симпсон | 1978 | Центрполиграф |  
 14 | Психология города | Пол Кидуэлл | 1979 | Проспект |  
 15 | Все флаги что-то значат | Роберт Фрессон | 1980 | Астрель-СПб |  
(10 rows)  
mybookshop=#
```

# Скрипты заполнения тестовых данных

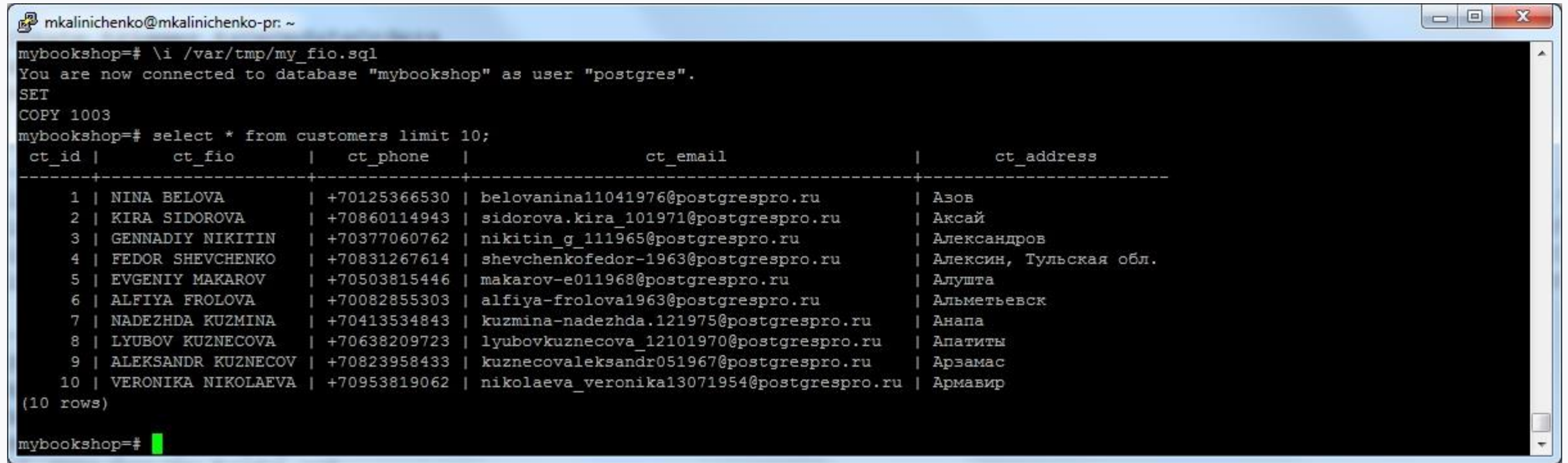
--Покупатели - большая таблица, скачала из интернет готовый список абонентов и сделала загрузку в таблицу

```
cd /var/tmp/
```

```
nano my_fio.sql
```

```
\i /var/tmp/my_fio.sql
```

```
select * from customers limit 10;
```



The screenshot shows a terminal window with the following content:

```
mkalinichenko@mkalinichenko-pr: ~  
mybookshop=# \i /var/tmp/my_fio.sql  
You are now connected to database "mybookshop" as user "postgres".  
SET  
COPY 1003  
mybookshop=# select * from customers limit 10;  
 ct_id |      ct_fio      | ct_phone |      ct_email      |      ct_address  
-----+-----+-----+-----+-----  
 1 | NINA BELOVA      | +70125366530 | belovanina11041976@postgrespro.ru | Азов  
 2 | KIRA SIDOROVA    | +70860114943 | sidorova.kira_101971@postgrespro.ru | Аксай  
 3 | GENNADIY NIKITIN | +70377060762 | nikitin_g_111965@postgrespro.ru    | Александров  
 4 | FEDOR SHEVCHENKO | +70831267614 | shevchenkofedor-1963@postgrespro.ru | Алексин, Тульская обл.  
 5 | EVGENIY MAKAROV  | +70503815446 | makarov-e011968@postgrespro.ru     | Алушта  
 6 | ALFIYA FROLOVA   | +70082855303 | alfiya-frolova1963@postgrespro.ru   | Альметьевск  
 7 | NADEZHDA KUZMINA | +70413534843 | kuzmina-nadezhda.121975@postgrespro.ru | Анапа  
 8 | LYUBOV KUZNECOVA | +70638209723 | lyubovkuznecova_12101970@postgrespro.ru | Апатиты  
 9 | ALEKSANDR KUZNECOV | +70823958433 | kuznecovaleksandr051967@postgrespro.ru | Арзамас  
10 | VERONIKA NIKOLAEVA | +70953819062 | nikolaeva_veronika13071954@postgrespro.ru | Армавир  
(10 rows)  
mybookshop=#
```

# Скрипты заполнения тестовых данных

--Номенклатура продаж - таблица зависит от данных в таблицах "Склады" и "Книги", сделала случайную генерацию данных по наличию книг на всех складах в цикле

```
DO $$
```

```
declare
```

```
  Rec record;
```

```
begin
```

```
  for Rec in (select st_id from Storages)loop
```

```
    insert into Products(pr_id,st_id,bk_id,pr_count,pr_price,pr_deldays)
```

```
      select nextval('products_pr_id_seq'),Rec.st_id,b.bk_id,floor(random()*2000)+1,
```

```
        floor(random()*1000)+99,floor(random()*10)+1
```

```
        from Books b where mod(b.bk_id,Rec.st_id)=0 limit 100;
```

```
  end loop;
```

```
end$$;
```

```
select * from Products limit 10;
```

```
select * from ProductsRange limit 10;
```

```
mkalinichenko@mkalinichenko-pr: ~  
declare  
  Rec record;  
begin  
  for Rec in (select st_id from Storages)loop  
    insert into Products(pr_id,st_id,bk_id,pr_count,pr_price,pr_deldays)  
      select nextval('products_pr_id_seq'),Rec.st_id,b.bk_id,  
        from Books b where mod(b.bk_id,Rec.st_id)=0 limit 100  
    end loop;  
end$$;  
DO  
mybookshop=# select * from Products limit 10;  
 pr_id | st_id | bk_id | pr_count | pr_price | pr_deldays  
-----+-----+-----+-----+-----+-----  
    1 |    1 |    5 |    731 |  405.00 |         7  
    2 |    1 |    6 |   1240 |  690.00 |         9  
    3 |    1 |    7 |   1793 |  519.00 |         7  
    4 |    1 |    8 |      1 |  300.00 |         1  
    5 |    1 |    9 |   432 |  734.00 |        10  
    6 |    1 |   10 |   1220 |  173.00 |         4  
    7 |    1 |   11 |   194 |  462.00 |         9  
    8 |    1 |   12 |   708 |  300.00 |         6  
    9 |    1 |   13 |   947 |  207.00 |         7  
   10 |    1 |   14 |   998 |  394.00 |         2  
  
(10 rows)  
  
mybookshop=#
```

# Скрипты заполнения тестовых данных

--Заказы - таблица зависит от данных в таблицах "Покупатели" и "Номенклатура продаж", сделала случайную генерацию данных по покупателю и выбранному экземпляру книги

DO \$\$

declare

BegDt date;

MaxCust integer;

MaxProd integer;

begin

select max(ct\_id) into MaxCust from Customers;

select max(pr\_id) into MaxProd from Products;

for i in 1..1000 loop

BegDt:= '2023-01-01'::date + floor(random()\*90)::int+1;

--'Create'

insert into Orders(or\_id,ct\_id,pr\_id,or\_date,or\_count,or\_tprice,or\_deldate,or\_status)

select nextval('bookshop.orders\_or\_id\_seq'),c.ct\_id,p.pr\_id,BegDt,least(x.OrdCnt,p.pr\_count) Cnt,

p.pr\_price\*least(x.OrdCnt,p.pr\_count) TPrice,BegDt+p.pr\_deldays,'Create'

from (select floor(random()\*MaxProd)::int+1 ProdID,floor(random()\*MaxCust)::int+1 CustID,floor(random()\*3)::int+1 OrdCnt) x

inner join Products p on p.pr\_id=x.ProdID

inner join Customers c on c.ct\_id=x.CustID

where p.pr\_count>=x.OrdCnt;

--'In delivery'

insert into Orders(or\_id,ct\_id,pr\_id,or\_date,or\_count,or\_tprice,or\_deldate,or\_status)

select nextval('bookshop.orders\_or\_id\_seq'),c.ct\_id,p.pr\_id,BegDt,least(x.OrdCnt,p.pr\_count) Cnt,

p.pr\_price\*least(x.OrdCnt,p.pr\_count) TPrice,BegDt+p.pr\_deldays,'In delivery'

from (select floor(random()\*MaxProd)::int+1 ProdID,floor(random()\*MaxCust)::int+1 CustID,floor(random()\*3)::int+1 OrdCnt) x

inner join Products p on p.pr\_id=x.ProdID

inner join Customers c on c.ct\_id=x.CustID

where p.pr\_count>=x.OrdCnt;



# Скрипты заполнения тестовых данных

--'Done'

```
insert into Orders(or_id,ct_id,pr_id,or_date,or_count,or_tprice,  
                  or_deldate,or_status)
```

```
select nextval('bookshop.orders_or_id_seq'),c.ct_id,p.pr_id,
```

```
    BegDt,least(x.OrdCnt,p.pr_count) Cnt,
```

```
    p.pr_price*least(x.OrdCnt,p.pr_count) TPrice,
```

```
    BegDt+p.pr_deldays,'Done'
```

```
from (select floor(random()*MaxProd)::int+1 ProdID
```

```
    floor(random()*MaxCust)::int+1 CustID,
```

```
    floor(random()*3)::int+1 OrdCnt) x
```

```
    inner join Products p on p.pr_id=x.ProdID
```

```
    inner join Customers c on c.ct_id=x.CustID
```

```
where p.pr_count>=x.OrdCnt;
```

```
end loop;
```

```
end$$;
```

mkalinichenko@mkalinichenko-pr: ~

```
mybookshop=# select * from Orders limit 10;
```

or_id	ct_id	pr_id	or_date	or_count	or_tprice	or_deldate	or_status
1	52	130	2023-03-22 00:00:00	1	595.00	2023-03-30 00:00:00	Create
2	483	128	2023-03-22 00:00:00	3	1635.00	2023-03-27 00:00:00	In delivery
3	751	302	2023-03-22 00:00:00	1	377.00	2023-03-26 00:00:00	Done
4	647	333	2023-01-20 00:00:00	2	974.00	2023-01-26 00:00:00	Create
5	162	324	2023-01-20 00:00:00	3	1260.00	2023-01-23 00:00:00	In delivery
6	809	72	2023-01-20 00:00:00	1	337.00	2023-01-26 00:00:00	Done
7	796	191	2023-02-18 00:00:00	2	1754.00	2023-02-21 00:00:00	Create
8	649	380	2023-02-18 00:00:00	3	1056.00	2023-02-23 00:00:00	In delivery
9	184	398	2023-02-18 00:00:00	1	956.00	2023-02-25 00:00:00	Done
10	497	379	2023-01-10 00:00:00	1	456.00	2023-01-19 00:00:00	Create

(10 rows)

```
mybookshop=# select * from OrdersSummary limit 10;
```

os_date	os_total_ord_cnt	os_done_ord_cnt	os_total_smm	os_done_smm	os_total_bk_cnt	os_done_bk_cnt
2023-02-16	12	4	17375.00	8782.00	27	10
2023-03-25	21	7	26487.00	7826.00	43	16
2023-03-05	51	17	63238.00	18850.00	103	33
2023-02-18	27	9	31748.00	9055.00	58	18
2023-01-12	15	5	16945.00	5364.00	31	8
2023-03-10	24	8	30572.00	9746.00	47	14
2023-02-26	27	9	41806.00	13011.00	60	19
2023-01-04	48	16	57997.00	18440.00	91	29
2023-01-05	15	5	20258.00	4217.00	31	10
2023-03-27	24	8	32663.00	10450.00	56	17

(10 rows)

```
mybookshop=#
```

# Скрипты для тестирования pgbench

По умолчанию программа pgbench может создавать 4 таблицы (pgbench\_accounts, pgbench\_branches, pgbench\_history и pgbench\_tellers), заполнять их данными и прогонять скрипт типа TPC-B для оценки производительности системы.

Но pgbench также позволяет задавать свои собственные скрипты тестирования. Скрипты можно написать на основе своих задач, что позволит более точно оценить производительность конкретной системы с учетом особенностей ее схемы базы данных.

Описание ключей для вызова pgbench с собственным тестовым сценарием:

```
pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /var/tmp/test3.sql -U postgres mybookshop
```

**-c <число>** число имитируемых клиентов, то есть число одновременных сеансов базы данных

**-j <число>** число рабочих потоков в pgbench, полезно на многопроцессорных компьютерах

**-P <число>** выводить отчёт о прогрессе через заданное число секунд (сек)

**-T <число>** выполнять тест с ограничением по времени (в секундах)

**-n** не производить очистку таблиц перед запуском теста. Этот параметр необходим, если применяется собственный сценарий, не затрагивающий стандартные таблицы pgbench\_accounts, pgbench\_branches, pgbench\_history и pgbench\_tellers

**-r** выводить по завершении тестирования среднее время ожидания операторов для каждой команды

**-f <имя файла>** выполнять тест по указанному в файле сценарию

**-U <пользователь>** имя пользователя для подключения

**<база данных>** имя уже существующей базы данных, в которой будет проводиться тест

# Скрипты для тестирования pgbench

Для разработанной схемы "Интернет-магазин книг" сделала три тестовых скрипта для тестирования наиболее встречающихся массовых операций:

1. Первый скрипт тестировал ситуацию «Создание нового заказа».

```
\set OrdCnt random(1,3)
begin;
select *
from (select p.pr_id,c.ct_id,d.OrdDt
      from (select floor(random()*MaxPrID)::int+1 ProdID,floor(random()*MaxCtID)::int+1 CustID,
            (current_date-(floor(random()*90)::int+1)::int)::timestamp OrdDt
            from (select (select max(ct_id) from bookshop.Customers) MaxCtID,(select max(pr_id) from bookshop.Products) MaxPrID) x
            )d
      inner join bookshop.Products p on p.pr_id=d.ProdID
      inner join bookshop.Customers c on c.ct_id=d.CustID
      where p.pr_count>=:OrdCnt
      )w,
bookshop.fNewOrder(w.ct_id,w.pr_id,w.OrdDt,:OrdCnt);
end;
```

Скрипт при работе вызывает функцию БД для создания нового заказа (основное действие Insert)

create or replace function fNewOrder(Cust\_id integer,Prod\_id integer,OrdDt timestamp,OrdCnt integer)

returns bookshop.Orders as \$BODY\$

insert into bookshop.Orders(or\_id,ct\_id,pr\_id,or\_date,or\_count,or\_tprice,or\_deldate,or\_status)

select nextval('bookshop.orders\_or\_id\_seq'),Cust\_id,Prod\_id,OrdDt,OrdCnt,OrdCnt\*p.pr\_price,OrdDt::date+p.pr\_deldays,'Create'

from bookshop.Products p where p.pr\_id=Prod\_id

returning \*;

\$BODY\$ language sql;



# Скрипты для тестирования pgbench

Результаты работы скрипта тестирования (test1.sql) ниже на двух скринах:

```
mkalinichenko@mkalinichenko-pr: ~  
mkalinichenko@mkalinichenko-pr:~$ sudo su postgres  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /var/tmp/test1.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 793.3 tps, lat 12.593 ms stddev 11.866, 0 failed  
progress: 40.0 s, 933.0 tps, lat 10.716 ms stddev 12.202, 0 failed  
progress: 60.0 s, 1094.8 tps, lat 9.133 ms stddev 9.713, 0 failed  
progress: 80.0 s, 1010.0 tps, lat 9.899 ms stddev 16.036, 0 failed  
progress: 100.0 s, 1008.3 tps, lat 9.916 ms stddev 11.412, 0 failed  
progress: 120.0 s, 1189.9 tps, lat 8.404 ms stddev 7.799, 0 failed  
transaction type: /var/tmp/test1.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 120597  
number of failed transactions: 0 (0.000%)  
latency average = 9.949 ms  
latency stddev = 11.711 ms  
initial connection time = 15.351 ms  
tps = 1004.951777 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set OrdCnt random(1,3)  
0.315 0 begin;  
2.632 0 select *  
6.998 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1162.8 tps, lat 8.587 ms stddev 11.377, 0 failed  
progress: 40.0 s, 1263.8 tps, lat 7.915 ms stddev 10.885, 0 failed  
progress: 60.0 s, 1504.4 tps, lat 6.645 ms stddev 6.671, 0 failed  
progress: 80.0 s, 1448.4 tps, lat 6.904 ms stddev 11.084, 0 failed  
progress: 100.0 s, 1582.3 tps, lat 6.319 ms stddev 9.283, 0 failed  
progress: 120.0 s, 1884.4 tps, lat 5.306 ms stddev 6.487, 0 failed  
transaction type: /var/tmp/test1.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 176931  
number of failed transactions: 0 (0.000%)  
latency average = 6.781 ms  
latency stddev = 9.334 ms  
initial connection time = 15.404 ms  
tps = 1474.264649 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set OrdCnt random(1,3)  
0.226 0 begin;  
2.017 0 select *  
4.541 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

По скринам видно, что при повторении теста за те же 2 минуты выросло количество выполненных транзакций и показатель tps.

# Скрипты для тестирования pgbench

2. Второй скрипт тестировал ситуацию «инвентаризация наличия книг на складе».

```
\set ProdCnt random(800,2500)
begin;
  select *
  from (select p.pr_id
        from (select floor(random()*MaxPrID)::int+1 ProdID
              from (select (select max(pr_id) from bookshop.Products) MaxPrID) x
              )d
        inner join bookshop.Products p on p.pr_id=d.ProdID
      )w,
  bookshop.fModifProducts(w.pr_id,:ProdCnt);
end;
```

Скрипт при работе вызывает функцию БД для инвентаризации кол-ва книг на складе (основное действие Update)

```
create or replace function fModifProducts(ProdID integer,ProdCnt integer)
returns table(st_name varchar(500),bk_name varchar(500),bk_author varchar(500),pr_count integer,
            pr_price numeric(9,2),pr_deldays integer,pr_id integer,st_id integer,bk_id integer)
as $BODY$
  update bookshop.Products set pr_count=ProdCnt where pr_id=ProdID;

  select s.st_name,b.bk_name,b.bk_author,pr_count,pr_price,pr_deldays,p.pr_id,p.st_id,p.bk_id
  from bookshop.Products p,bookshop.Storages s,bookshop.Books b
  where p.st_id=s.st_id and p.bk_id=b.bk_id
  and p.pr_id=ProdID;
$BODY$ language sql;
```



# Скрипты для тестирования pgbench

Результаты работы скрипта тестирования (test2.sql) ниже на двух скринах:

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /var/tmp/test2.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1255.7 tps, lat 7.955 ms stddev 7.141, 0 failed  
progress: 40.0 s, 799.1 tps, lat 12.510 ms stddev 17.933, 0 failed  
progress: 60.0 s, 825.5 tps, lat 12.117 ms stddev 15.558, 0 failed  
progress: 80.0 s, 1114.9 tps, lat 8.968 ms stddev 9.483, 0 failed  
progress: 100.0 s, 1038.7 tps, lat 9.623 ms stddev 8.610, 0 failed  
progress: 120.0 s, 871.1 tps, lat 11.482 ms stddev 13.032, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 118110  
number of failed transactions: 0 (0.000%)  
latency average = 10.158 ms  
latency stddev = 12.090 ms  
initial connection time = 15.244 ms  
tps = 984.256980 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.315 0 begin;  
1.547 0 select *  
8.332 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1022.2 tps, lat 9.772 ms stddev 9.571, 0 failed  
progress: 40.0 s, 1051.2 tps, lat 9.511 ms stddev 9.036, 0 failed  
progress: 60.0 s, 1225.0 tps, lat 8.163 ms stddev 5.155, 0 failed  
progress: 80.0 s, 1154.8 tps, lat 8.658 ms stddev 10.166, 0 failed  
progress: 100.0 s, 1070.7 tps, lat 9.338 ms stddev 7.249, 0 failed  
progress: 120.0 s, 1182.1 tps, lat 8.461 ms stddev 7.205, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 134128  
number of failed transactions: 0 (0.000%)  
latency average = 8.945 ms  
latency stddev = 8.204 ms  
initial connection time = 15.271 ms  
tps = 1117.694773 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.302 0 begin;  
1.500 0 select *  
7.164 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

По скринам видно, что при повторении теста также выросло количество выполненных транзакций и показатель tps.

# Скрипты для тестирования rgbench

3. Третий скрипт тестировал ситуацию «поиска книги и получение подробной информации по стоимости и времени доставки в зависимости от склада».

```
begin;  
  select *  
  from (select b.bk_id  
        from (select floor(random()*MaxBkID)::int+1 BookID  
              from (select (select max(bk_id) from bookshop.Books) MaxBkID) x  
            )d  
        inner join bookshop.Books b on b.bk_id=d.BookID  
      )w,  
  bookshop.fSearchBook(w.bk_id);  
end;
```

Скрипт при работе вызывает функцию БД для поиска книги (основное действие Select)

```
create or replace function fSearchBook(BookID integer)  
returns table(bk_name varchar(500),bk_author varchar(500),bk_year varchar(30),bk_publish varchar(500),bk_annotation varchar(4000),  
             pr_count integer,pr_price numeric(9,2),pr_deldays integer, st_name varchar(500),st_address varchar(1000),st_phone varchar(100))  
as $BODY$  
  select b.bk_name,b.bk_author,b.bk_year,b.bk_publish,b.bk_annotation, p.pr_count,p.pr_price,p.pr_deldays, s.st_name,s.st_address,s.st_phone  
  from bookshop.Products p,bookshop.Books b,bookshop.Storages s  
  where p.bk_id=b.bk_id and p.st_id=s.st_id and p.bk_id=BookID;  
$BODY$  
language sql;
```



# Скрипты для тестирования pgbench

Результаты работы скрипта тестирования (test3.sql) ниже на двух скринах:

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /var/tmp/test3.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 3235.7 tps, lat 3.087 ms stddev 0.628, 0 failed  
progress: 40.0 s, 3229.0 tps, lat 3.096 ms stddev 0.567, 0 failed  
progress: 60.0 s, 3247.5 tps, lat 3.078 ms stddev 0.427, 0 failed  
progress: 80.0 s, 3259.8 tps, lat 3.067 ms stddev 0.387, 0 failed  
progress: 100.0 s, 3241.8 tps, lat 3.084 ms stddev 0.422, 0 failed  
progress: 120.0 s, 3212.8 tps, lat 3.112 ms stddev 0.574, 0 failed  
transaction type: /var/tmp/test3.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 388541  
number of failed transactions: 0 (0.000%)  
latency average = 3.087 ms  
latency stddev = 0.509 ms  
initial connection time = 15.322 ms  
tps = 3237.978444 (without initial connection time)  
statement latencies in milliseconds and failures:  
    0.153      0  begin;  
    0.913      0  select *  
    2.030      0  end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

По скринам видно, что при повторении теста показатели тестирования существенно не поменялись.

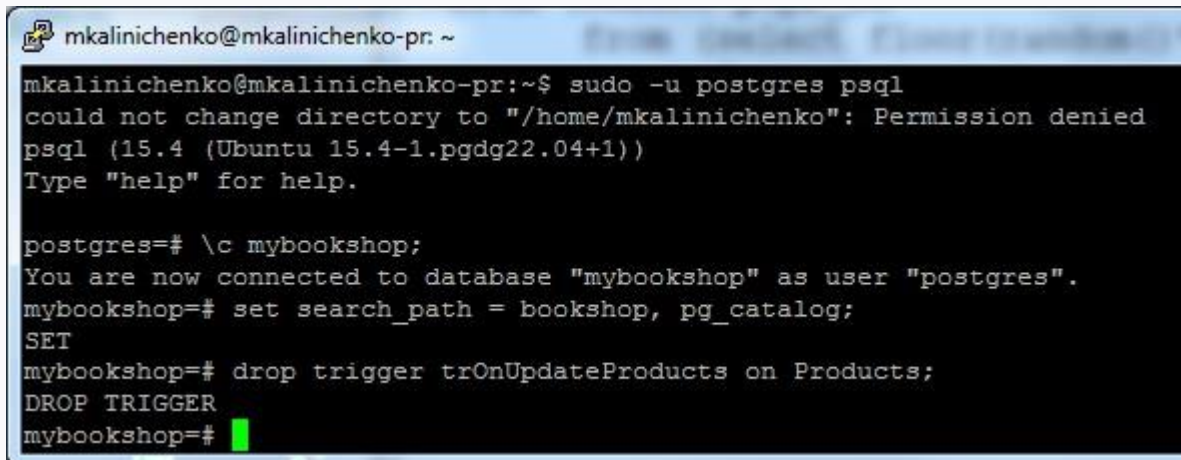
```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 3252.0 tps, lat 3.072 ms stddev 0.654, 0 failed  
progress: 40.0 s, 3262.3 tps, lat 3.065 ms stddev 0.529, 0 failed  
progress: 60.0 s, 3259.8 tps, lat 3.067 ms stddev 0.595, 0 failed  
progress: 80.0 s, 3263.8 tps, lat 3.063 ms stddev 0.625, 0 failed  
progress: 100.0 s, 3272.1 tps, lat 3.055 ms stddev 0.592, 0 failed  
progress: 120.0 s, 3236.0 tps, lat 3.089 ms stddev 0.579, 0 failed  
transaction type: /var/tmp/test3.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 390930  
number of failed transactions: 0 (0.000%)  
latency average = 3.069 ms  
latency stddev = 0.597 ms  
initial connection time = 15.167 ms  
tps = 3257.868233 (without initial connection time)  
statement latencies in milliseconds and failures:  
    0.183      0  begin;  
    1.072      0  select *  
    1.821      0  end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

# Скрипты для тестирования pgbench

Самый медленный тест получился по инвентаризации наличия книг на складе, т.е. по обновлению таблицы "номенклатура продаж".

На данной таблице висят триггеры, которые обеспечивают заполнение "витрины наличия книг".

Для оценки влияния их работы на общую скорость отключила триггер trOnUpdateProducts (на update таблицы Products):

A terminal window with a black background and white text. The prompt is 'mkalinichenko@mkalinichenko-pr: ~'. The user enters 'sudo -u postgres psql'. The output shows a permission error for the directory, then the psql prompt. The user enters '\c mybookshop;', followed by 'set search\_path = bookshop, pg\_catalog;', then 'drop trigger trOnUpdateProducts on Products;'. The output shows 'SET' and 'DROP TRIGGER'. The prompt returns to 'mybookshop=#' with a green cursor.

```
mkalinichenko@mkalinichenko-pr: ~  
mkalinichenko@mkalinichenko-pr:~$ sudo -u postgres psql  
could not change directory to "/home/mkalinichenko": Permission denied  
psql (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
Type "help" for help.  
  
postgres=# \c mybookshop;  
You are now connected to database "mybookshop" as user "postgres".  
mybookshop=# set search_path = bookshop, pg_catalog;  
SET  
mybookshop=# drop trigger trOnUpdateProducts on Products;  
DROP TRIGGER  
mybookshop=#
```



# Скрипты для тестирования pgbench

Сравним результаты работы скрипта тестирования (test2.sql) при включенном триггере (слева) и отключенном (справа):

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /var/tmp/test2.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1255.7 tps, lat 7.955 ms stddev 7.141, 0 failed  
progress: 40.0 s, 799.1 tps, lat 12.510 ms stddev 17.933, 0 failed  
progress: 60.0 s, 825.5 tps, lat 12.117 ms stddev 15.558, 0 failed  
progress: 80.0 s, 1114.9 tps, lat 8.968 ms stddev 9.483, 0 failed  
progress: 100.0 s, 1038.7 tps, lat 9.623 ms stddev 8.610, 0 failed  
progress: 120.0 s, 871.1 tps, lat 11.482 ms stddev 13.032, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 118110  
number of failed transactions: 0 (0.000%)  
latency average = 10.158 ms  
latency stddev = 12.090 ms  
initial connection time = 15.244 ms  
tps = 984.256980 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.315 0 begin;  
1.547 0 select *  
8.332 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

Тест при выключенном триггере показал улучшение производительности примерно на 20%.

```
mkalinichenko@mkalinichenko-pr: ~  
mkalinichenko@mkalinichenko-pr:~$ sudo su postgres  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1346.6 tps, lat 7.417 ms stddev 8.431, 0 failed  
progress: 40.0 s, 1418.9 tps, lat 7.026 ms stddev 7.643, 0 failed  
progress: 60.0 s, 1335.7 tps, lat 7.508 ms stddev 6.522, 0 failed  
progress: 80.0 s, 1282.5 tps, lat 7.793 ms stddev 10.086, 0 failed  
progress: 100.0 s, 1331.7 tps, lat 7.508 ms stddev 4.926, 0 failed  
progress: 120.0 s, 1115.7 tps, lat 8.967 ms stddev 8.175, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 156631  
number of failed transactions: 0 (0.000%)  
latency average = 7.660 ms  
latency stddev = 7.792 ms  
initial connection time = 15.263 ms  
tps = 1305.226877 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.249 0 begin;  
1.022 0 select *  
6.402 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```



# Скрипты для тестирования pgbench

Посмотрела параметры PostgreSQL в файле /etc/postgresql/15/main/postgresql.conf. Попробовала изменить настройки системы:

```
mkalinichenko@mkalinichenko-pr: ~  
mkalinichenko@mkalinichenko-pr:~$ sudo su postgres  
postgres@mkalinichenko-pr:/home/mkalinichenko$ cp /etc/postgresql/15/main/postgresql.conf /etc/postgresql/15/main/postgresql_prev.conf  
postgres@mkalinichenko-pr:/home/mkalinichenko$ nano /etc/postgresql/15/main/postgresql.conf  
postgres@mkalinichenko-pr:/home/mkalinichenko$ cd ~  
postgres@mkalinichenko-pr:~$ exit  
mkalinichenko@mkalinichenko-pr:~$ sudo pg_ctlcluster 15 main restart  
mkalinichenko@mkalinichenko-pr:~$ pg_lsclusters  
Ver Cluster Port Status Owner    Data directory          Log file  
15  main    5432  online postgres /var/lib/postgresql/15/main /var/log/postgresql/postgresql-15-main.log  
mkalinichenko@mkalinichenko-pr:~$
```

**shared\_buffers** - используется для кеширования данных, чем больше данных в памяти, тем выше скорость.

**work\_mem** - используется для сортировок, построения hash таблиц, увеличение размера позволяет выполнять данные операции в памяти, что также повышает скорость.

**random\_page\_cost** - определяет стоимость, которую будет иметь страница диска с непоследовательной выборкой, и напрямую влияет на решения планировщика запросов. Т.к. у нас SSD диск, то можно поставить "1"

**min\_wal\_size, max\_wal\_size** - управляют контрольными точками, изменила параметры так, чтобы растянуть время между контрольными точками, уменьшить запись изменений во время теста, снизить нагрузку на дисковую подсистему.

```
GNU nano 6.2 /etc/postgresql/15/main/postgresql.conf  
# - Memory -  
shared_buffers = 1024MB          # min 128kB  
  
# - Checkpoints -  
#checkpoint_timeout = 5min        # range 30s-1d  
#checkpoint_completion_target = 0.9 # checkpoint target duration, 0.0 - 1.0  
#checkpoint_flush_after = 256kB    # measured in pages, 0 disables  
#checkpoint_warning = 30s         # 0 disables  
max_wal_size = 2GB  
min_wal_size = 80MB  
  
# Caution: it is not advisable to set max_prepared_transactions nonzero unless  
# you actively intend to use prepared transactions.  
work_mem = 128MB                 # min 64kB  
#hash_mem_multiplier = 2.0        # 1-1000.0 multiplier on hash table work_mem  
#maintenance_work_mem = 64MB      # min 1MB  
#autovacuum_work_mem = -1          # min 1MB, or -1 to use maintenance_work_mem  
#logical_decoding_work_mem = 64MB # min 64kB  
#max_stack_depth = 2MB            # min 100kB  
#shared_memory_type = mmap        # the default is the first option  
#                                # supported by the operating system:  
#                                # same scale as above  
  
random_page_cost = 1
```

# Скрипты для тестирования pgbench

Изменив вышеперечисленные настройки, запустила тестирование pgbench тесте 1 «создание нового заказа». Сравним результаты работы скрипта тестирования (test1.sql) на старых настройках (слева) и с новыми настройками памяти (справа):

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1162.8 tps, lat 8.587 ms stddev 11.377, 0 failed  
progress: 40.0 s, 1263.8 tps, lat 7.915 ms stddev 10.885, 0 failed  
progress: 60.0 s, 1504.4 tps, lat 6.645 ms stddev 6.671, 0 failed  
progress: 80.0 s, 1448.4 tps, lat 6.904 ms stddev 11.084, 0 failed  
progress: 100.0 s, 1582.3 tps, lat 6.319 ms stddev 9.283, 0 failed  
progress: 120.0 s, 1884.4 tps, lat 5.306 ms stddev 6.487, 0 failed  
transaction type: /var/tmp/test1.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 176931  
number of failed transactions: 0 (0.000%)  
latency average = 6.781 ms  
latency stddev = 9.334 ms  
initial connection time = 15.404 ms  
tps = 1474.264649 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set OrdCnt random(1,3)  
0.226 0 begin;  
2.017 0 select *  
4.541 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:~$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f /  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1397.5 tps, lat 7.147 ms stddev 9.085, 0 failed  
progress: 40.0 s, 1416.6 tps, lat 7.058 ms stddev 8.996, 0 failed  
progress: 60.0 s, 1626.5 tps, lat 6.147 ms stddev 9.329, 0 failed  
progress: 80.0 s, 1754.8 tps, lat 5.698 ms stddev 8.892, 0 failed  
progress: 100.0 s, 1880.4 tps, lat 5.316 ms stddev 8.035, 0 failed  
progress: 120.0 s, 2263.8 tps, lat 4.417 ms stddev 7.604, 0 failed  
transaction type: /var/tmp/test1.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 206802  
number of failed transactions: 0 (0.000%)  
latency average = 5.801 ms  
latency stddev = 8.643 ms  
initial connection time = 15.733 ms  
tps = 1723.233596 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set OrdCnt random(1,3)  
0.212 0 begin;  
1.655 0 select *  
3.935 0 end;  
postgres@mkalinichenko-pr:~$
```

Изменив настройки системы для работы с памятью по умолчанию, по тесту 1 один мы получили повышение производительности примерно на 20% для операции добавления новых данных.



# Скрипты для тестирования pgbench

Проверила производительность системы при тестировании pgbench тесте 2 по обновлению таблицы "номенклатура продаж". Сравним результаты работы скрипта тестирования (test2.sql) на старых настройках (слева) и с новыми настройками памяти (справа):

```
mkalinichenko@mkalinichenko-pr: ~  
mkalinichenko@mkalinichenko-pr:~$ sudo su postgres  
postgres@mkalinichenko-pr:/home/mkalinichenko$ pgbench -c 10 -j 2 -P 20  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1346.6 tps, lat 7.417 ms stddev 8.431, 0 failed  
progress: 40.0 s, 1418.9 tps, lat 7.026 ms stddev 7.643, 0 failed  
progress: 60.0 s, 1335.7 tps, lat 7.508 ms stddev 6.522, 0 failed  
progress: 80.0 s, 1282.5 tps, lat 7.793 ms stddev 10.086, 0 failed  
progress: 100.0 s, 1331.7 tps, lat 7.508 ms stddev 4.926, 0 failed  
progress: 120.0 s, 1115.7 tps, lat 8.967 ms stddev 8.175, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 156631  
number of failed transactions: 0 (0.000%)  
latency average = 7.660 ms  
latency stddev = 7.792 ms  
initial connection time = 15.263 ms  
tps = 1305.226877 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.249 0 begin;  
1.022 0 select *  
6.402 0 end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

```
mkalinichenko@mkalinichenko-pr: ~  
postgres@mkalinichenko-pr:~$ pgbench -c 10 -j 2 -P 20 -T 120 -n -r -f  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 1464.5 tps, lat 6.820 ms stddev 8.667, 0 failed  
progress: 40.0 s, 1553.9 tps, lat 6.434 ms stddev 4.205, 0 failed  
progress: 60.0 s, 1135.6 tps, lat 8.801 ms stddev 7.770, 0 failed  
progress: 80.0 s, 1044.9 tps, lat 9.574 ms stddev 13.915, 0 failed  
progress: 100.0 s, 1529.9 tps, lat 6.535 ms stddev 7.546, 0 failed  
progress: 120.0 s, 1474.3 tps, lat 6.781 ms stddev 7.831, 0 failed  
transaction type: /var/tmp/test2.sql  
scaling factor: 1  
query mode: simple  
number of clients: 10  
number of threads: 2  
maximum number of tries: 1  
duration: 120 s  
number of transactions actually processed: 164073  
number of failed transactions: 0 (0.000%)  
latency average = 7.312 ms  
latency stddev = 8.528 ms  
initial connection time = 15.675 ms  
tps = 1367.284309 (without initial connection time)  
statement latencies in milliseconds and failures:  
0.001 0 \set ProdCnt random(800,2500)  
0.246 0 begin;  
0.975 0 select *  
6.094 0 end;  
postgres@mkalinichenko-pr:~$
```

По результатам теста видно, что произведенные настройки системы не оказали существенного влияния на тест по обновлению таблицы "номенклатура продаж".





Спасибо за внимание!

Калиниченко Майя Евгеньевна

Должность: Старший инженер

Компания: ПАО «Ростелеком»