



Защита проекта

Тема: Создание БД для проекта "Интернет-магазин книг" и тестирование его работы с использованием rgbench



Калиниченко Майя Евгеньевна

Должность: Старший инженер

Компания: ПАО «Ростелеком»

Введение

Целью данного проекта является изучение особенностей работы программы `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



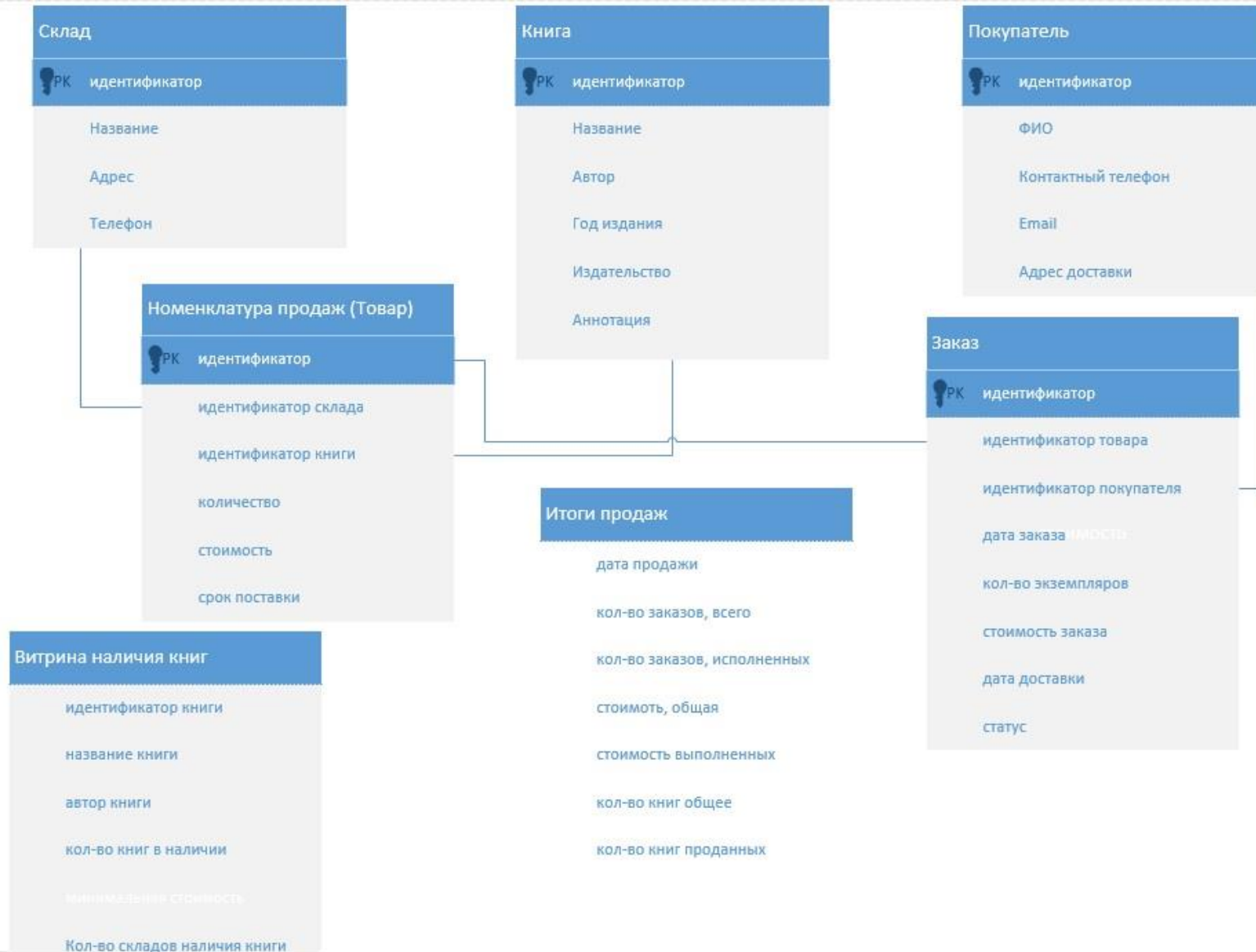
Что получилось

Репозиторий с текущей версией проекта

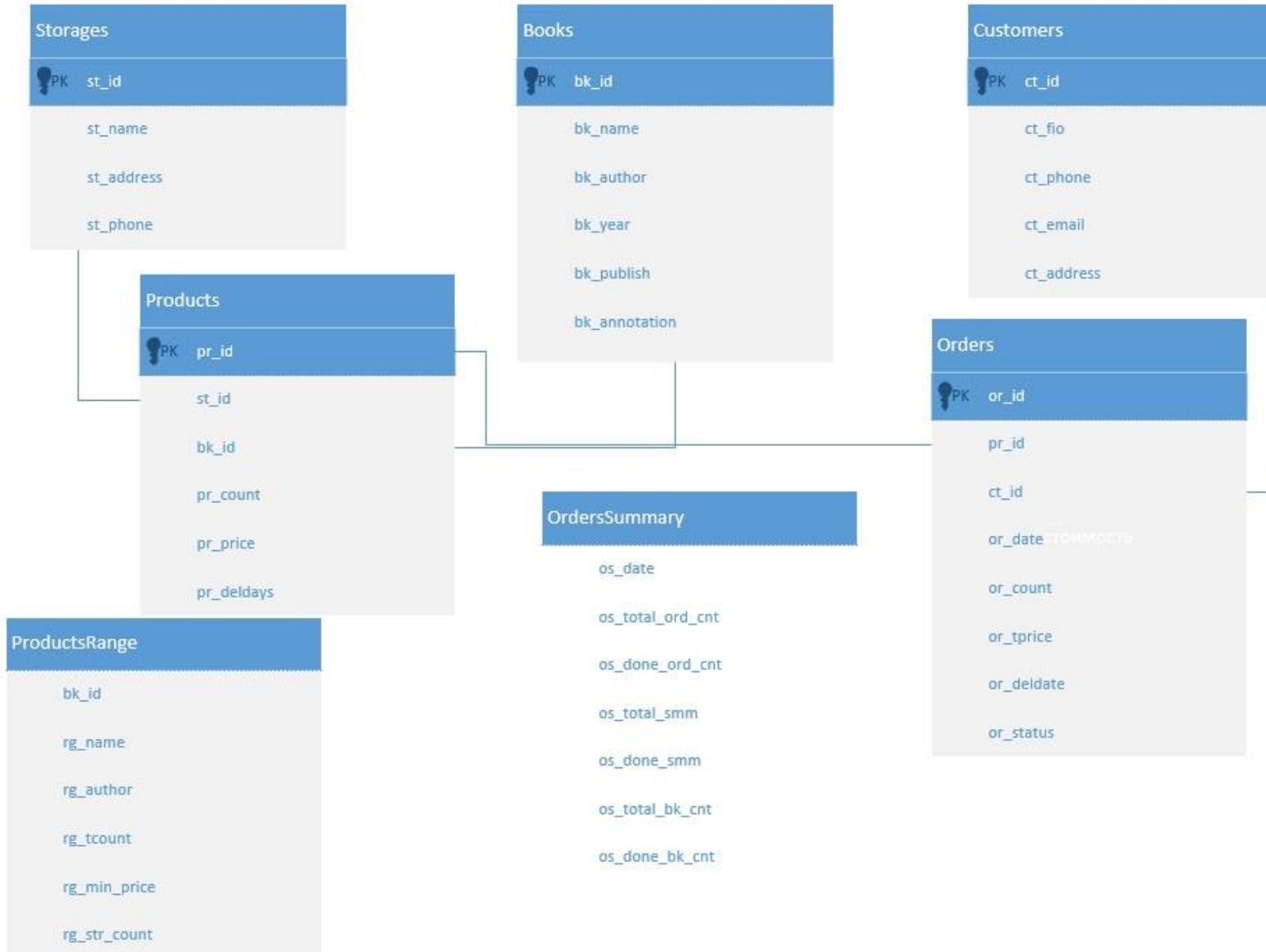
https://github.com/mkalinichenko2023/otus_project/blob/main/README.md

- 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 Customes_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 $BODY$
declare
begin
    if Old.ct_id <> New.ct_id or Old.pr_id <> New.pr_id then
        RAISE EXCEPTION 'Обновления для столбцов "ct_id" и "pr_id" запрещены!';
    end if;
return New;
end; $BODY$
language plpgsql;
```

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

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
    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;
return New;
end; $BODY$
language plpgsql;
```

Создание триггерных функций для поддержания витрин в актуальном состоянии

```
create or replace function fOnRemOrders()
returns trigger as $BODY$
declare
    ord_cnt    integer;
begin
    begin
        select os_total_ord_cnt into ord_cnt from bookshop.OrdersSummary os where os.os_date= Old.or_date::date;
        if ord_cnt>1 then
            update bookshop.OrdersSummary
            set os_total_ord_cnt= os_total_ord_cnt-1, os_total_smm= os_total_smm-Old.or_tprice, os_total_bk_cnt=os_total_bk_cnt-Old.or_count
            where os_date=New.or_date::date;
            if Old.or_status='Done' then
                update bookshop.OrdersSummary
                set os_done_ord_cnt= os_done_ord_cnt-1, os_done_smm= os_done_smm-Old.or_tprice, os_done_bk_cnt= os_done_bk_cnt-Old.or_count
                where os_date=New.or_date::date;
            end if;
        else
            delete from bookshop.OrdersSummary os where os.os_date= Old.or_date::date;
        end if;
    exception when no_data_found then
        ord_cnt:=0;
    end;
    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
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;
return New;
end; $BODY$
language plpgsql;
```

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

```
create trigger trOnDeletetOrders
after delete on Orders for each row execute function fOnRemOrders();
```

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


Скрипты заполнения тестовых данных

--заполнение данными

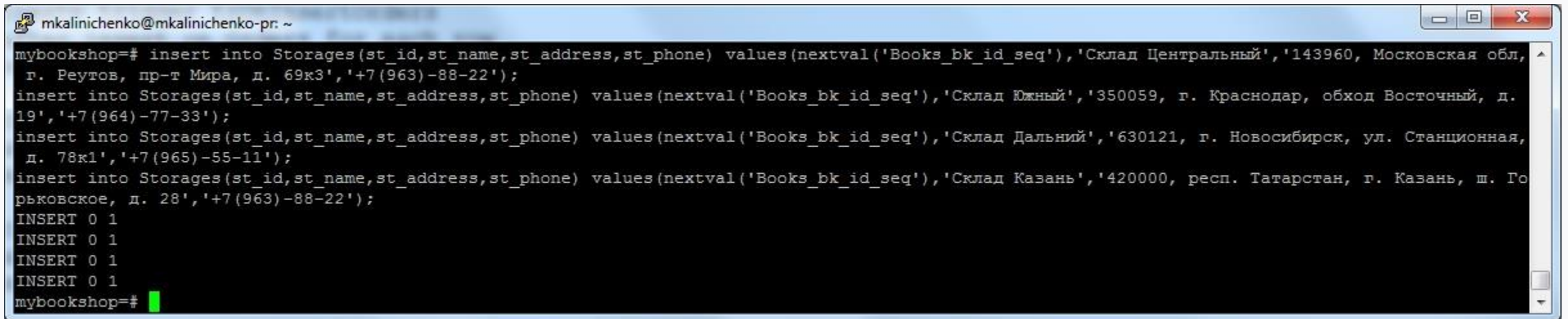
--Склады - небольшая таблица, сделала ручную вставку значений по смыслу

```
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');
```



```
mksalinichenko@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

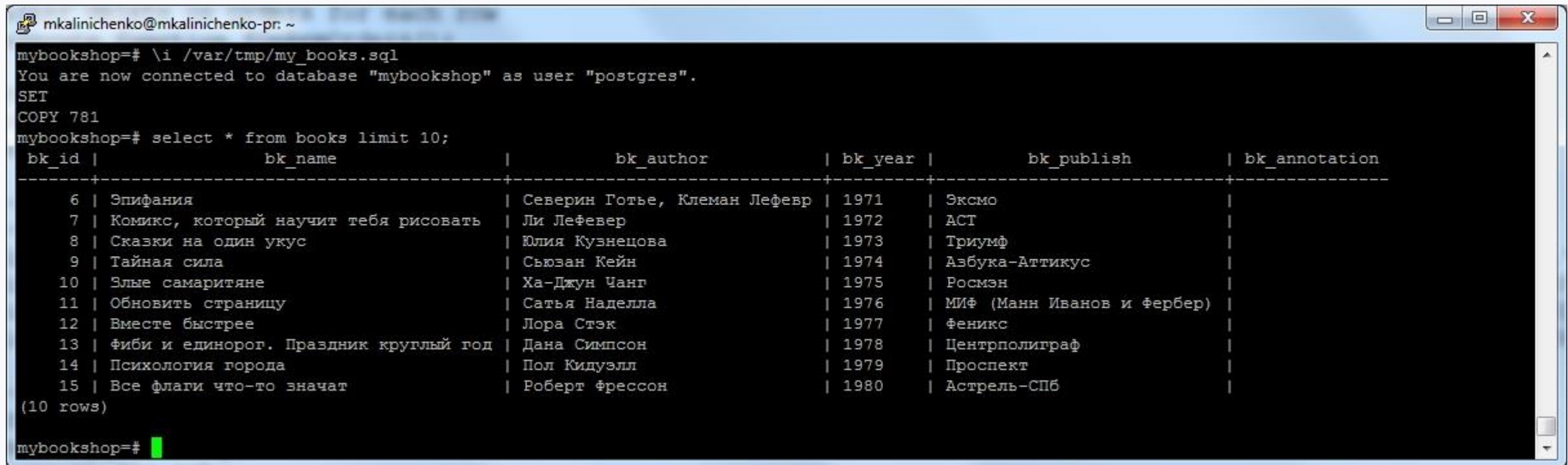
```
cd /var/tmp/
```

```
nano my_books.sql
```

```
sudo -u postgres psql
```

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

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



```
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;
```

```
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()*20)+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: ~  
mybookshop=# DO $$  
declare  
  Rec record;  
begin  
  for Rec in (select st_id from Storages)loop  
    insert into Products(pr_id,st_id,bk_id,pr_count,pr_price,pr_deldays)  
      select nextval('products_pr_id_seq'),Rec.st_id,b.bk_id,floor  
      from Books b where mod(b.bk_id,Rec.st_id)=0 limit 100;  
    end loop;  
end$$;  
DO  
mybookshop=# select * from Products limit 10;  
 pr_id | st_id | bk_id | pr_count | pr_price | pr_deldays  
-----+-----+-----+-----+-----+-----  
  201 |    1 |    6 |        16 |   120.00 |         1  
  202 |    1 |    7 |        13 |   364.00 |         8  
  203 |    1 |    8 |         1 |   751.00 |         6  
  204 |    1 |    9 |        17 |   556.00 |         2  
  205 |    1 |   10 |         5 |   706.00 |         3  
  206 |    1 |   11 |        15 |   297.00 |         3  
  207 |    1 |   12 |        10 |   945.00 |         9  
  208 |    1 |   13 |        19 |   374.00 |         8  
  209 |    1 |   14 |        11 |   256.00 |         9  
  210 |    1 |   15 |        17 |   649.00 |         9  
(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

Rec record;

BegDt date:= '2023-01-01';

CorDt integer;

DelDt integer;

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

CorDt:= floor(random()*90)+1;

DelDt:= CorDt + floor(random()*15)+1;

--'Create'

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

select nextval('orders_or_id_seq'),xc.ct_id,xp.pr_id,BegDt+CorDt,xp.Cnt,xp.pr_price*Cnt,BegDt+DelDt,'Create'

from (select p.pr_id,p.pr_price,floor(random()*5)+1 Cnt from Products p where p.pr_id=floor(random()*MaxProd)+1) xp,
(select ct_id from Customers where ct_id=floor(random()*MaxCust)+1) xc;

--'In delivery'

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

select nextval('orders_or_id_seq'),xc.ct_id,xp.pr_id,BegDt+CorDt,xp.Cnt,xp.pr_price*Cnt,BegDt+DelDt,'In delivery'

from (select p.pr_id,p.pr_price,floor(random()*5)+1 Cnt from Products p where p.pr_id=floor(random()*MaxProd)+1) xp,
(select ct_id from Customers where ct_id=floor(random()*MaxCust)+1) xc;

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('orders_or_id_seq'),xc.ct_id,xp.pr_id,  
       BegDt+CorDt,xp.Cnt,xp.pr_price*Cnt,BegDt+DelDt,'Done'  
from (select p.pr_id,p.pr_price,floor(random()*5)+1 Cnt  
      from Products p  
      where p.pr_id=floor(random()*MaxProd)+1) xp,  
     (select ct_id from Customers  
      where ct_id=floor(random()*MaxCust)+1) xc;  
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 | 321 | 202 | 2023-02-27 00:00:00 | 1 | 364.00 | 2023-03-01 00:00:00 | Create  
 2 | 579 | 202 | 2023-02-27 00:00:00 | 1 | 364.00 | 2023-03-01 00:00:00 | Create  
 3 | 835 | 202 | 2023-02-27 00:00:00 | 1 | 364.00 | 2023-03-01 00:00:00 | Create  
 4 | 942 | 202 | 2023-02-27 00:00:00 | 1 | 364.00 | 2023-03-01 00:00:00 | Create  
 5 | 21 | 201 | 2023-01-31 00:00:00 | 5 | 600.00 | 2023-02-05 00:00:00 | Done  
 6 | 544 | 312 | 2023-03-12 00:00:00 | 3 | 2730.00 | 2023-03-13 00:00:00 | In delivery  
 7 | 196 | 382 | 2023-02-19 00:00:00 | 4 | 1536.00 | 2023-02-24 00:00:00 | Create  
 8 | 227 | 382 | 2023-02-19 00:00:00 | 4 | 1536.00 | 2023-02-24 00:00:00 | Create  
 9 | 531 | 382 | 2023-02-19 00:00:00 | 4 | 1536.00 | 2023-02-24 00:00:00 | Create  
10 | 818 | 382 | 2023-02-19 00:00:00 | 4 | 1536.00 | 2023-02-24 00:00:00 | Create  
(10 rows)  
  
mybookshop=# select or_date,or_status,count(distinct ct_id) cst,count(distinct pr_id) prd  
from Orders group by or_date,or_status order by or_date,or_status limit 20;  
 or_date | or_status | cst | prd  
-----+-----+-----+-----  
2023-01-02 00:00:00 | Create | 5 | 2  
2023-01-02 00:00:00 | Done | 8 | 5  
2023-01-02 00:00:00 | In delivery | 4 | 2  
2023-01-03 00:00:00 | Create | 6 | 5  
2023-01-03 00:00:00 | Done | 3 | 6  
2023-01-03 00:00:00 | In delivery | 2 | 2  
2023-01-04 00:00:00 | Create | 10 | 9  
2023-01-04 00:00:00 | Done | 5 | 2  
2023-01-04 00:00:00 | In delivery | 9 | 7  
2023-01-05 00:00:00 | Create | 6 | 5  
2023-01-05 00:00:00 | Done | 9 | 6  
2023-01-05 00:00:00 | In delivery | 2 | 2  
2023-01-06 00:00:00 | Create | 9 | 5  
2023-01-06 00:00:00 | Done | 6 | 4  
2023-01-06 00:00:00 | In delivery | 9 | 5  
2023-01-07 00:00:00 | Create | 9 | 7  
2023-01-07 00:00:00 | Done | 7 | 5  
2023-01-07 00:00:00 | In delivery | 4 | 3  
2023-01-08 00:00:00 | Create | 1 | 1  
2023-01-08 00:00:00 | Done | 2 | 1  
(20 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 программа pgbench позволяет задавать свои собственные скрипты тестирования. Скрипты можно написать на основе своих таблиц, что позволит более точно оценить производительность конкретной системы с учетом особенностей ее схемы базы данных.

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

1. Создание нового заказа.

```
\set OrdCnt random(1,10)  
begin;  
  select * from (select floor(random()*max(ct_id))+1 id from bookshop.Customers) c,  
    (select floor(random()*max(pr_id))+1 id from bookshop.Products) p,  
    bookshop.fNewOrder(c.id::int,p.id::int,(current_date-(floor(random()*90)+1)::int)::timestamp,:OrdCnt);  
end;
```

--Функция 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

Вызов скрипта тестирования:

```
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, 2037.0 tps, lat 4.899 ms stddev 5.569, 0 failed  
progress: 40.0 s, 2186.8 tps, lat 4.578 ms stddev 4.168, 0 failed  
progress: 60.0 s, 1852.4 tps, lat 5.387 ms stddev 6.858, 0 failed  
progress: 80.0 s, 1497.5 tps, lat 6.690 ms stddev 8.795, 0 failed  
progress: 100.0 s, 2107.8 tps, lat 4.744 ms stddev 4.868, 0 failed  
progress: 120.0 s, 2143.3 tps, lat 4.665 ms stddev 4.784, 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: 236503  
number of failed transactions: 0 (0.000%)  
latency average = 5.073 ms  
latency stddev = 5.877 ms  
initial connection time = 11.605 ms  
tps = 1970.852289 (without initial connection time)  
statement latencies in milliseconds and failures:  
    0.001      0  \set OrdCnt random(1,10)  
    0.157      0  begin;  
    1.062      0  select * from (select floor(random()*max(ct_id))+1 id from bookshop.Customers) c,  
    3.854      0  end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```


Скрипты для тестирования pgbench

2. Инвентаризация наличия книг на складе.

```
\set ProdCnt random(1,499)
begin;
  select * from (select floor(random()*max(pr_id))+1 id from bookshop.Products) p,bookshop.fModifProducts(p.id::int,:ProdCnt);
end;
```

--Функция 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

Вызов скрипта тестирования:

```
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/test2.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 2311.9 tps, lat 4.321 ms stddev 4.326, 0 failed  
progress: 40.0 s, 2183.4 tps, lat 4.577 ms stddev 4.895, 0 failed  
progress: 60.0 s, 2303.7 tps, lat 4.343 ms stddev 4.918, 0 failed  
progress: 80.0 s, 1671.4 tps, lat 5.980 ms stddev 5.823, 0 failed  
progress: 100.0 s, 999.5 tps, lat 10.008 ms stddev 9.977, 0 failed  
progress: 120.0 s, 1553.1 tps, lat 6.438 ms stddev 7.020, 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: 220468  
number of failed transactions: 0 (0.000%)  
latency average = 5.442 ms  
latency stddev = 6.130 ms  
initial connection time = 12.165 ms  
tps = 1837.268793 (without initial connection time)  
statement latencies in milliseconds and failures:  
    0.001      0  \set ProdCnt random(1,499)  
    0.154      0  begin;  
    0.883      0  select * from (select floor(random()*max(pr_id))+1 id from bookshop.Products) p,bookshop.fModifProducts(p.id::int,:ProdCnt  
) ;  
    4.409      0  end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

Скрипты для тестирования pgbench

2. Поиск книги и получение подробной информации по стоимости и времени доставки в зависимости от склада.

```
begin;  
  select * from (select floor(random()*max(bk_id))+1 id from bookshop.Books) b,bookshop.fSearchBook(b.id::int);  
end;
```

--Функция 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

Вызов скрипта тестирования:

```
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/test3.sql -U postgres mybookshop  
pgbench (15.4 (Ubuntu 15.4-1.pgdg22.04+1))  
progress: 20.0 s, 4936.6 tps, lat 2.024 ms stddev 0.407, 0 failed  
progress: 40.0 s, 4886.3 tps, lat 2.046 ms stddev 0.416, 0 failed  
progress: 60.0 s, 4901.3 tps, lat 2.040 ms stddev 0.354, 0 failed  
progress: 80.0 s, 4913.7 tps, lat 2.035 ms stddev 0.361, 0 failed  
progress: 100.0 s, 4907.4 tps, lat 2.037 ms stddev 0.215, 0 failed  
progress: 120.0 s, 4934.0 tps, lat 2.026 ms stddev 0.218, 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: 589596  
number of failed transactions: 0 (0.000%)  
latency average = 2.035 ms  
latency stddev = 0.339 ms  
initial connection time = 11.652 ms  
tps = 4913.386885 (without initial connection time)  
statement latencies in milliseconds and failures:  
    0.117      0  begin;  
    0.630      0  select * from (select floor(random()*max(bk_id))+1 id from bookshop.Books) b,bookshop.fSearchBook(b.id::int);  
    1.293      0  end;  
postgres@mkalinichenko-pr:/home/mkalinichenko$
```

Скрипты для тестирования pgbench

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

The background of the slide features an aerial view of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that contains a white network pattern of interconnected dots and lines, resembling a digital or communication network. The text is centered in the middle of the slide.

Спасибо за внимание!

Калиниченко Майя Евгеньевна

Должность: Старший инженер

Компания: ПАО «Ростелеком»