```
-- sudo pg_ctlcluster 16 main start
CREATE DATABASE wb;
\l
\c wb
CREATE SCHEMA eugene;
CREATE TABLE eugene.test (i int);
\dt+
\dt+ eugene.*
\d+ eugene.test
insert into eugene.test values (1);

\c postgres
\dt+ eugene.*

-- весь текст приводится к нижнему регистру
sELEcT * from test;
sELEcT * from eugene.test;

-- НЕ используйте принужительно разный регистр/пробелы/русские названия
CREATE TABLE "te S t" (i int);


-- кроме выборок возможны и логически операции
SELECT 1+1;


BEGIN;
DROP TABLE IF EXISTS warehouse CASCADE;
CREATE TABLE warehouse (
      id serial UNIQUE,
      name text NOT NULL DEFAULT '',
      kolvo int NOT NULL DEFAULT 0,
      price decimal NOT NULL DEFAULT 0.0
);
-- \d+ warehouse

DROP TABLE IF EXISTS sales;
CREATE TABLE sales(
      id serial PRIMARY KEY,
      kolvo int NOT NULL,
      summa numeric NOT NULL DEFAULT 0.0,
      fk_warehouse int references warehouse(id) ON DELETE CASCADE,
      salesDate date default current_date
);

-- \d+ sales
INSERT INTO warehouse(name) VALUES ('apple');
--INSERT INTO warehouse(id,name) VALUES (1,'banana');
INSERT INTO warehouse(name) VALUES ('banana');
--INSERT INTO sales(fk_warehouse) VALUES (2);
--INSERT INTO sales(id, fk_warehouse) VALUES (1, 2);
INSERT INTO sales(fk_warehouse,kolvo,summa) VALUES (2,10,100);
INSERT INTO warehouse(name) VALUES ('apple2');
```

```
--INSERT INTO sales(fk_warehouse,kolvo,summa) VALUES (4,10,100);
COMMIT;


SELECT * FROM warehouse;
TABLE warehouse;
SELECT * FROM sales;


-- Физический уровень

-- Введение в линукс
-- https://aristov.tech/blog/likbez-po-linux/

-- Как посмотреть конфигурационные файлы?

show hba_file;
show config_file;
show data_directory;

\! nano /etc/postgresql/16/main/postgresql.conf


cd /var/lib/postgresql/16/main
ls -l

\! ls -l /var/lib/postgresql/16/main

psql

CREATE TABLE test5(i int);

-- всегда можем посмотреть, где лежит таблица
SELECT pg_relation_filepath('test5');

-- посмотрим на файловую систему
\! ls -l /var/lib/postgresql/16/main/base

\! ls -l /var/lib/postgresql/16/main/base/131080 | grep 131128

INSERT INTO test5 VALUES (1);

ls -l | grep

INSERT INTO test5 VALUES (2);

ls -l | grep


-- Табличные пространства
-- перейдем в домашний каталог пользователя postgres в ОС линукс
sudo su postgres
cd ~
pwd
```

```
mkdir temp_tblspce

psql

CREATE TABLESPACE ts location '/var/lib/postgresql/temp_tblspce';
\db
CREATE DATABASE app TABLESPACE ts;
\c app
\l+ -- посмотреть дефолтный tablespace
CREATE TABLE test (i int);
SELECT pg_relation_filepath('test');
CREATE TABLE test2 (i int) TABLESPACE pg_default;
SELECT tablename, tablespace FROM pg_tables WHERE schemaname = 'public';
ALTER TABLE test set TABLESPACE pg_default;
SELECT oid, spcname FROM pg_tablespace; -- oid унимальный номер, по
ктором у можем найти файлы
SELECT oid, datname,dattablespace FROM pg_database;

-- всегда можем посмотреть, где лежит таблица
SELECT pg_relation_filepath('test2');

-- Узнать размер, занимаемый базой данных и объектами в ней, можно с
помощью ряда функций.
SELECT pg_database_size('app');

-- Для упрощения восприятия можно вывести число в отформатированном виде:
SELECT pg_size_pretty(pg_database_size('app'));

-- Полный размер таблицы (вместе со всеми индексами):
SELECT pg_size_pretty(pg_total_relation_size('test2'));

-- И отдельно размер таблицы...
SELECT pg_size_pretty(pg_table_size('test2'));

-- ...и индексов:
SELECT pg_size_pretty(pg_indexes_size('test2'));


-- !!! с дефолтным неймспейсом не все так просто !!!
SELECT count(*) FROM pg_class WHERE reltablespace = 0;


-- Логический уровень
-- database
-- system catalog
-- schema
\dn

-- current schema
SELECT current_schema();

-- view table
\d pg_database
```

```
select * from pg_database;

-- seach path
SHOW search_path;
-- SET search_path to .. - в рамках сессии
-- параметр можно установить и на уровне отдельной базы данных:
-- ALTER DATABASE otus SET search_path = public, special;
-- в рамках кластера в файле postgresql.conf

\dt
-- интересное поведение и search_path
\d pg_database
CREATE TABLE pg_database(i int);

-- все равно видим pg_catalog.pg_database
\d pg_database

-- чтобы получить доступ к толко что созданной таблице используем
указание схемы
\d public.pg_database
SELECT * FROM pg_database limit 1;

-- в 1 схеме или разных?
CREATE TABLE t1(i int);
CREATE SCHEMA postgres;
CREATE TABLE t1(i int);

\dt
\dt+
\dt public.*
SET search_path TO public, "$user";
\dt

SET search_path TO public, "$user", pg_catalog;
\dt

CREATE TEMP TABLE t1(i int);
\dt

SET search_path TO public, "$user", pg_catalog, pg_temp;
\dt

-- можем переносить таблицу между схемами - при этом меняется только
запись в pg_class, физически данные на месте
ALTER TABLE t2 SET SCHEMA public;

-- чтобы не было вопросов указываем схему прямо при создании
CREATE TABLE public.t10(i int);


-- Пример склада и магазина в виртуальном окружении
https://www.db-fiddle.com/f/5RSrPHeFvaRbHaYJkxBZAR/0

-- Пример развертывания в YC
```

```
-- 01_ya_create_vm.txt
-- там же пример создания ssh ключей для подключения
```