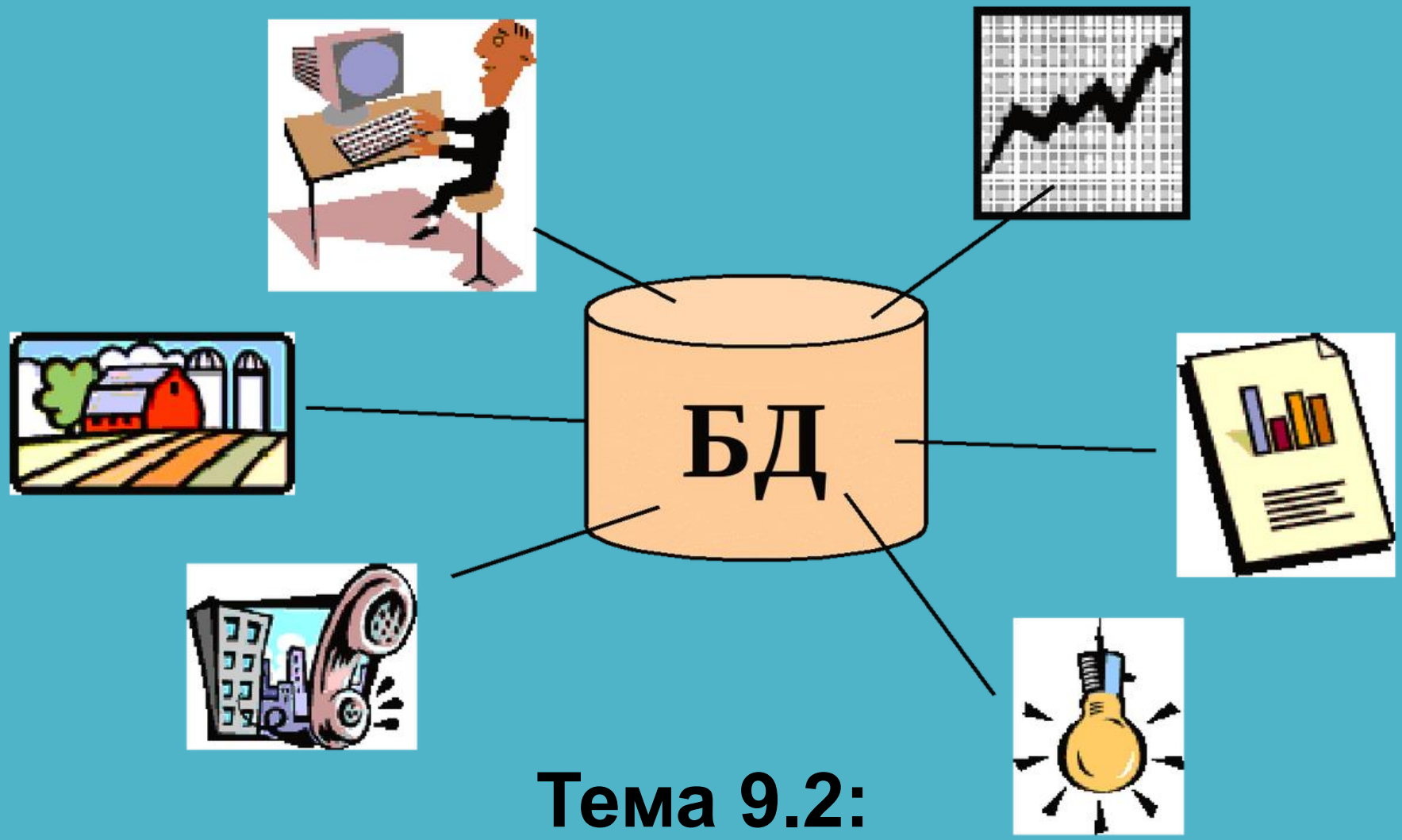


Тема 9.2:
Базы данных для разработчиков.

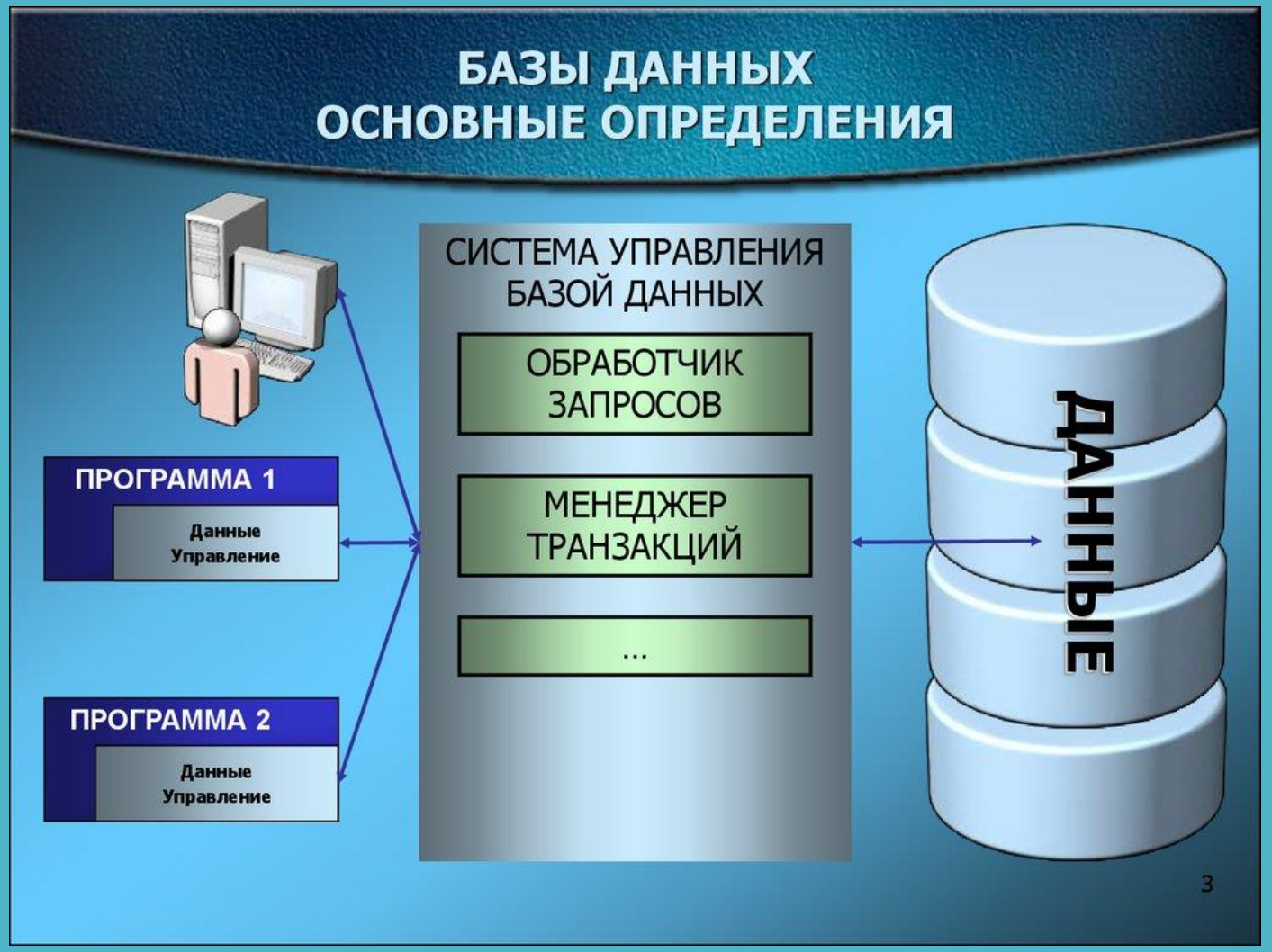
Цель занятия:

Ознакомиться с основными понятиями связанными с базами данных. Научиться их проектировать и использовать.

РАБОТА С POSTGRESQL, СОЗДАНИЕ БД

План занятия:

1. Работа через консоль
2. Создание ролей и БД (DCL)
3. DDL запросы



Повторение

Какие есть типы БД?

Реляционные – это БД, в которых информация строго структурирована и связана с другой информацией жесткими правилами.

Пример:

- Microsoft Access
- SQLite
- PostgreSQL
- MySQL
- Microsoft SQL

Нереляционные (NoSQL) – это БД, в которых нет жестких ограничений ни на структуру, ни на связь между информацией.

Пример:

- Redis
- MongoDB
- Cassandra

PostgreSQL

Установка

PostgreSQL

Работа через консоль

Работа через консоль

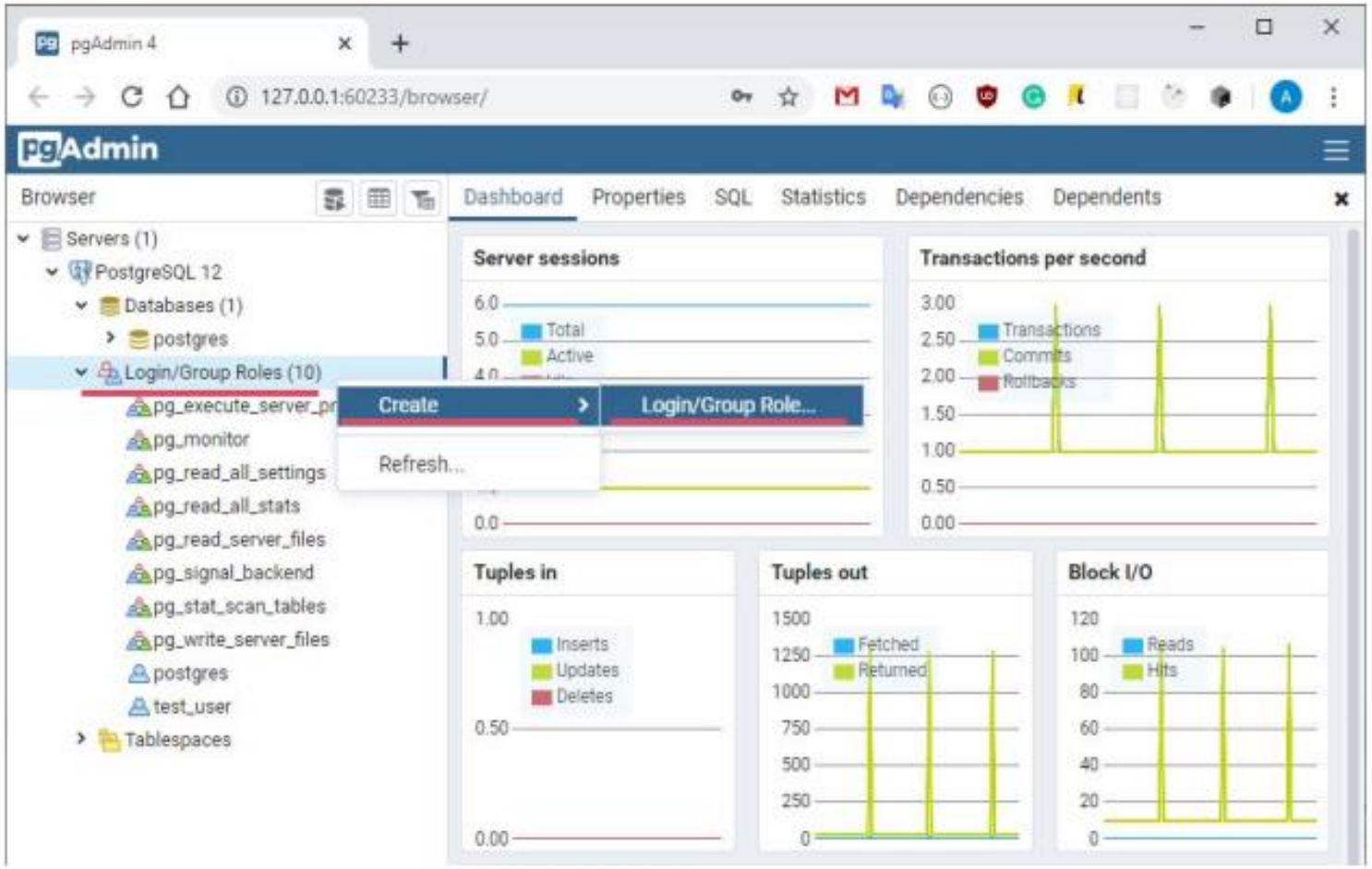
```
psql # запустить консольное приложение
     #для управления БД от текущего пользователя

psql -U <user>       # запустить консольное приложение для
                     # управления БД от пользователя <user>
                     # например, postgres

psql -d <database>   # запустить консольное приложение для
                     # управления конкретной БД - <database>

psql -U <user> -d <database>  # запустить консольное
                              # приложение для управления
                              # конкретной БД, <database>,
                              # от пользователя <user>

# Если БД не указана явно, то будет попытка подключиться к БД
# с таким же названием, как и имя пользователя

# Пользователь должен иметь права для управления БД
```

Создание ролей и БД. Консоль

```
psql -U postgres     # входим в режим управления от
                     # пользователя postgres (БД тоже postgres)

create database <name>;  # создаем БД с именем <name>
drop database <name>;    # удаляем БД с именем <name>

# создаем пользователя с именем <name> и паролем <pass>
create user <name> with password '<pass>';
drop user <name>;        # удаляем пользователя с именем <name>

# указываем, что владельцем БД <db_name>
# является пользователь <user_name>
alter database <db_name> owner to <user_name>;
```

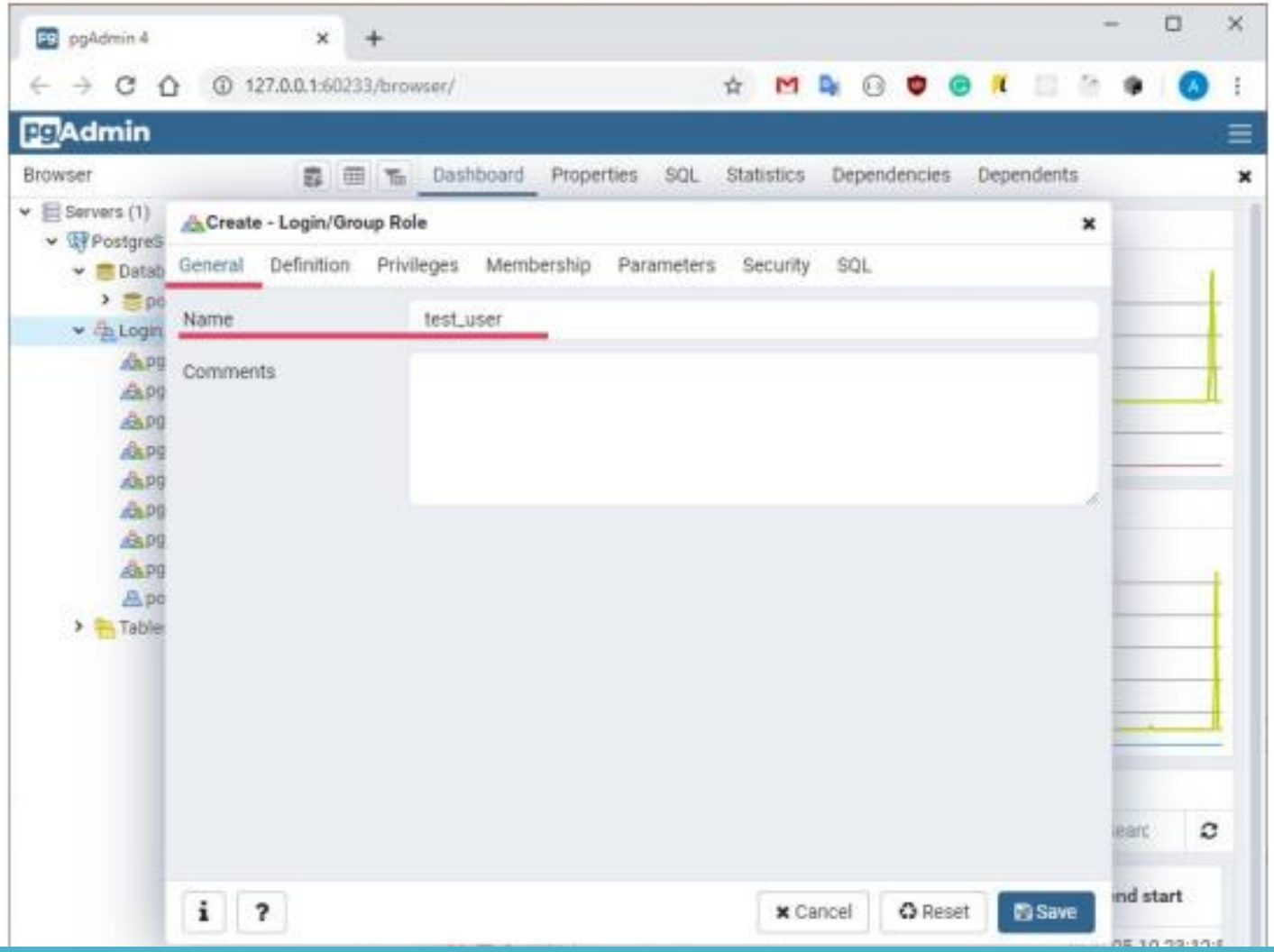
Создание ролей и БД. pgAdmin4

1. Правой клавишей мыши по разделу Login/Group Roles
2. Create
3. Login/Group Role



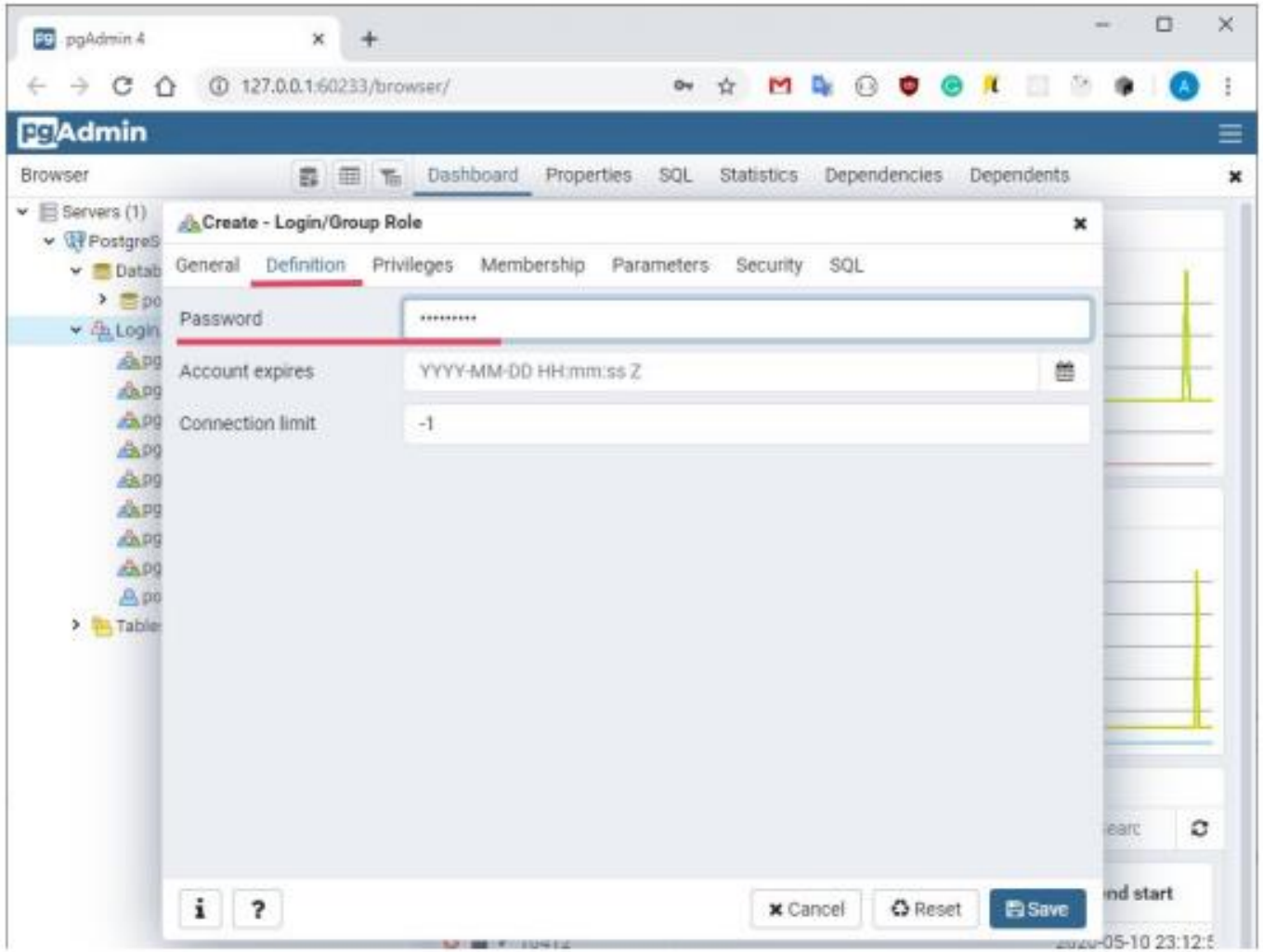
Создание ролей и БД. pgAdmin4

На вкладке **General** заполняем поле **Name** – это имя пользователя.



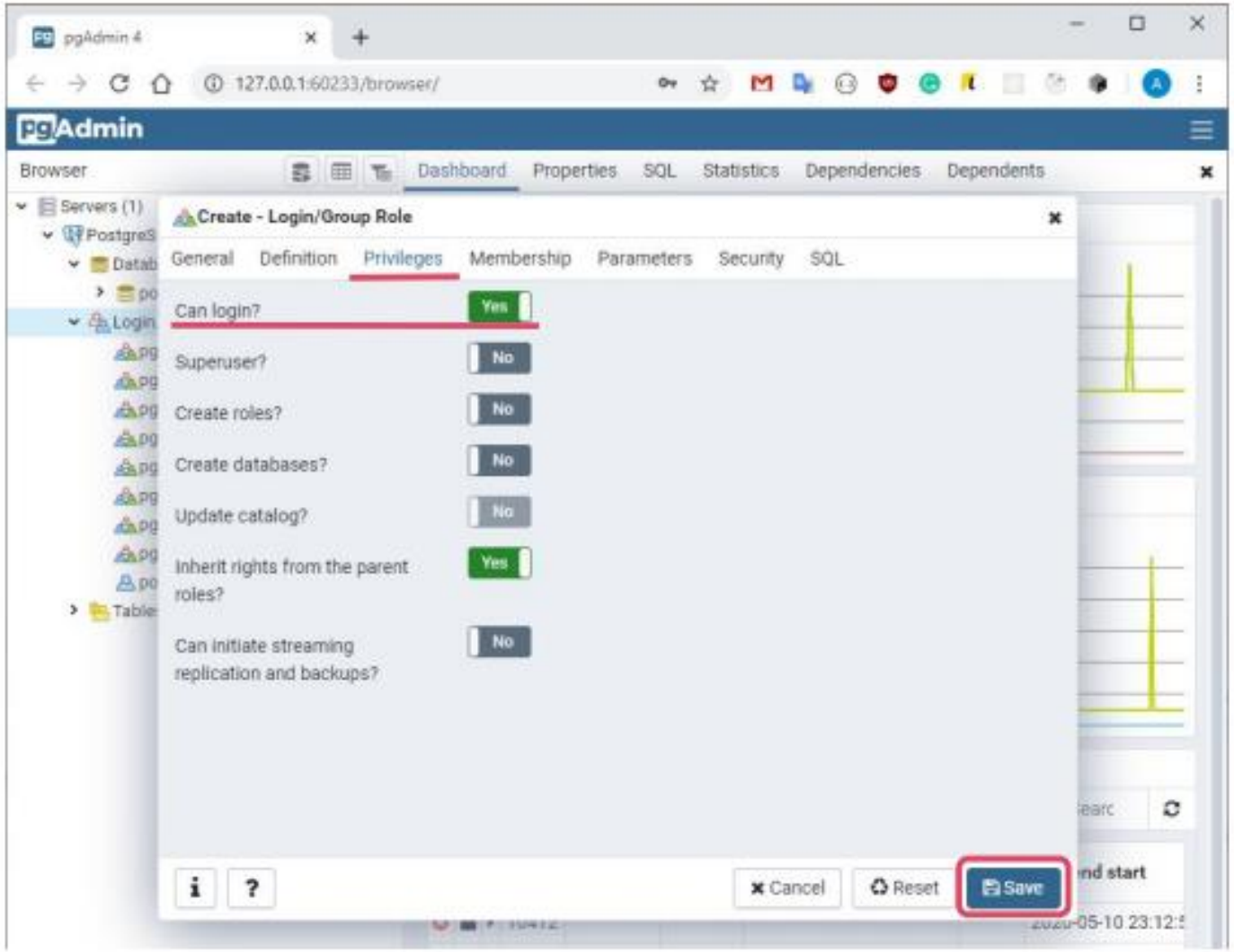
Создание ролей и БД. pgAdmin4

На вкладке **Definition** заполняем поле **Password** – это пароль для пользователя.



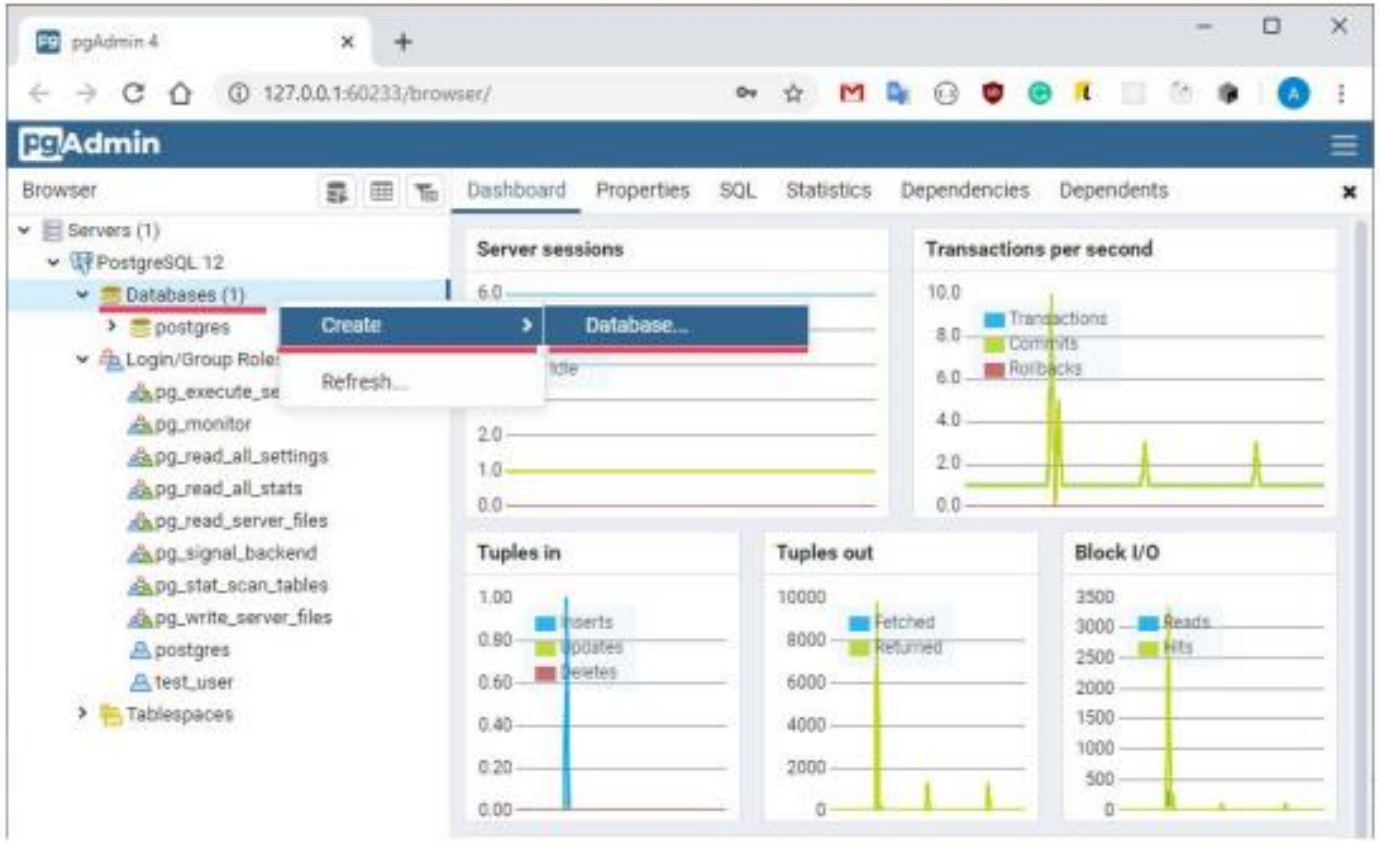
Создание ролей и БД. pgAdmin4

На вкладке **Privileges** отмечаем пункт **Can login?** и нажимаем **Save**.



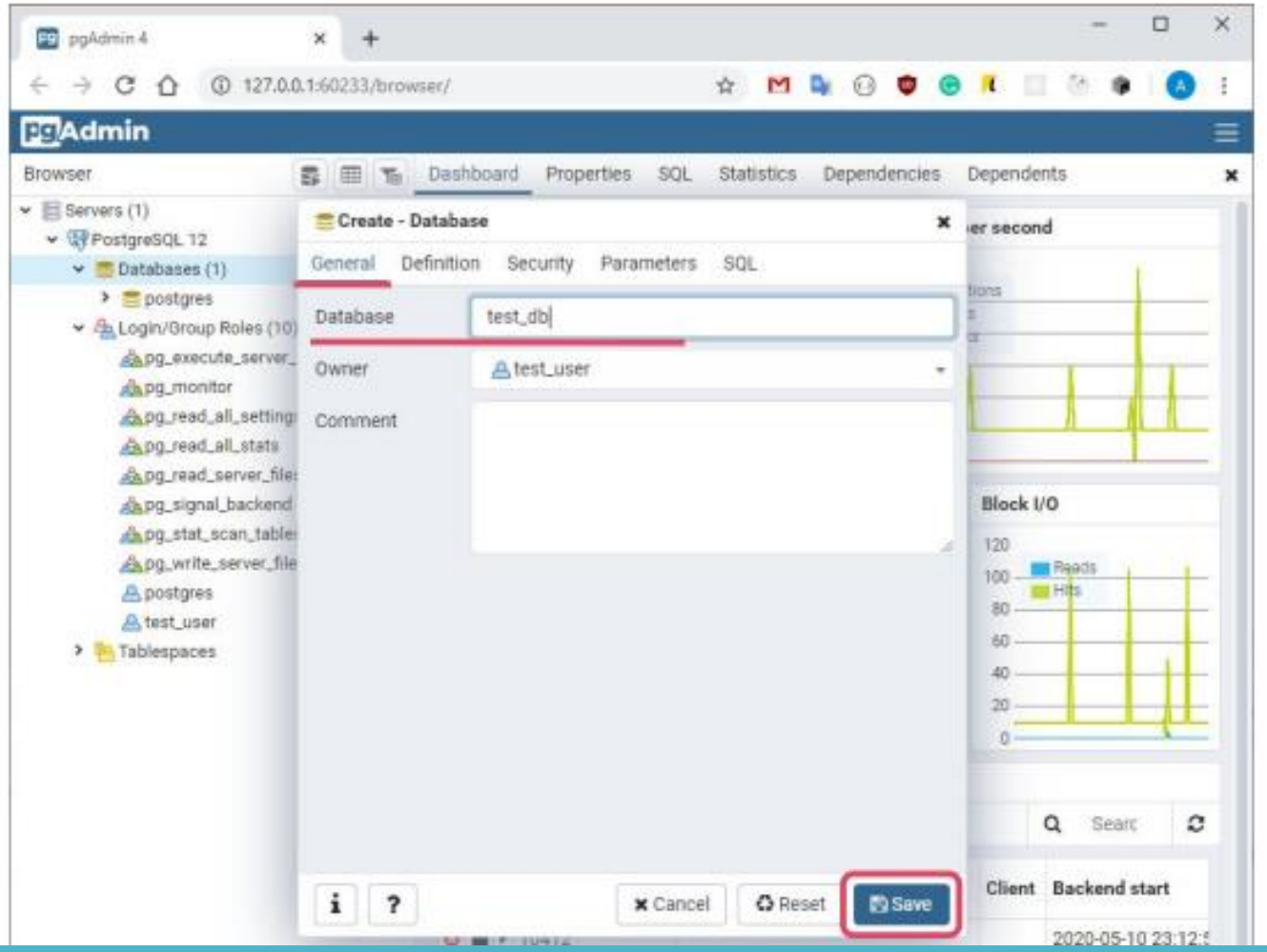
Создание ролей и БД. pgAdmin4

1. Правой клавишей мыши по разделу **Databases**
2. Create
3. Database



Создание ролей и БД. pgAdmin4

1. На вкладке **General** заполняем поле **Database** – это название БД.
2. В поле Owner пользователя, которого только что создали – это будет владелец БД.
3. Нажимаем кнопку **Save**.



Какие есть типы SQL запросов?

DDL (Data Definition Language). CREATE

Синтаксис:

```
create table [if not exists] <name> (
     <col_name_1> <col_type_1> [constraints],
     ...,
     <col_name_N> <col_type_N> [constraints],
     [constraints]
);
```

Пример:

```
create table if not exists Student (
     Id serial primary key,
     Name varchar(40) not null,
     GPA real,
     check(GPA > 0)
);
```

Создание таблицы: https://www.postgresql.org/docs/12/sql-createtable.html

DDL (Data Definition Language). Типы полей

Столбцы могут хранить только записи одного типа.
Основные типы:

Тип	Пояснение	Пример
integer	целые числа	id **integer**
serial	целые числа с автоинкрементом	id **serial**
numeric	десятичные числа	gpa **numeric**(3, 2)
character varying	строки ограниченной длины	name **varchar**(40)
text	строки произвольной длины	message **text**
date	дата (без времени)	birthday **date**
timestamp	дата + время	created_at **timestamp**
boolean	булевые значений	active **boolean**
jsonb*	JSON-поля	data **jsonb**

Типы полей: https://www.postgresql.org/docs/12/datatype.html

DDL (Data Definition Language). CONSTRAINTS

Ограничение	Описание	Пример
primary key	первичный ключ, обязывает поле быть уникальным и не пустым	`id serial primary key`
not null	значение не может быть пустым (не может отсутствовать)	`name varchar(40) not null`
unique	все значения в этом поле должны быть уникальным	`tag varchar(80) unique`
check	добавить проверку значения на описанное условие	`price numeric check(price > 0)`
foreign key	внешний ключ, обязывает значение соответствовать значению из другой таблицы	`product_id integer references products(id)`

Напоминание: ограничения можно описывать не только в конце описания атрибута, но и после описания всех атрибутов.

Ограничения: https://www.postgresql.org/docs/current/ddl-constraints.html

Практика

Создадим таблицы с прошлой лекции и установим связи между ними.

Напомню постановку задачи:

Есть категории интернет-магазина и есть товары. Каждый товар принадлежит строго одной категории. К товарам могут написать отзывы (к одному товару можно написать множество отзывов). Необходимо хранить информацию о категориях, товарах и отзывах.

DDL (Data Definition Language). ALTER и DROP

```
alter table <name> …
      # добавить атрибут
      add column <col_name> <col_type> [constraints];

      # переименовать таблицу
      rename to <new_table_name>;

      # переименовать атрибут
      rename <col_name> to <new_col_name>;

      # изменить тип атрибута
      alter column <col_name> set data type <col_type>;

      # добавить ограничение
      add constraint <constraint_name> <constraint>;

      # удалить ограничение
      drop constraint <constraint_name>;

      # удалить атрибут
      drop column <col_name>;



# удалить таблицу
drop table <name>;
```

Итоги

1. создали БД, пользователя и назначили владельца БД,
2. создали таблицы и связали их между собой,
3. изучили команды DDL.

INSERT INTO

Инструкция INSERT INTO используется для вставки новых записей в таблицу.

Если необходимо вставить значения только для части полей, то используется следующий синтаксис:

```
INSERT INTO rental(rental_date, inventory_id, customer_id, staff_id)
    VALUES(NOW(), 1, 3, 2);
```

Если необходимо вставить значения для всех полей, то нужно убедиться, что они все перечислены и их порядок соответствуют порядку полей:

```
INSERT INTO inventory
    VALUES(999, 999, 999);
```

Дополнительные материалы к занятию

Создание таблиц

Типы полей

Ограничения