Кафедра ЦТ
Институт информационных технологий
РТУ МИРЭА

Дисциплина «Проектирование баз данных»

PostgreSQL

PostgreSQL – это мощная объектно-реляционная СУБД с открытым исходным кодом.

Плюсы:

- Поддержка сложных запросов и транзакций
- Расширяемость при помощи сторонних модулей
- Высокая надежность и масштабируемость

Минусы:

- Сравнительно сложная настройка
- Требовательность к ресурсам.
- Относительно низкая скорость работы в некоторых простых операциях

DBeaver

DBeaver – это универсальный инструмент для работы с базами данных, который поддерживает множество СУБД (например, MySQL, PostgreSQL, Oracle, SQLite, MongoDB и другие). Он предоставляет удобный графический интерфейс для выполнения SQL-запросов, управления структурами баз данных, визуализации данных и администрирования.

К основным функциям можно отнести:

- Подключение к различным базам данных.
- Выполнение и редактирование SQL-запросов.
- Визуализация и экспорт данных.
- Управление структурами таблиц и схем.
- Поддержка плагинов и расширений.

SQL

Для работы с СУБД используется специальный язык SQL

SQL (Structured Query Language) – это язык структурированных запросов, используемый для взаимодействия с реляционными базами данных.

Стоит отметить, что в различных СУБД могут использоваться разные вариации (диалекты) языка SQL, однако большинство запросов будет иметь одинаковый или очень похожий синтаксис.

Далее будут рассмотрены запросы, применимые в СУБД PostgreSQL

SQL. Создание таблиц

Для создания таблиц используется оператор **CREATE TABLE**, определяющий структуру таблицы, имена и типы столбцов.

Шаблон команды с оператором `CREATE TABLE` выглядит следующим образом:

```
CREATE TABLE table_name (
  column1 datatype1 constraints,
  column2 datatype2 constraints, …
);
```

`table_name` – имя таблицы

`column1, column2` – имена столбцов

`datatype1, datatype2` – типы данных столбцов

`constraints` – ограничения (`PRIMARY KEY, NOT NULL, FOREIGN KEY, …`)

SQL. Создание таблиц

Пример создания таблицы отзывов:

```
CREATE TABLE Review (
  ID_Review bigserial NOT NULL PRIMARY KEY,
  ID_Client bigint NOT NULL,
  ID_Medicine bigint NOT NULL,
  Contents text,
  Rating smallint NOT NULL,
  FOREIGN KEY (ID_Client) REFERENCES Client (ID_Client),
  FOREIGN KEY (ID_Medicine) REFERENCES Medicine (ID_Medicine)
);
```

Обратите внимание на синтаксис создания внешнего ключа: указывается какой столбец является внешним ключом, на какую таблицу он ссылается и на какой столбец.

SQL. Заполнение таблиц

Для заполнения таблиц используется оператор **INSERT INTO**, позволяющий произвести вставку строки в таблицу, указывая значения для каждого столбца

Шаблон команды с оператором `INSERT INTO` выглядит следующим образом:

```
INSERT INTO table_name (column1, column2, …)
VALUES (value1, value2, …);
```

`table_name` – имя таблицы

`column1, column2` – имена столбцов

`value1, value2` – значения, заполняемые в столбцах

Далее рассмотрим пример вставки строки в таблицу `Client` со столбцами

`ID_Client: bigserial, Surname: varchar(100), Name: varchar(100),`

`Fathers_name: varchar(100), Phone: varchar(20)`

SQL. Заполнение таблиц

Пример вставки одной строки в таблицу:

```
INSERT INTO Client (Surname, Name, Fathers_name, Phone)
VALUES ('Иванов', 'Иван', 'Иванович', '+79899988989');
```

Обратите внимание, что столбец `ID_Client` не указан в команде, так как он имеет тип `bigserial`, который самостоятельно подставляет автоматически увеличивающиеся значения

Если бы в таблице были столбцы без ограничения `NOT NULL`, то этот столбец можно было бы не указывать, в таком случае вместо него подставилось бы значение по умолчанию для типа этого столбца

SQL. Просмотр записей в таблице

Для просмотра таблиц используется оператор **SELECT**, позволяющий произвести выборку данных из таблицы по какому-либо условию.

Шаблон команды с оператором `SELECT` выглядит следующим образом:

```
SELECT column1, column2, …
FROM table_name
WHERE condition;
```

`table_name` – имя таблицы из которой производится выборка

`column1, column2` – имена столбцов, которые должны присутствовать в выборке

`condition` – условие, которое определяет, какие строки будут выбраны. Это необязательное поле. Если оно не указано, будут выбраны все строки из таблицы.

Если необходимо вывести значения всех столбцов, то можно указать символ `*` вместо списка имен столбцов для выборки

SQL. Просмотр записей в таблице

Пример выборки всех записей из таблицы клиентов:

```
SELECT *
FROM Client;
```

Пример выборки номеров телефонов всех клиентов по имени Иван:

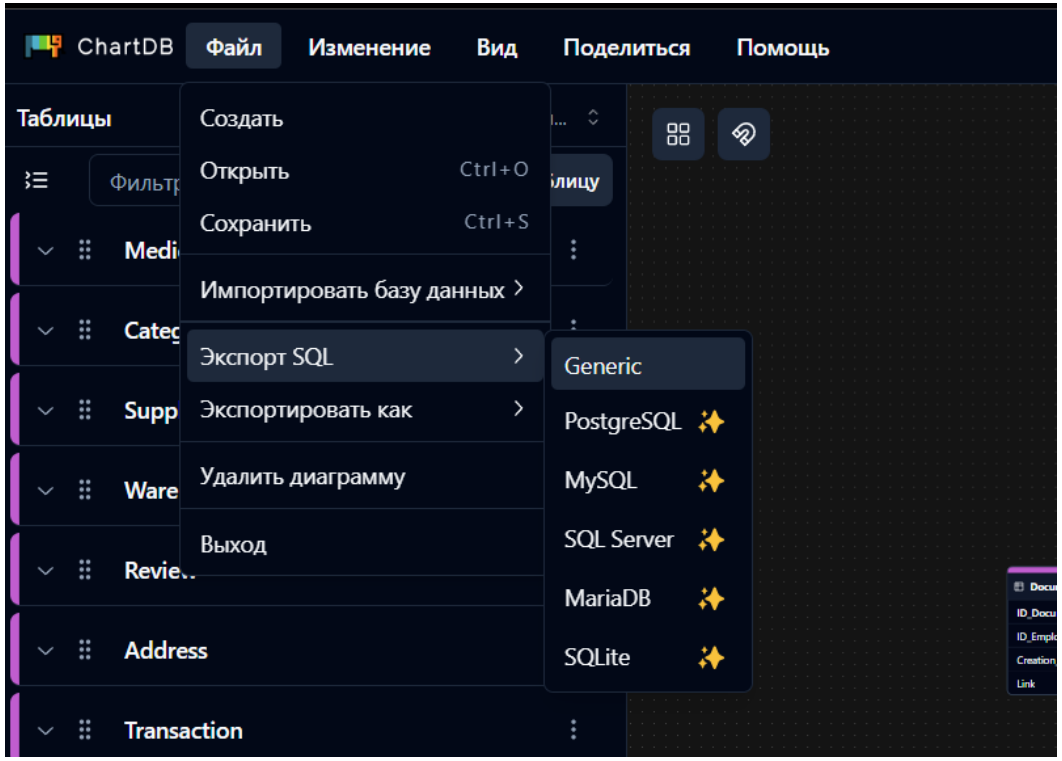
```
SELECT Phone
FROM Client
WHERE Name = ‘Иван’;
```

Пример выборки всех клиентов с фамилией Иванов и именем Пётр:

```
SELECT *
FROM Client
WHERE Surname = ‘Иванов’ AND Name = ‘Пётр’;
```

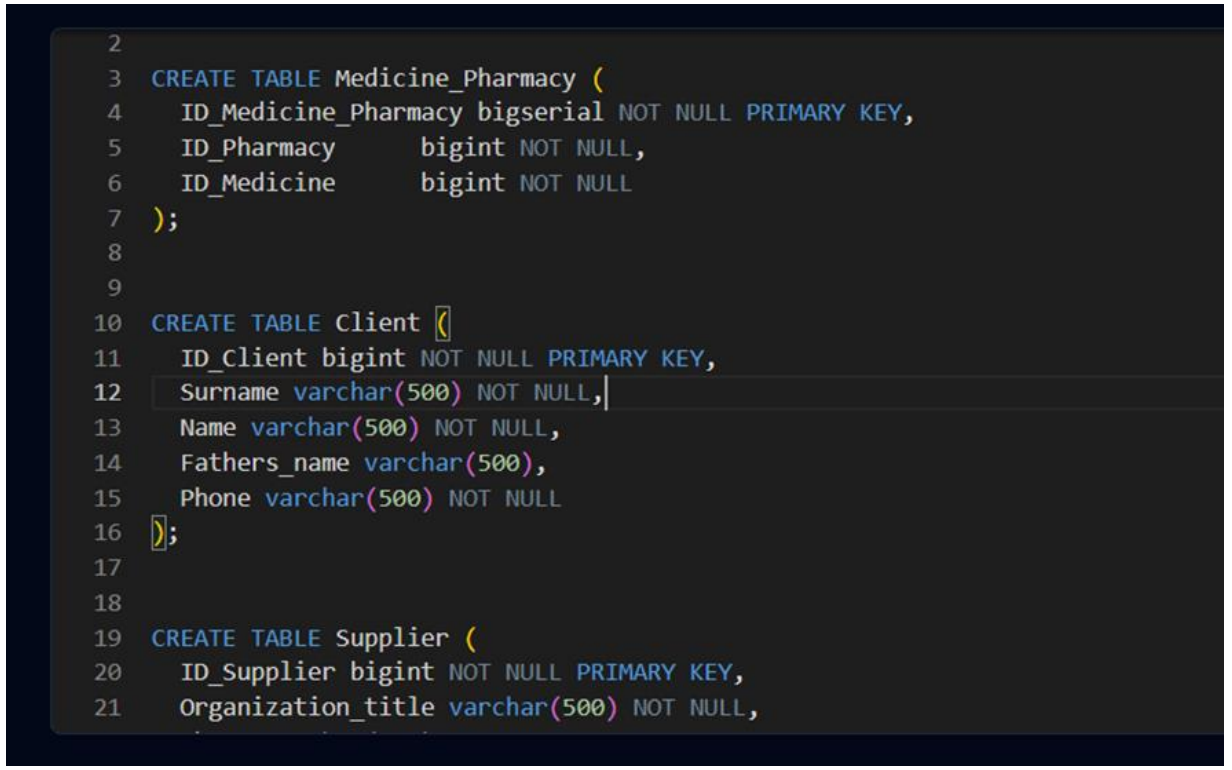
Выгрузка SQL из ChartDB

В ChartDB есть возможность сгенерировать код для создания таблиц на основе построенной схемы. Для этого необходимо перейти в пункт меню «Файл» -> «Экспорт SQL» -> «Generic».



Выгрузка SQL из ChartDB

После этого ChartDB сгенерирует SQL код, который можно использовать в Dbeaver для автоматического создания таблиц. Обратите внимание, что в сгенерированном коде у всех столбцов типа `varchar` стоит длина по умолчанию 500, её стоит изменить.



```sql

CREATE TABLE Medicine_Pharmacy (
  ID_Medicine_Pharmacy bigserial NOT NULL PRIMARY KEY,
  ID_Pharmacy      bigint NOT NULL,
  ID_Medicine      bigint NOT NULL
);


CREATE TABLE Client (
  ID_Client bigint NOT NULL PRIMARY KEY,
  Surname varchar(500) NOT NULL,
  Name varchar(500) NOT NULL,
  Fathers_name varchar(500),
  Phone varchar(500) NOT NULL
);


CREATE TABLE Supplier (
  ID_Supplier bigint NOT NULL PRIMARY KEY,
  Organization_title varchar(500) NOT NULL,
```

Практическая работа №7. Основы SQL в PostgreSQL

Постановка задачи: на основе спроектированной физической схемы данных создать и заполнить базу данных при помощи DBeaver.

Решение:

- При помощи ChartDB сгенерировать и отредактировать запросы на создание таблиц
- Создать таблицы в PostgreSQL при помощи DBeaver
- Заполнить созданные таблицы
- Вывести наполнение всех созданных таблиц

Результат выполнения практической работы