МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**по дисциплине  
  «Технологии проектирования ПО»**

Выполнил студент группы 35/2                                                              Д.А.Вербицкий

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    3

Краснодар

2025 г.

**Таблица 1. Классификация сущностей**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Сущность | Атрибуты | Описание |
| Справочные | | | |
| 1 | Товар | id, название, цена, количество | Информация о товаре и его количестве на складе |
| 2 | Заказ | id, полная\_цена, дата, дата\_записи | Информация о заказе |
| Оперативные | | | |
| 1 | Заказы\_Товары | id, id\_заказа, id\_товара, цена, количество, дата | Таблица-связка для информации о наполнении заказов |

**Таблицы**

|  |  |  |
| --- | --- | --- |
| № | Имя таблицы | Описание |
| 1 | Заказы | Информация о заказах |
| 2 | Товары | Информация о товарах |
| 3 | Чек\_товары | Таблица-связка для информации о содержимом заказов |

**Почему для проекта была выбрана PostgreSQL, а не MySQL**

При выборе СУБД для проекта было рассмотрено несколько вариантов, включая MySQL и PostgreSQL. В итоге предпочтение было отдано PostgreSQL по следующим причинам:

1. **Расширенные возможности и богатый функционал**

PostgreSQL предлагает более продвинутые функции, такие как:

* Поддержка сложных типов данных (JSON/JSONB, массивы, hstore, геоданные через PostGIS).
* Полноценные **хранимые процедуры** на нескольких языках (PL/pgSQL, Python, JavaScript и др.).
* Расширенные индексы (GIN, GiST, BRIN), что ускоряет поиск по нетривиальным данным.
* Поддержка **рекурсивных запросов** и **оконных функций** для аналитики.

2. **Гибкость и соответствие стандартам SQL**

PostgreSQL строже следует стандарту SQL, что делает его более предсказуемым и удобным для сложных запросов. В отличие от MySQL, который иногда допускает нестрогое поведение (например, неявные преобразования типов).

3. **Масштабируемость и параллельные запросы**

PostgreSQL лучше справляется с высокой нагрузкой и сложными аналитическими запросами благодаря:

* Эффективному планировщику запросов.
* Поддержке **многопоточной обработки** (параллельные запросы, индексы, агрегация).
* Возможностям **шардинга** и репликации (через расширения, например, Citus).

4. **Надежность и ACID-совместимость**

PostgreSQL гарантирует целостность данных даже в сложных сценариях:

* Полная поддержка транзакций (включая DDL-транзакции).
* Механизм **MVCC (Multiversion Concurrency Control)** без блокировок на чтение.
* Жесткая проверка типов и ограничений.

5. **Открытость и экосистема**

PostgreSQL — полностью открытая СУБД с активным сообществом. В отличие от MySQL (который принадлежит Oracle), PostgreSQL развивается без коммерческих ограничений, предлагая:

* Большое количество расширений (от Full-Text Search до TimescaleDB для временных рядов).
* Поддержку **частичных индексов**, **составных типов**, **наследования таблиц**.

6. **Безопасность**

PostgreSQL предоставляет более тонкие механизмы управления доступом:

* Ролевая модель с наследованием прав.
* Поддержка **SSL**, **SE-PostgreSQL** (интеграция с SELinux).
* Возможность шифрования данных на уровне таблиц.

Хотя MySQL остается популярным выбором для простых веб-приложений, он уступает PostgreSQL в:

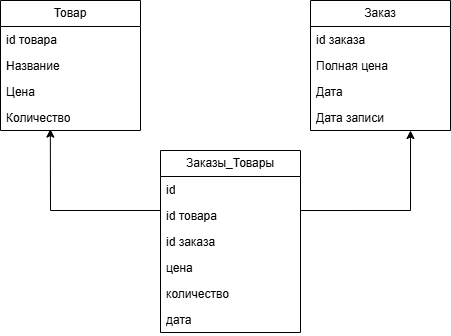
* **Ограниченной поддержке сложных запросов** (например, рекурсивные CTE появились только в 8.0).
* **Менее гибкой системе типов** (нет встроенной работы с JSON до 5.7, ограниченные массивы).
* **Проблемах с параллелизмом** (InnoDB может быть узким местом при высокой нагрузке на запись).

**Вывод**

PostgreSQL был выбран как более надежная, гибкая и производительная СУБД, особенно для проектов с требованием к сложной логике данных, аналитике и долгосрочной масштабируемости.

Если у проекта появятся специфические требования (например, сверхвысокая скорость простых запросов или интеграция с legacy-системами), можно будет рассмотреть альтернативы, но на текущем этапе PostgreSQL — оптимальный выбор.

**Диаграмма классов UML**



**SQL-код**

CREATE TABLE Products (

product\_id SERIAL PRIMARY KEY,

title VARCHAR(100) NOT NULL,

quantity INTEGER NOT NULL CHECK (quantity >= 0)

);

CREATE TABLE Orders (

order\_id SERIAL PRIMARY KEY,

total\_price DECIMAL(10, 2) NOT NULL CHECK (total\_price >= 0),

date DATE NOT NULL,

date\_of\_recording TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Orders\_Products (

id SERIAL PRIMARY KEY,

order\_id INTEGER NOT NULL REFERENCES Orders(order\_id) ON DELETE CASCADE,

product\_id INTEGER NOT NULL REFERENCES Products(product\_id) ON DELETE RESTRICT,

price DECIMAL(10, 2) NOT NULL CHECK (price >= 0),

quantity INTEGER NOT NULL CHECK (quantity > 0),

UNIQUE(order\_id, product\_id)

);