Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ “Информатика и системы управления”

КАФЕДРА "Программное обеспечение ЭВМ и информационные технологии”

КУРСОВАЯ РАБОТА

НА ТЕМУ:

“Проектирование базы данных и создание SQL-запросов к ней”

Студент ИУ7-54Б(В) _____ М. А. Сергеев
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ А. А. Павлюк
(Подпись, дата) (И.О.Фамилия)

Москва
2022 г.

Оглавление

Введение

Целью курсовой работы является проектирование базы данных и реализация запросов к созданной БД с использованием языка SQL (Structured Query Language - язык структурированных запросов).

Выбор реляционной модели данных не является случайным – несмотря на то, что она не являются исторически первой моделью хранения данных, благодаря своим преимуществам, в числе которых более простой доступ к оперативно составляемым отчетам (обычно через SQL) и обеспечение повышенной надежности и целостности данных благодаря отсутствию избыточной информации, системы управления именно реляционных баз данных статистически являются наиболее популярными (по данным портала https://pypl.github.io/DB.html).

Реляционная модель данных (РМД) основана на реляционной алгебре, представляющей собой замкнутую систему операций (как и в любой другой алгебре) над отношениями.

Реляционная база создается и затем управляется с помощью реляционной системы управления базами данных. Фактически реляционная база данных — это тело связанной информации, сохраняемой в двухмерных таблицах. Связь между таблицами может находить свое отражение в структуре данных, а может только подразумеваться, то есть присутствовать на неформализованном уровне. Каждая таблица БД представляется как совокупность строк и столбцов, где строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы - атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.

Отношения в рамках реляционной модели могут иметь простую графическую интерпретацию в виде двумерной таблицы. С учётом реляционной алгебры можно производить соответствующие операции над такими таблицами. Такие операции зачастую лежат в основе запросов, позволяющих на практике получить необходимую информацию из базы данных[1].

Принципы реляционной модели были сформулированы в 1969—1970 годах Э. Ф. Коддом. Несмотря на то, что конкретные программные реализации данной модели были созданы с определёнными отклонениями от неё, существующие реляционные СУБД, такие, как, например, MySQL, SQLite или PostgreSQL (используемая в настоящем курсовом проекте) считаются удобными средствами, используемыми во множестве программных продуктов.

[1] Национальная библиотека им. Н.Э. Баумана. URL: https://ru.bmstu.wiki/Реляционная_база_данных

Распространённость реляционных систем управления базами данных в различных сферах индустрии программной инженерии лишь подчёркивает актуальность выбранной темы.

1. Аналитический раздел

1.1. Постановка задачи

Для реализации запросов с помощью РСУБД PostgreSQL была инициализирована БД, наполненная тестовыми данными.

Перед началом выполнения таких запросов важно осознавать, какие возможности для пользователя СУБД они предоставляют. Функционально их можно разделить на следующие виды:

- запросы, предназначенные для работы со структурой данных - для создания, описания и модификации БД;

- запросы, используемые непосредственно в работе с данными, с помощью которых можно добавлять, обновлять, сохранять и удалять данные;

- запросы, применяемые для предоставления или отмены прав доступа к БД;

В свою очередь, каждый из видов SQL-запросов подразделяется на типы:

- команды, работающие со структурой БД. К ним относятся CREATE - «создать» (например, CREATE TABLE (создать таблицу), CREATE USER (создать пользователя)), ALTER - «модифицировать» (этот запрос используется при внесении изменений в саму БД или в ее часть), DROP - «удалить» (также относятся к БД и ее частям);

- команды, работающие с данными. К наиболее востребованным запросам относятся: SELECT (выборка данных), INSERT (вставка новых данных), UPDATE (обновление данных), DELETE (удаление данных), MERGE (слияние данных);
- команды, работающие с правами доступа. В их список входят GRANT - разрешение пользователю на проведение определенных операций с БД или данными; REVOKE – отзыв выданного разрешения; DENY – установка запрета, имеющего приоритет над разрешением.

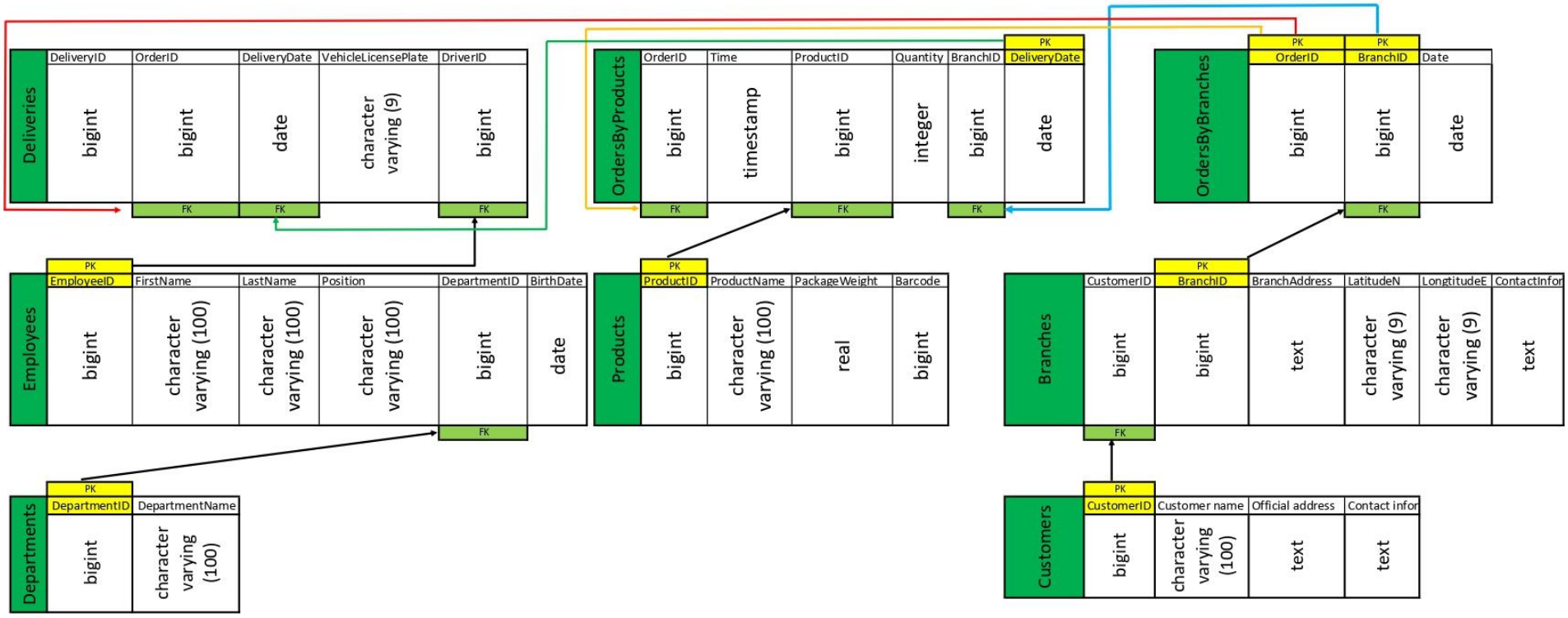
При составлении SQL-запроса для работы с базами данных в СУБД (MySQL, Microsoft SQL Server, PostgreSQL) вводятся следующие параметры отбора:

- названия таблиц, из которых необходимо извлечь данные;
- поля, значения которых требуется вернуть к исходным после внесения изменений в БД;
- связи между таблицами;
- условия выборки;
- вспомогательные критерии отбора (ограничения, способы представления информации, тип сортировки)[2].

2 Виды и типы SQL запросов. URL: https://hsbi.hse.ru/articles/vidy-i-tipy-sql-zaprosov/

2. Конструкторский раздел

2.1. Проектирование БД



На диаграмме выше изображены отношения, составляющие базу данных, а также связи между ними. Так, желтым цветом отмечены первичные ключи, а светло-зелёным – соответствующие внешние ключи. Такие связи обеспечивают необходимые свойства базы данных.
Кроме того, на диаграмме отмечены используемые для хранения типы данных (в соответствии с нотацией PostgreSQL).
Настоящая база данных содержит в себе информацию об отделах, входящих в состав предприятия (Departments), о сотрудниках предприятия (Employees), о клиентах (Customers) и их филиалах с точным указанием их местонахождения (Branches) .
Присутствует информация о производимых предприятием продукцией (Products), фактах заказа продукции соответствующими филиалами (OrdersByClients) и детализированной расшифровкой заказа (OrdersByProducts).
Также в базе присутствует информация касательно доставки того или иного заказа (Deliveries).
Спроектированная таким образом база обеспечивает соблюдение необходимых требований к ней при её проектировании.
Для имитации работы с реальными данными были использованы вымышленные данные, в том числе сгенерированные с помощью сторонних ресурсов (к примеру, Рандомус - https://randomus.ru/name), а также были использованы реальные данные, полученные из доступных источников (в частности, геоинформационных систем).

2.2. Создание базы данных

Скрипты создания и наполнения базы данных в формате .sql находятся в приложении к настоящей работе.

3. Технологический раздел

3.1. SQL-запросы

В настоящей работе реализовано 6 (шесть) сложных запросов, все они приведены ниже, а также находятся в приложении к работе в формате .sql

1) Вывод списка всех сотрудников предприятия с указанием названий отделов.

```
SELECT
"FirstName", "LastName", "DepartmentName"
FROM
"Employees" JOIN "Departments"
ON
"Departments"."DepartmentID" = "Employees"."DepartmentID"
ORDER BY
"FirstName";
```

2) Вывод количества сотрудников, работающих в каждом отделе.

```
SELECT
"DepartmentName",
COUNT (*) AS "NumberOfEmployees"
FROM
"Departments" JOIN "Employees"
ON
"Departments"."DepartmentID" = "Employees"."DepartmentID"
GROUP BY
"DepartmentName";
```

3) Количество проданной продукции по каждой позиции за всё время.

```
SELECT
"ProductName", SUM("Quantity")
FROM
"Products" JOIN "OrdersByProducts"
ON
"Products"."ProductID" = "OrdersByProducts"."ProductID"
GROUP BY
"ProductName";
```

4) Количество проданной продукции по каждой позиции по дням.

```
SELECT
"ProductName", SUM("Quantity"), "DeliveryDate"
FROM
"Products" JOIN "OrdersByProducts"
ON
"Products"."ProductID" = "OrdersByProducts"."ProductID"
GROUP BY
"DeliveryDate", "ProductName";
```

5) Вывод массы продукции, предназначенной для загрузки (для каждого автомобиля доставки) с указанием даты доставки и государственного регистрационного номера транспортного средства.

```
SELECT
"DeliveryID", SUM("ProductWeight") AS "TotalTruckWeight", "DeliveryDate",
"VehicleLicensePlate"
FROM
(SELECT
"OrdersByProducts"."OrderID", "Products"."ProductName", "PackageWeight",
"Quantity", "PackageWeight" * "Quantity" AS "ProductWeight"
FROM
"OrdersByProducts" JOIN "Products" ON "OrdersByProducts"."ProductID" =
"Products"."ProductID"
ORDER BY
"OrderID")
AS "TempTable"
JOIN
"Deliveries"
ON
"Deliveries"."OrderID" = "TempTable"."OrderID"
GROUP BY
"DeliveryID", "DeliveryDate", "VehicleLicensePlate"
ORDER BY
"DeliveryID";
```

6) Вывод координат торговых точек по каждой доставке.

```
SELECT
"DeliveryID", "Deliveries"."OrderID", "TempTable"."LatitudeN",
"TempTable"."LongtitudeE", "DeliveryDate"
FROM
(SELECT
"OrderID", "LatitudeN", "LongtitudeE"
FROM
"OrdersByBranches" JOIN "Branches" ON "OrdersByBranches"."BranchID" =
"Branches"."BranchID") AS "TempTable" JOIN "Deliveries"
ON "TempTable"."OrderID" = "Deliveries"."OrderID"
GROUP BY
"DeliveryID", "Deliveries"."OrderID", "TempTable"."LatitudeN",
"TempTable"."LongtitudeE", "DeliveryDate"
ORDER BY
"DeliveryID", "OrderID";
```

Заключение

В результате проведённой работы была достигнуты следующие цели:

- Инициализирована база данных, наполненная данными.

- Реализованы сложные запросы, позволяющие, как, например, непосредственно получить информацию оператору базы данных, так и использовать полученные данные, применяя сторонние программные продукты

Список использованных источников информации

1. К. Дж. Дейт. Введение в системы баз данных. 2018

2. Документация к PostgreSQL 14.5. [Электронный ресурс].

URL: https://postgrespro.ru/docs/postgresql/14/index

3. Рандомус. [Электронный ресурс].
URL: https://randomus.ru

4. Павлюк А.А. Лекционные материалы по дисциплине «Базы данных».