МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий
Базовая кафедра № 231 – информационных процессов и систем

КУРСОВАЯ РАБОТА

Проектирование модели данных системы управления логистикой

по дисциплине

«ФОРМАЛИЗОВАННЫЕ МОДЕЛИ ДАННЫХ»

Выполнил студент группы *ИКБО-11-23* *Мусатов И.А.*

Принял

_____ *Силаев Ю.В.*

Практическая работа выполнена «__» _____ 2024 г. _____

«Зачтено» «__» _____ 2024 г. _____

Москва 2024

РЕФЕРАТ

Отчет 36 стр., 40 рисунков, 2 таблицы, 8 источников.

Ключевые слова: логистика, транспортная логистика, математическая оптимизация, базы данных, реляционная модель данных.

Объектом исследования является модель данных управления транспортной логистикой.

Цель работы – создание модели данных для оптимизации маршрутов доставки, учета транспортных средств и грузов.

Методы исследования и реализации включают в себя теоретический анализ области транспортной логистики, анализ математической основы оптимизации, изучение реляционных СУБД на примере PostgreSQL, формирование запросов на языке SQL.

Результатом работы является модель данных системы управления транспортной логистикой, которая включает в себя аспект структуры в виде системы сущностей (таблиц) и связей между ними, аспект манипуляции в виде набора оптимизационных SQL-запросов, аспект целостности в виде ограничений на данные в таблицах.

Область применения результатов включает в себя торговые предприятия, транспортные компании и, в общем плане, всякую систему, использующую в процессе своего функционирования модель данных транспортной логистики.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

TSP	–	Traveler Salesman Problem (задача о коммивояжере/ задача китайского почтальона)
DEF STAN	–	Defence Standard of Ministry of Defence, GB (Оборонный Стандарт Министерства Обороны Великобритании)
GIS	–	Geospatial Information Systems (Геоинформационные системы)
SQL	–	Structured Query Language (Структурированный язык запросов)
ER-диаграмма	–	Диаграмма entity-relationship (сущность-связь)
Сущность	–	(англ. entity) конкретный или абстрактный объект в рассматриваемой области. Описывается набором атрибутов.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ERD	–	Entity-relationship diagram
ПО	–	Программное обеспечение
БД	–	База данных
СУБД	–	Система управления базами данных
ООП	–	Объектно-ориентированная парадигма
ЛА	–	Логистический анализ
ИЛП	–	Интегрированная логистическая поддержка

ВВЕДЕНИЕ

Еще со времен формирования первых человеческих цивилизаций возникла проблема оптимального распределения ресурсов. Глобальные стройки, войны, торговля – все эти естественные для общества процессы требуют организации сложного учета колоссального количества ресурсов, материалов и товаров.

Для формального описания механизмов подобных процессов, а также исследования оптимальных стратегий возникла научная дисциплина – логистика. В эпоху глобализма и развитой экономики эта область знания находит особенное применение. Оптимизация издержек соответствует интересам как небольших торговых предприятий, так и целых государственных министерств. В этом видим актуальность данной работы.

Данная курсовая работа направлена на исследование предметной области транспортной логистики. Особое внимание уделяется рассмотрению математических методов оптимизации в некоторых задачах логистики.

Целью данной работы является создание модели данных для оптимизации маршрутов доставки, учета транспортных средств и грузов.

В соответствии с целью автором работы ставятся следующие задачи:

- Изучить предметную область транспортной логистики;
- Изучить математические методы оптимизации для решения задач логистики;
- Спроектировать структуру базы данных для модели данных системы управления логистикой;
- Реализовать SQL-запросы для оптимизационной обработки данных, тем самым формируя аспект манипуляции модели данных.

1 ИЗУЧЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Общее понятие логистики

Логистика – интегральный инструмент менеджмента, способствующий достижению стратегических, тактических или оперативных целей организации бизнеса за счет эффективного (с точки зрения снижения общих затрат и удовлетворения требований конечных потребителей к качеству продуктов и услуг) управления материальными и (или) сервисными, а также сопутствующими им потоками (финансовыми, информационными и т.п.). [1]

Основным понятием в логистике являются материальные потоки. То есть, всякие потоки материалов, продукции, начиная от поставщиков через предприятия снабжения и распределения, и заканчивая доставкой конечным потребителям. Для решения экономических задач логистики вводят понятие логистической системы.

Логистическая система — это система управления материальными, информационными и финансовыми потоками на основе оптимизации процессов движения в целях минимизации затрат. [2]

Оптимизация процессов подразумевает создание моделей данных для решения конкретных задач. Приведем основные задачи логистики:

- Планирование, менеджмент и реализация продукции (производственные структуры);
- Снабжение, складирование материалов (складские структуры);
- Транспортировка и распределение продукции (транспортные структуры).

В данной работе нас будет интересовать транспортная сторона логистики, поэтому перейдем к ее более подробному рассмотрению.

1.2 Транспортная логистика

Значительная часть логистических операций на пути движения материального потока осуществляется при помощи различных транспортных средств. Затраты на транспортные операции составляют до половины затрат на логистику в целом. Это говорит о важности оптимизации данной отрасли.

Транспортная логистика – система по организации оптимального способа доставки ресурсов. Оптимальным способом доставки является маршрут, по которому возможно доставить груз в кратчайшие сроки при минимально возможных затратах.

Система доставки грузов обыкновенно осуществляется следующим образом. Для удобства взаимодействия с заказчиком используется типовой бланк заказа, содержащий пункты:

- Информация о заказчике;
- Название и количество груза;
- Места отправления и назначения;
- Время отправления и прибытия;
- Дополнительные услуги.

На основе требований заказчика, а также информации о состоянии транспортной системы, оператор-диспетчер разрабатывает несколько вариантов плана доставки, определяя возможные схемы. Существующие варианты ранжируются по соответствующим критериям, и выбираются наилучшие. [3]

В сложных транспортных системах выделяют также понятие транспортной цепи – этапов перевозки груза на определенные расстояния с использованием транспортных средств одного или нескольких видов.

1.3 Методы оптимизации

Проанализировав понятие транспортной логистики, приходим к выводу, что она базируется на методах оптимизации.

Вопросами оптимизации занимаются различные разделы прикладной математики: исследование операций, математическое моделирование, теория графов, линейное программирование; а также классической математики: теория вероятностей, теория случайных процессов, функциональный анализ.

Рассмотрим некоторые задачи и методы, применяющиеся в практике транспортной логистики.

1.3.1 Метод Свира

Метод Свира заключается в составлении кольцевых маршрутов движения транспорта. Положение потребителя материального потока задается в полярной системе координат. Полюс системы располагается в точке нахождения распределительного склада. Суть метода заключается в том, что полярная ось постепенно вращается против часовой стрелки, покрывая при этом часть координатного поля. Как только сумма заказов покрытых магазинов достигнет вместимости транспортного средства, фиксируется сектор, обслуживаемый одним кольцевым маршрутом, и размечается путь объезда потребителей. Схема метода представлена на рис. 1.1.

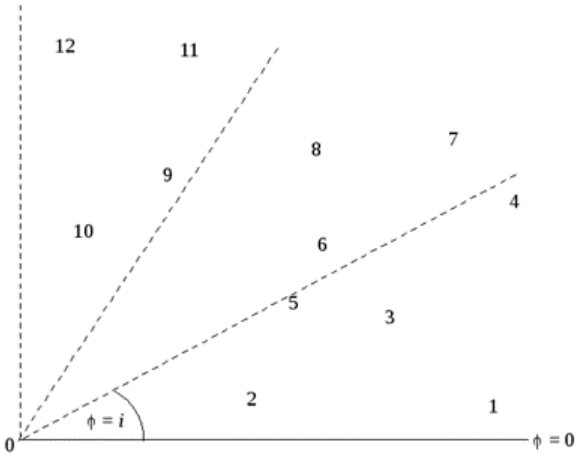


Рисунок 1.1 – Схема метода Свира

1.3.2 Транспортная задача и метод потенциалов

Начнем с постановки условия транспортной задачи. Имеется m пунктов производства и n пунктов потребления однородного продукта. Пусть $a_i > 0$, $i = \overline{1,m}$ – объем производства в пункте i; $b_j > 0$, $j = \overline{1,n}$ – объем потребления в пункте j. Известна матрица $(c_{ij})_{m\times n}$ – матрица транспортных расходов, представляющая стоимость перевозки единицы продукта из i в j.

Требуется составить план перевозок, минимизирующий суммарные затраты, удовлетворяющий всех потребителей и не выходящий за пределы суммарного объема производства. Предполагается, что $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. Это условие разрешимости транспортной задачи. [4]

Обозначим за $x_{ij} \geq 0$ – объем перевозок из i в j, тогда суммарные затраты имеют вид: $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to min$.

Для нахождения решения данной задачи методом потенциалов вводится теорема: пусть $X^0 = (x_{ij}^0)_{m\times n}$ – оптимальный план. Тогда ему соответствует m чисел u_i (потенциалы производителей) и n чисел v_j (потенциалы потребителей) таких, что: $u_i + v_j = \mathrm{c}_{ij} \forall x_{ij}^0 > 0$; $u_i + v_j \leq \mathrm{c}_{ij} \forall x_{ij}^0 = 0$.

Условия этой теоремы удобнее переписать в виде матрицы, отсутствие отрицательных элементов в которой будет говорить об оптимальности решения: $\Delta_{ij} = \left(u_i + v_j - \mathrm{c}_{ij}\right)_{m\times n}$. Для исходной матрицы нужно построить систему потенциалов, опираясь на условие $u_i + v_j = \mathrm{c}_{ij}$. Так как занятых клеток $m+n-1$, то число уравнений на одно меньше, чем число неизвестных. Поэтому одному из потенциалов нужно задать произвольное значение (например, нулевое), тогда остальные потенциалы вычисляются однозначно.

Занятые клетки будут создавать некоторый цикл пересчета. Соответственно, если после очередной итерации метода получаем отрицательное значение в матрице Δ_{ij}, то организуем пересчет по данному

циклу, тем самым сокращая объем товара от невыгодного для данного потребителя поставщика. Как только пересчитываемая матрица будет состоять из неотрицательных значений, она представляет собой оптимальное решение транспортной задачи, т.е. показывает, какое кол-во товаров должен каждый поставщик доставить каждому потребителю.

1.3.3 Задача коммивояжера и метод ветвей и границ

Определим условие задачи коммивояжера (в англоязычной литературе имеет название TSP). Пусть имеется n городов и для каждой пары городов i и j задано расстояние $c_{ij} \geq 0$ между ними. Коммивояжер находится в городе 1, выезжает из этого города, посещает все остальные города по одному разу и возвращается в город 1. Необходимо найти такую последовательность посещения городов, при которой суммарная длина пути минимальна.

При небольших размерностях (n ~ 20) задача может быть за разумное время решена методом ветвей и границ. Суть метода заключается в последовательном разбиении исходного множества возможных решений на ветви – некоторые непересекающиеся подмножества исходного множества. Для каждой из ветвей пересчитывается значение ограничивающей функции. Если это значение хуже известного на данный момент оптимального, то данная ветвь отсекается, обеспечивая тем самым оптимизацию алгоритма полного перебора. Пример ветвления в методе ветвей и границ представлен на рисунке 1.2.

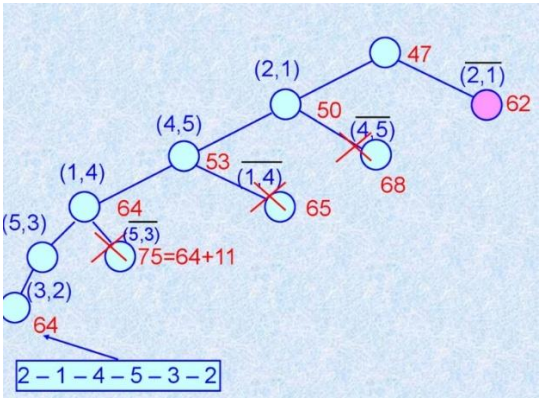


Рисунок 1.2 – Схема метода ветвей и границ

1.4 Анализ подходов к моделированию

Модель данных – это структурное представление элементов данных, их отношений и ограничений в системе управления базами данных. Таким образом, модель данных описывается множеством структур данных, ограничениями целостности и операциями манипулирования с данными.

Традиционно выделяют несколько видов моделей данных:

- Иерархическая;
- Сетевая;
- Реляционная;
- Объектно-ориентированная.

Проанализируем основные различия моделей. Для этого сформируем сравнительную таблицу (см. таблицу 1.1).

Таблица 1.1 - Сравнительная таблица моделей данных

Критерий	**Иерархическая**	**Сетевая**	**Реляционная**	**Объектно-ориентированная**
1 Структура данных	Дерево	Граф	Таблица (набор кортежей)	Дерево объектов
2 Использование памяти компьютера	Эффективное	Эффективное	Значительное	Значительное
3 Скорость выполнения запросов	Высокая	Высокая	Средняя	Низкая
4 Распространенность	Узкая	Ограниченная	Широкая	Широкая
5 Особенности реализации	Громоздкость	Жесткость схемы	Наглядность	Соответствует ООП
6 Понятийная сложность	Высокая	Высокая	Низкая	Высокая

Данные для заполнения таблицы взяты, в том числе, из [5].

В Министерстве Обороны Великобритании в сентябре 2004 г. вышел стандарт DEF STAN 00-60, описывающий правила построения интегрированной логистической поддержки и ставший негласным международным стандартом. Стандарт предписывает использовать реляционную модель данных для описания логистической модели. [6] Таким образом, приходим к выводу, что в рамках задач транспортной логистики возможностей реляционной модели данных будет достаточно.

2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

2.1 Анализ инструментов построения БД

Существует множество СУБД для практической реализации реляционной модели данных. Наиболее распространенными являются SQLite, PostgreSQL и MySQL. Проведем их сравнительный анализ (см таблицу 2.1).

Таблица 2.1 - Сравнительная таблица реляционных СУБД

Критерий	**SQLite**	**PostgreSQL**	**MySQL**
1 Тип архитектуры	Закрытая	Открытая	Открытая
2 Поддержка сообществом	Отсутствует	Активная	Заторможена
3 Широта диалекта SQL	Узкий	Широкий	Широкий
4 SQL-совместимость	Стандартная	По стандарту	Нестандартная
5 Многопоточность	Отсутствует	Поддерживается	Поддерживается
6 Структура СУБД	Файловая	Серверная	Серверная
7 Поддержка ООП	Отсутствует	Присутствует	Отсутствует

Видим, что по различным параметрам лидируют PostgreSQL и MySQL. Однако учтем, что PostgreSQL активно поддерживается сообществом разработчиков, легко масштабируема, а также поддерживает дополнительную функциональность в виде ООП. Это позволяет говорить о ее надежности для целей дальнейшего развития создаваемого продукта. Поэтому в данной работе отдадим предпочтение PostgreSQL.

2.2 Настройка СУБД

Код PostgreSQL открыто предоставляется компанией EDB. Поэтому загружаем установщик СУБД с официального сайта. Для работы с СУБД в формате графического интерфейса используется графический клиент PgAdmin 4. Внешний вид интерфейса PgAdmin 4 представлен на рисунке 2.1.

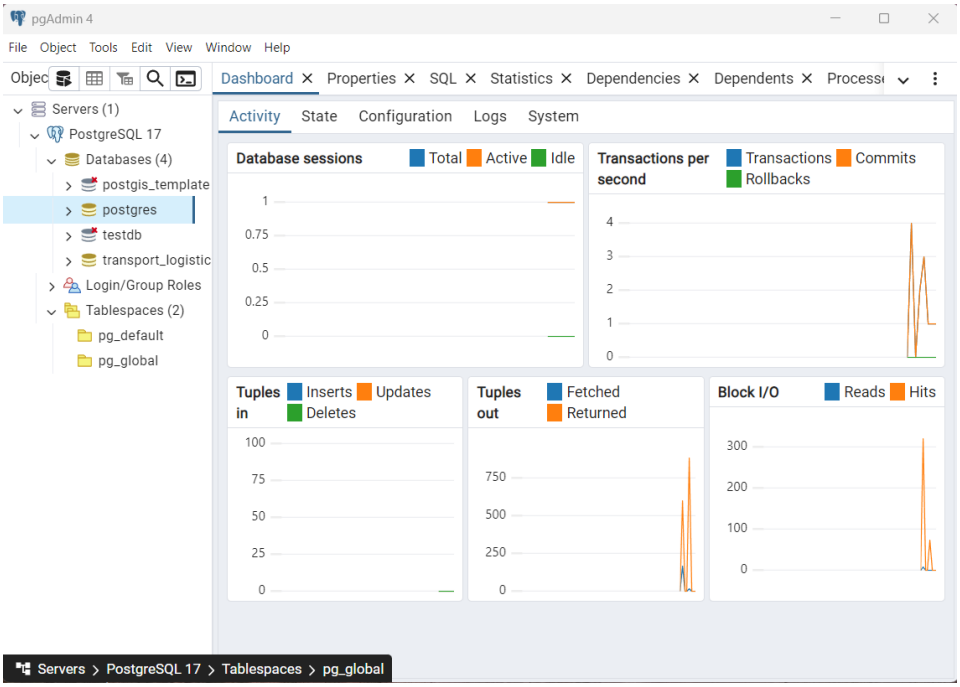


Рисунок 2.1 – Графический интерфейс PgAdmin 4

СУБД предоставляет инструмент запросов – Query Tool, который после выбора конкретной БД может быть вызван сочетанием клавиш alt+shift+Q. Интегрированная среда позволяет писать запросы на языке SQL и получать ответы от БД в виде таблиц (отношений). Инструмент запросов изображен на рис. 2.2.

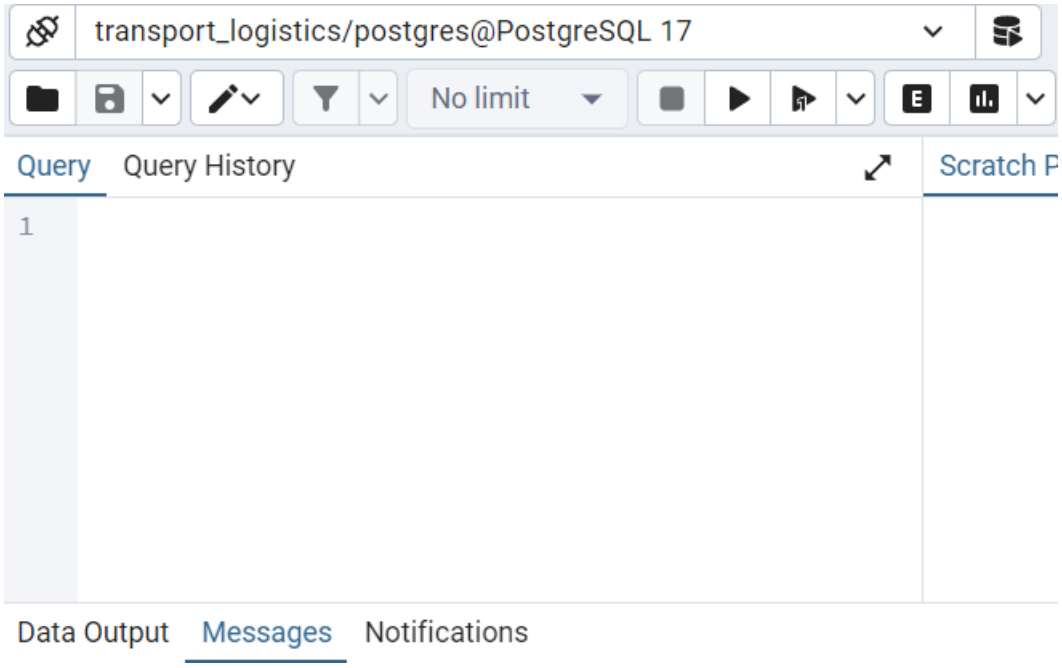


Рисунок 2.2 – Графический интерфейс инструмента запросов

2.3 Привлечение дополнительных модулей

Предваряя проектирование БД модели данных управления транспортной логистикой, подумаем о логике представления геопространства в цифровом виде. Уже существует и широко применяется координатный метод описания точки на карте. Он реализован в геоинформационных системах (GIS).

Примечательно, что для СУБД PostgreSQL разработано расширение PostGIS – открытое ПО, предоставляющее поддержку географических объектов в реляционной БД.

У PostgreSQL есть встроенный мастер установки дополнительных модулей Stack Builder. С его помощью можно установить расширение PostGIS (см. рис. 2.3).

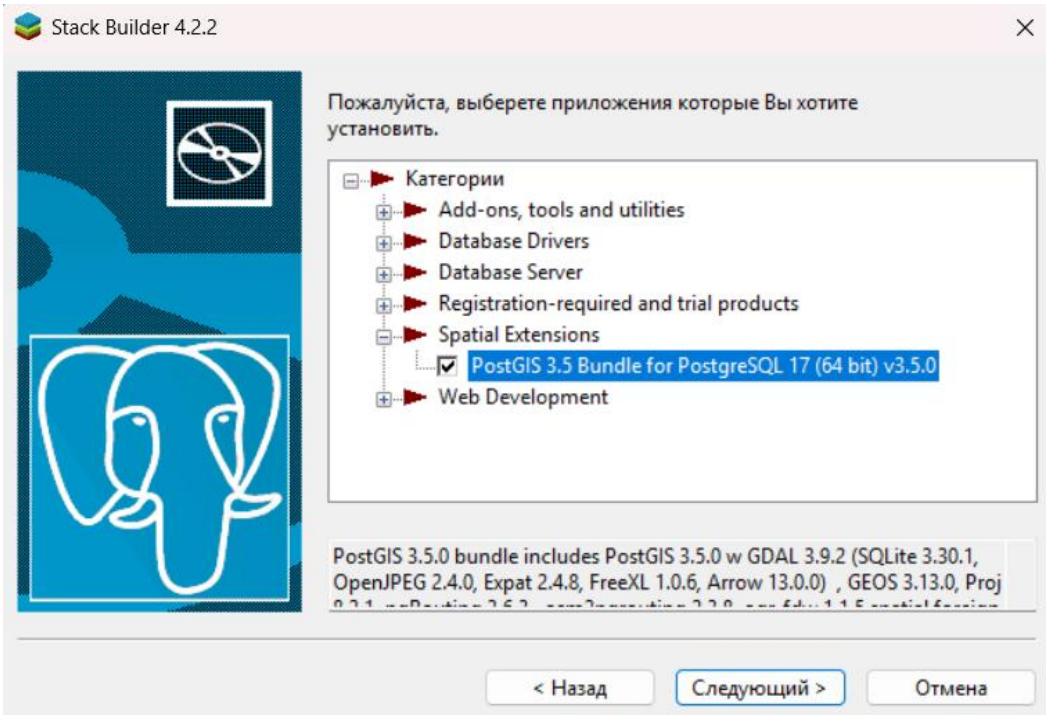


Рисунок 2.3 – Мастер-установщик Stack Builder

В процессе установки была возможность создания шаблона БД для работы с геоданными. Сохраним этот шаблон в БД «postgis_template». Шаблон предоставляет набор базовых таблиц. Их названия представлены на рис. 2.4.

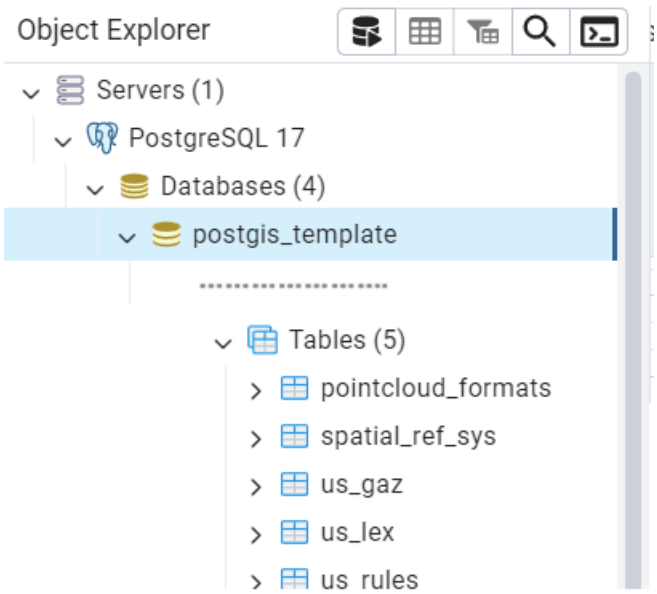


Рисунок 2.4 – PostGIS-шаблон БД

2.4 Проектирование БД

Перейдем непосредственно к построению логической архитектуры БД. Для начала рассмотрим особенности проектирования и определим основные таблицы, которые будут составлять нашу БД.

В модели данных управления транспортной логистикой основными сущностями процесса являются: заказчик, продукт (товар), транспортное средство, склад, маршрут и заказ. Для удобства программирования будем называть эти сущности на английском языке. Рассмотрим подробнее каждую из них.

2.4.1 Сущность – Заказчик

Помимо стандартных атрибутов: id – уникальный идентификатор, name - ФИО и email – электронная почта, заказчик (Customer) описывается геоданными, а именно координатами точки на карте. Т.е. понадобятся также поля latitude и longitude, задающие точную широту и долготу соответственно конечной точки доставки.

2.4.2 Сущность – Товар

Товар (Product) имеет следующие поля: id – уникальный идентификатор, name – наименование, cost – цена товара, weight – вес единицы товара. Данная таблица лишь описывает существующие в данном предприятии виды товаров. Она не хранит информацию об их местонахождении и количестве.

2.4.3 Сущность – Склад

Склад (Storage) описывается полями: id – уникальный идентификатор, latitude – широта точки нахождения склада. longitude – долгота точки нахождения склада, description – описание особенностей склада.

Стоит также отметить, что между складами и товарами ожидается связь many-to-many (многие ко многим), для чего будет необходимо организовать таблицу StorageProduct, которая будет хранить все соответствия. Т.е. id соответствия, fk_storage_id – чужой ключ идентификатора склада, fk_product_id – чужой ключ идентификатора товара и quantity – количество данного товара на этом складе.

2.4.4 Сущность – Транспортное средство

Транспортное средство (Vehicle) описывается полями: id – уникальный идентификатор, license – уникальная лицензия транспорта, type – тип транспортного средства, capacity – грузоподъемность. Для приближения модели к реальности также стоит добавить логическое поле availability, которое будет говорить о статусе доступности транспортного средства.

2.4.5 Сущность – Заказ

Заказ может состоять не из одного вида товара. В таком случае нужно усложнить представление заказов. Определим сущность заказа (Order), которая описывается атрибутами: id – уникальный идентификатор заказа, fk_customer_id – чужой ключ идентификатора заказчика, status – состояние заказа.

Для подробного описания заказа определим таблицу OrderDetail, в которой id – уникальный идентификатор заказа, fk_order_id – чужой ключ идентификатора заказа, fk_product_id – чужой ключ идентификатора товара, quantity – количество единиц данного товара. По сути, данная таблица описывает связь one-to-many (один ко многим) между заказом и его подробным описанием.

2.5 Реализация таблиц и связей

Реализуем описанные сущности на практике. Будем писать SQL-запросы в инструменте запросов СУБД. Для начала определим таблицу Customer. SQL CREATE-запрос представлен на рис. 2.5.

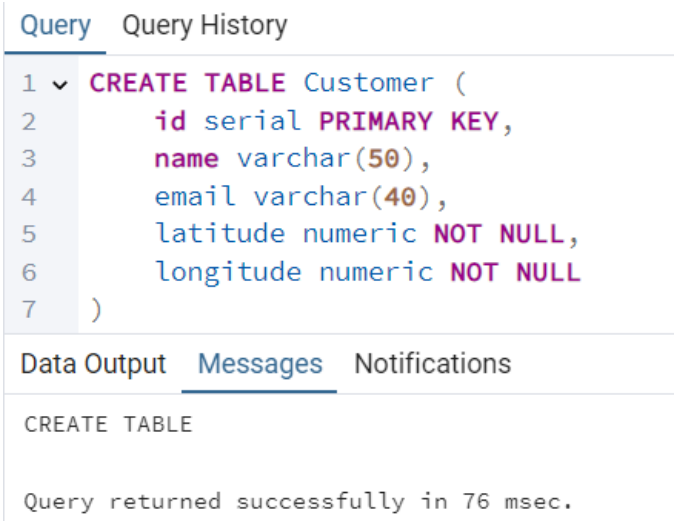


Рисунок 2.5 – Создание таблицы Customer

Подобным образом реализуем и остальные таблицы. На рисунках 2.6 – 2.8 представлено создание таблиц Product, Storage и StorageProduct соответственно. В процессе создания таблиц будем учитывать их связи. Связи один ко многим обеспечиваются указанием ключевого слова REFERENCES и поля, на которое ссылается атрибут.

```
CREATE TABLE Product (
    id serial PRIMARY KEY,
    name varchar(50),
    cost numeric NOT NULL,
    weight numeric NOT NULL
)
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 38 msec.
```

Рисунок 2.6 – Создание таблицы Product

Query | Query History

```
CREATE TABLE Storage (
    id serial PRIMARY KEY,
    latitude numeric NOT NULL,
    longitude numeric NOT NULL,
    description text
)
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 87 msec.
```

Рисунок 2.7 – Создание таблицы Storage

Query | Query History

```
CREATE TABLE StorageProduct (
    id serial PRIMARY KEY,
    fk_storage_id integer REFERENCES Storage(id) NOT NULL,
    fk_product_id integer REFERENCES Product(id) NOT NULL,
    quantity integer NOT NULL
)
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 115 msec.
```

Рисунок 2.8 – Создание таблицы StorageProduct

Статус заказа может принимать только одно из двух значений: «в обработке» либо «закрыт». В этих целях создадим свой перечисляемый тип данных orderStatus и определим таблицу Заказов. Отметим, что слово Order является ключевым словом языка SQL, поэтому назовем общую таблицу заказов OrderGeneral. Запрос на создание типа и таблицы представлен на рис. 2.9.

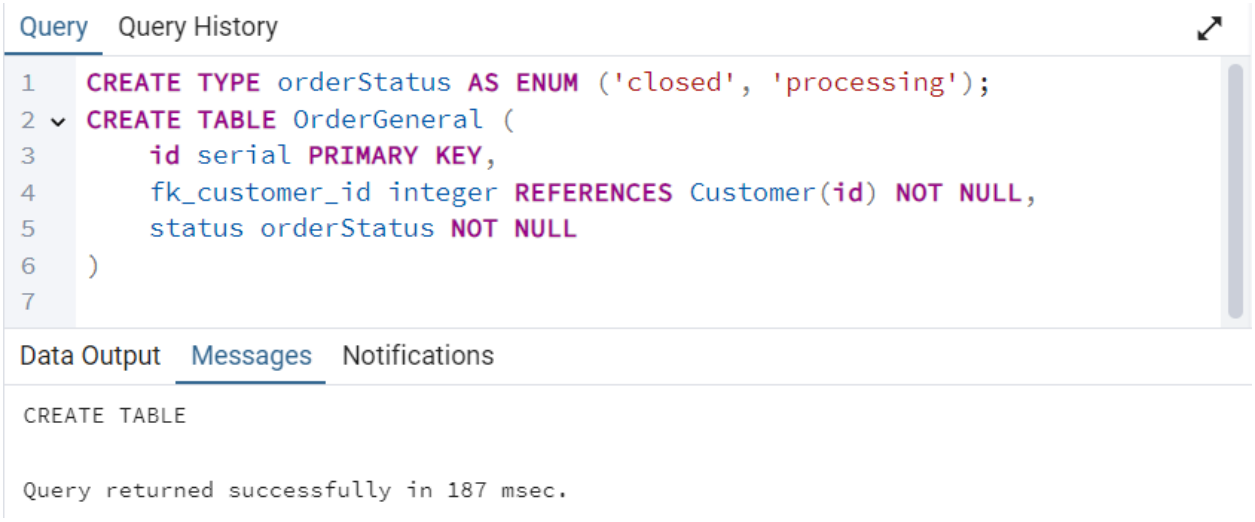


Рисунок 2.9 – Создание таблицы OrderGeneral

Далее требуется реализовать таблицу деталей заказа. Реализация представлена на рис. 2.10.

Рисунок 2.10 – Создание таблицы OrderDetail

Реализуем также и таблицу транспортных средств. Отметим, что атрибут лицензии должен быть уникальным. Запрос представлен на рис. 2.11.

Query Query History

```
1  CREATE TABLE Vehicle (
2      id serial PRIMARY KEY,
3      license varchar(6) UNIQUE NOT NULL,
4      type varchar(20),
5      capacity integer NOT NULL,
6      availability boolean NOT NULL
7  )
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 145 msec.
```

Рисунок 2.11 – Создание таблицы Vehicle

Для проверки правильности задания отношений между таблицами было бы правильным визуализировать эти зависимости. Встроенный в pgAdmin ERD формирует диаграмму, неудобную для чтения. Попытки наладить гладкость связей в диаграмме не увенчались успехом. Пример «неудачной» диаграммы представлен на рис. 2.12.

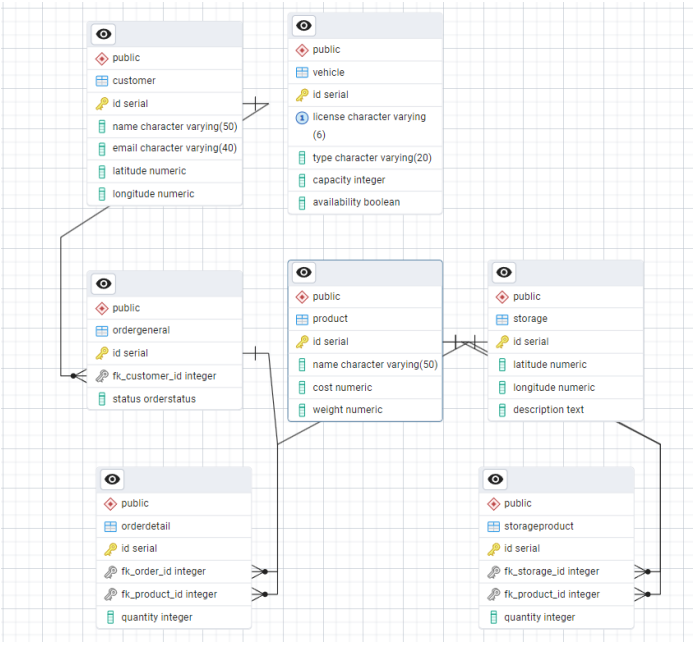


Рисунок 2.12 – Неудачный вариант ERD

Тем не менее, визуализация БД очень важна. Поэтому было принято решение воспользоваться средством pgModeler. Визуализатор позволяет делать гибкие и подробные диаграммы БД. Реализация ERD через pgModeler представлена на рис. 2.13.

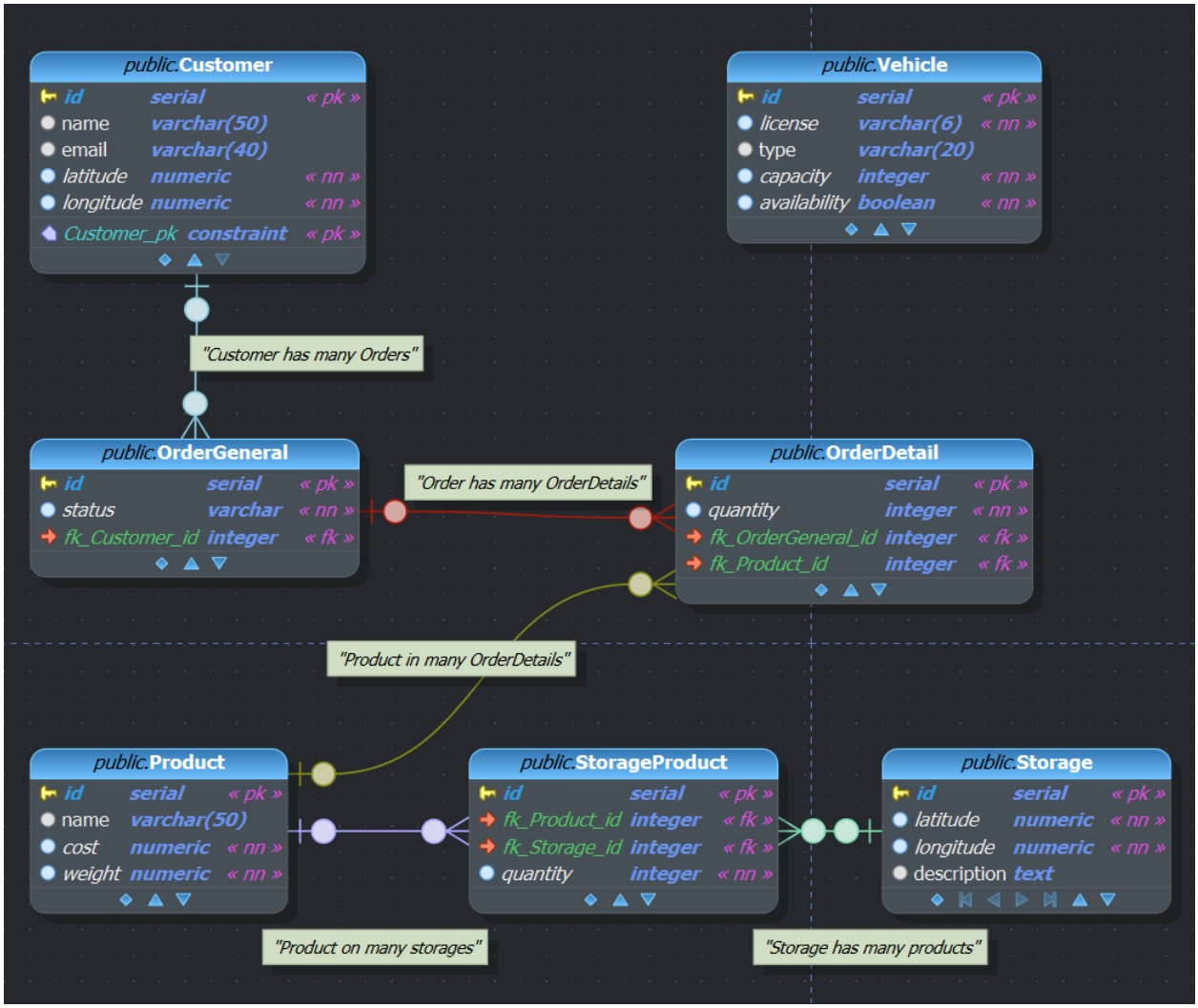


Рисунок 2.13 – Диаграмма БД в pgModeler

2.6 Заполнение БД тестовыми данными

Чтобы иметь возможность в дальнейшем тестировать алгоритмы оптимизации доставки, необходимо заполнить нашу БД изначальными данными. Реализуем заполнение через SQL-запросы. Изображения запросов представлены на рисунках 2.14-2.20.

Query Query History

```
INSERT INTO Customer (id, name, email, latitude, longitude) VALUES
(1, 'Иван Иванов', 'ivan.ivanov@mail.ru', 55.7558, 37.6173),
(2, 'Мария Смирнова', 'maria.smirnova@mail.ru', 55.8094, 37.5075),
(3, 'Алексей Кузнецов', 'alex.kuznetsov@mail.ru', 55.8315, 37.4951),
(4, 'Ольга Соколова', 'olga.sokolova@mail.ru', 55.7532, 37.6225),
(5, 'Дмитрий Орлов', 'dmitry.orlov@mail.ru', 55.8425, 37.6517),
(6, 'Татьяна Крылова', 'tatyana.krylova@mail.ru', 55.7106, 37.7205),
(7, 'Сергей Новиков', 'sergey.novikov@mail.ru', 55.7681, 37.5695),
(8, 'Екатерина Морозова', 'katya.morozova@mail.ru', 55.6890, 37.8529),
(9, 'Андрей Фролов', 'andrey.frolov@mail.ru', 55.7516, 37.4424),
(10, 'Наталья Васильева', 'natalia.v@mail.ru', 55.8348, 37.3224);
```

Data Output Messages Notifications

```
INSERT 0 10

Query returned successfully in 86 msec.
```

Рисунок 2.14 – Заполнение таблицы Customer

Query Query History

```
INSERT INTO Product (id, name, cost, weight) VALUES
(1, 'Стол обеденный деревянный', 15000, 30.5),
(2, 'Стул офисный эргономичный', 3500, 7.8),
(3, 'Диван угловой 3-х местный', 45000, 80.0),
(4, 'Кровать двуспальная', 32000, 70.0),
(5, 'Тумба прикроватная', 4500, 12.5),
(6, 'Шкаф-купе 3-дверный', 52000, 100.0),
(7, 'Полка настенная деревянная', 2000, 4.3),
(8, 'Стол компьютерный угловой', 12000, 25.0),
(9, 'Комод 4 ящика', 18000, 45.5),
(10, 'Кресло-кровать раскладное', 25000, 60.0);
```

Data Output Messages Notifications

```
INSERT 0 10

Query returned successfully in 78 msec.
```

Рисунок 2.15 – Заполнение таблицы Products

Query Query History

```
INSERT INTO Storage (id, latitude, longitude, description) VALUES
(1, 55.751244, 37.618423, 'Центральный склад'),
(2, 55.796289, 37.537611, 'Северный склад'),
(3, 55.707014, 37.653352, 'Южный склад'),
(4, 55.773239, 37.580284, 'Западный склад'),
(5, 55.743662, 37.656177, 'Восточный склад');
```

Data Output Messages Notifications

```
INSERT 0 5

Query returned successfully in 73 msec.
```

Рисунок 2.16 – Заполнение таблицы Storage

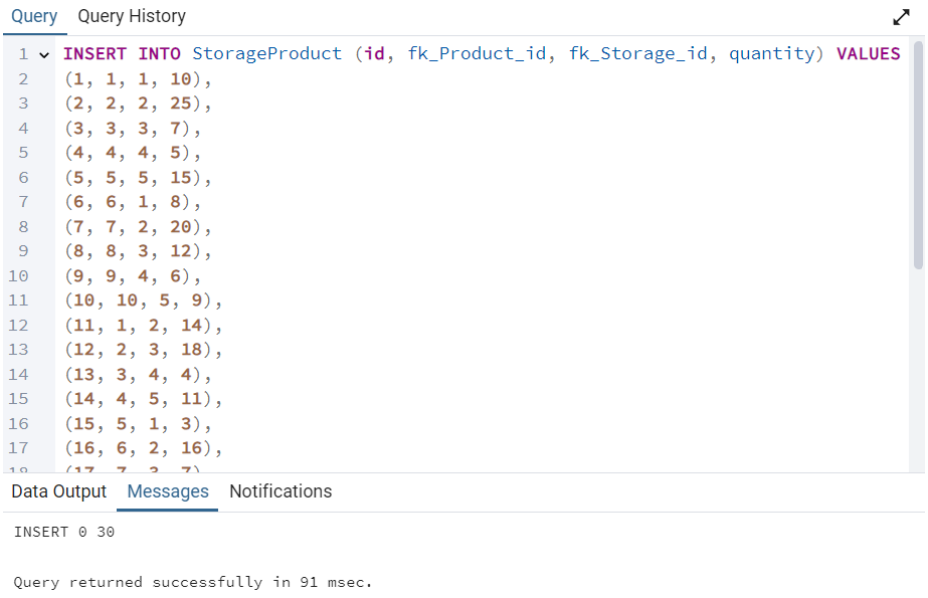


Рисунок 2.17 – Заполнение таблицы StorageProduct

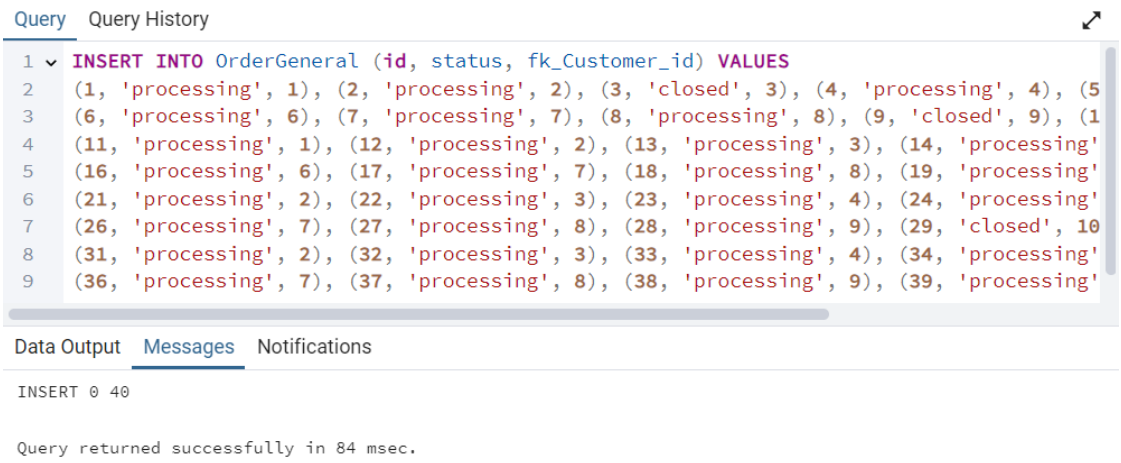


Рисунок 2.18 – Заполнение таблицы OrderGeneral

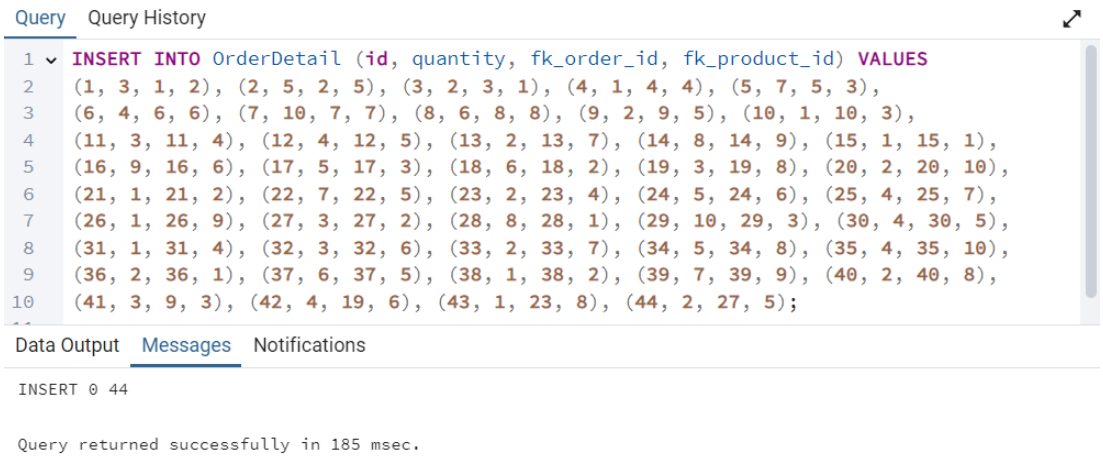


Рисунок 2.19 – Заполнение таблицы OrderDetail

Рисунок 2.20 – Заполнение таблицы Vehicle

2.7 Формирование базовых запросов

Сформируем запросы для управления базой данных. Сначала опишем некоторые базовые запросы.

Рассмотрим поиск ближайшего склада к конкретному заказчику. Для данного запроса будем применять встроенные функции PostGIS. ST_MakePoint создает географическую точку для дальнейшей работы. ST_SetSRID(point, 4326) устанавливает систему координат WGS84. ST_Distance высчитывает расстояние между двумя географическими точками. Реализация запроса и форма ответа представлены на рисунке 2.21.

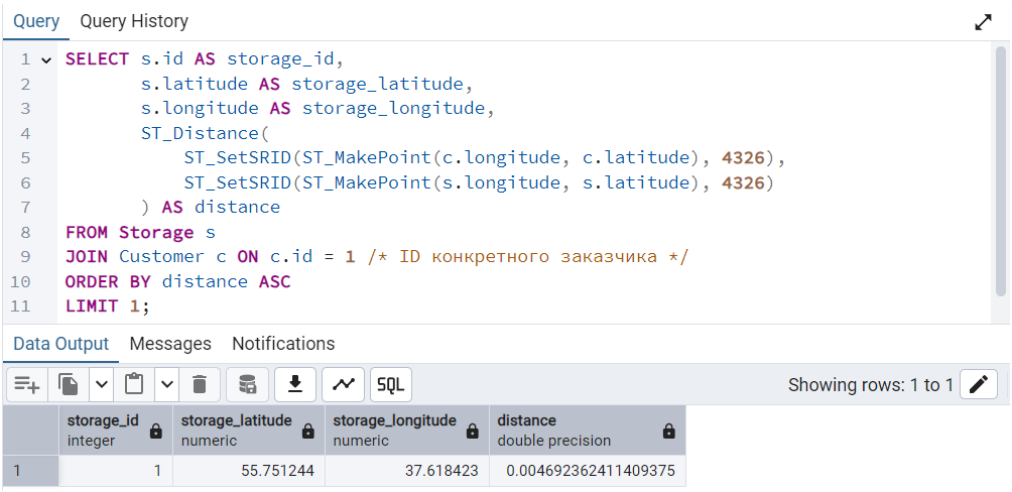


Рисунок 2.21 – Запрос на поиск ближайшего к заказчику склада

Чтобы было удобно ориентироваться в содержании заказа, напишем запрос, который будет возвращать все товары в том кол-ве, в котором они составляют данный заказ. Реализация представлена на рис. 2.22. А результат тестирования показан на рис 2.23.

Query | Query History

```
1  SELECT
2      o.id AS order_id,
3      c.name AS customer_name,
4      p.name AS product_name,
5      od.quantity AS product_quantity,
6      p.cost AS product_cost,
7      (p.cost * od.quantity) AS total_cost,
8      p.weight AS product_weight,
9      (p.weight * od.quantity) AS total_weight
10 FROM
11     OrderGeneral o
12 JOIN
13     Customer c ON o.fk_customer_id = c.id
14 JOIN
15     OrderDetail od ON o.id = od.fk_order_id
16 JOIN
17     Product p ON od.fk_product_id = p.id
18 WHERE
19     o.id = 9; /* ID заказа */
20
```

Рисунок 2.22 – Запрос на вывод содержания заказа

Data Output | Messages | Notifications

Showing rows: 1 to 2 | Page No: 1 of 1

	order_id integer	customer_name character varying (50)	product_name character varying (50)	product_quantity integer	product_cost numeric	total_cost numeric	product_weight numeric	total_weight numeric
1	9	Андрей Фролов	Диван угловой 3-х местный	3	45000	135000	80.0	240.0
2	9	Андрей Фролов	Тумба прикроватная	2	4500	9000	12.5	25.0

Рисунок 2.23 – Результат выдачи товаров в заказе

2.8 Оптимизационные запросы

2.8.1 Подбор оптимального транспортного средства

Рассмотрим задачу подбора самого удобного транспортного средства для осуществления доставки товара. Критерием пригодности транспорта является факт вместимости в него заказа целиком.

Для начала нужно сформировать суммарный вес, который затрачивает данный заказ в целом. Для этого будем в таблице OrderDetail искать

соответствующие элементы заказа и формировать суммарные веса для каждой категории.

Определив суммарный вес заказа, можем перейти к рассмотрению характеристик транспортных средств. А именно: будем среди доступных автомобилей искать те, чья грузоподъемность не меньше суммарного веса заказа. Реализация запроса представлена на рис. 2.24. А тестирование на рис. 2.25.

Query | Query History

```
WITH OrderTotalWeight AS (
    SELECT
        od.fk_order_id AS order_id,
        SUM(p.weight * od.quantity) AS total_weight
    FROM
        OrderDetail od
    JOIN
        Product p ON od.fk_product_id = p.id
    WHERE
        od.fk_order_id = 1
    GROUP BY
        od.fk_order_id
),
SuitableVehicles AS (
    SELECT
        v.id AS vehicle_id,
        v.type AS vehicle_type,
        v.capacity AS vehicle_capacity,
        v.availability
    FROM
        Vehicle v
    JOIN
        OrderTotalWeight otw ON v.capacity >= otw.total_weight
    WHERE
        v.availability = TRUE
)
SELECT
    sv.vehicle_id,
    sv.vehicle_type,
    sv.vehicle_capacity
FROM
    SuitableVehicles sv
ORDER BY
    sv.vehicle_capacity ASC
LIMIT 1;
```

Рисунок 2.24 – Запрос на подбор оптимального транспортного средства

Data Output | Messages | Notifications

	vehicle_id integer	vehicle_type character varying (20)	vehicle_capacity integer
1	5	ИЖ-2717	95

Рисунок 2.25 – Результат нахождения оптимального транспорта

2.8.2 Метод ближайшего соседа для распределения товаров

Пусть стоит задача распределения ресурсов складов так, чтобы все заказы были удовлетворены и при этом заказчики получали товары с ближайших к ним складов.

Декомпозируем задачу на части. Сначала сформируем таблицу с заказами и координатами, куда их нужно доставить (см. рис. 2.26).

Query | Query History

```
1   /* Формирование таблицы с заказами и их координатами */
2   WITH OrderCustomer AS (
3       SELECT
4           o.id AS order_id,
5           c.id AS customer_id,
6           c.latitude AS customer_latitude,
7           c.longitude AS customer_longitude
8       FROM OrderGeneral o
9       JOIN Customer c ON o.fk_customer_id = c.id
10  ),
```

Рисунок 2.26 – Формирование таблицы координат доставки заказов

Затем сформируем таблицу из всех пунктов заказа и их наличия в достаточном количестве на складах. Реализация представлена на рис.2.27.

```
12  /* Формирование таблицы товаров и их наличия на складах */
13  OrderDetailsWithAvailability AS (
14      SELECT
15          od.fk_order_id AS order_id,
16          od.fk_product_id AS product_id,
17          od.quantity AS order_quantity,
18          sp.fk_storage_id AS storage_id,
19          sp.quantity AS available_quantity,
20          s.latitude AS storage_latitude,
21          s.longitude AS storage_longitude
22      FROM OrderDetail od
23      JOIN StorageProduct sp ON od.fk_product_id = sp.fk_product_id
24      JOIN Storage s ON sp.fk_storage_id = s.id
25      WHERE sp.quantity >= od.quantity /* Товара на складе достаточно */
26  ),
```

Рисунок 2.27 – Создание соответствия доступных складов и запрашиваемых товаров

Теперь определим таблицу расстояний между заказчиками и складами. Для этого будем пользоваться встроенными функциями PostGIS (см. рис. 2.28)

```
/* Вычисление расстояния между заказчиками и складами */
CustomerStorageDistance AS (
    SELECT
        oc.order_id,
        odwa.product_id,
        odwa.storage_id,
        oc.customer_id,
        odwa.order_quantity,
        odwa.available_quantity,
        ST_DistanceSphere(
            ST_MakePoint(oc.customer_longitude, oc.customer_latitude),
            ST_MakePoint(odwa.storage_longitude, odwa.storage_latitude)
        ) AS distance
    FROM OrderCustomer oc
    JOIN OrderDetailsWithAvailability odwa ON oc.order_id = odwa.order_id
),
```

Рисунок 2.28 – Формирование таблицы расстояний

Методом ближайшего соседа найдем ближайшие склады для каждого пункта заказа. Реализация представлена на рис. 2.29.

```
/* Нахождение ближайшийего склада для каждого пункта заказа */
ClosestStorage AS (
    SELECT
        csd.order_id,
        csd.product_id,
        csd.storage_id,
        csd.distance,
        ROW_NUMBER() OVER (PARTITION BY csd.order_id, csd.product_id ORDER BY csd.distance ASC) AS rn
    FROM CustomerStorageDistance csd
),
```

Рисунок 2.29 – Поиск ближайших складов для каждого пункта заказа

Выберем из таблицы ближайших складов только те, которые являются наиболее оптимальными (см. рис. 2.30).

```
/* Фильтрация ближайших складов */
BestStorageForOrder AS (
    SELECT
        order_id,
        product_id,
        storage_id,
        distance
    FROM ClosestStorage
    WHERE rn = 1
)
```

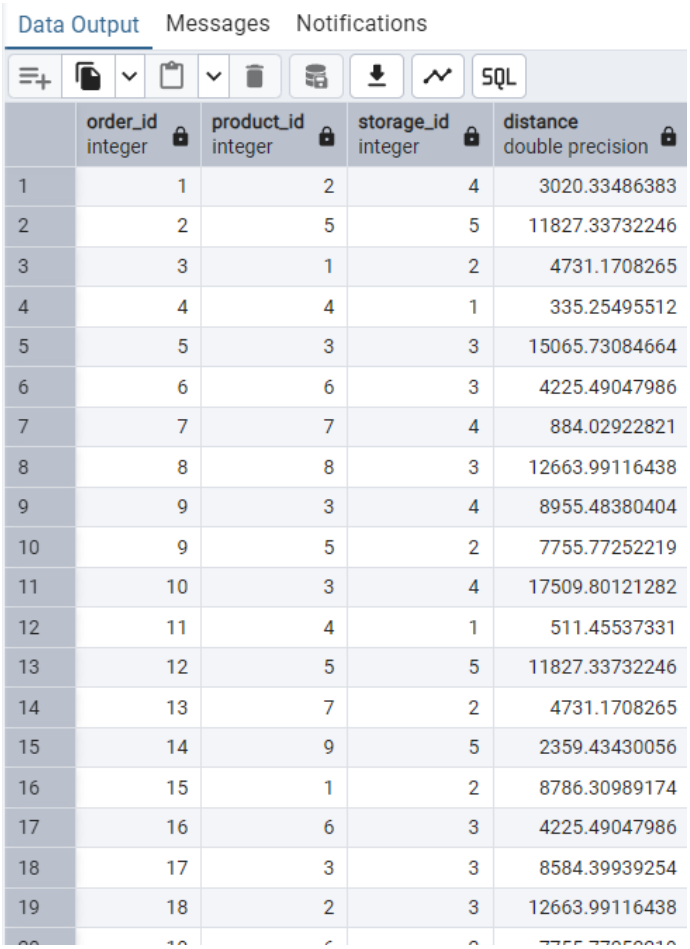
Рисунок 2.30 – Выборка ближайших складов

Осталось вывести результат нахождения оптимального распределения товаров со складов (рис. 2.31).

```
67  /* Вывод результата */
68  SELECT
69      bso.order_id,
70      bso.product_id,
71      bso.storage_id,
72      bso.distance
73  FROM BestStorageForOrder bso
74  ORDER BY bso.order_id, bso.product_id;
```

Рисунок 2.31 – Вывод результата оптимального распределения

Протестируем данный запрос на наших тестовых данных. Для каждого заказа и его пунктов получим id ближайшего склада, который способен обеспечить данный товар в требуемом объеме. Результат тестирования представлен на рис. 2.32.



Data Output | Messages | Notifications

SQL

	order_id integer	product_id integer	storage_id integer	distance double precision
1	1	2	4	3020.33486383
2	2	5	5	11827.33732246
3	3	1	2	4731.1708265
4	4	4	1	335.25495512
5	5	3	3	15065.73084664
6	6	6	3	4225.49047986
7	7	7	4	884.02922821
8	8	8	3	12663.99116438
9	9	3	4	8955.48380404
10	9	5	2	7755.77252219
11	10	3	4	17509.80121282
12	11	4	1	511.45537331
13	12	5	5	11827.33732246
14	13	7	2	4731.1708265
15	14	9	5	2359.43430056
16	15	1	2	8786.30989174
17	16	6	3	4225.49047986
18	17	3	3	8584.39939254
19	18	2	3	12663.99116438

Рисунок 2.32 – Результат нахождения ближайших подходящих складов

2.8.3 Задача коммивояжера для одного транспортного средства

Рассмотрим задачу, в которой данному транспортному средству требуется объехать нескольких заказчиков. Требуется найти такой порядок объезда заказчиков, чтобы суммарное пройденное расстояние было минимальным.

Разобьем данную задачу на подзадачи. Для начала определим необходимые таблицы. На основе заданного значения id транспортного средства будем формировать кортеж, который описывает начальную точку маршрута. Код SQL-запроса представлен на рис. 2.33.

Query | Query History

```
1  /* Определение начальной точки маршрута */
2  WITH VehicleStart AS (
3      SELECT
4          v.id AS vehicle_id,
5          s.id AS start_point,
6          s.latitude AS start_latitude,
7          s.longitude AS start_longitude
8      FROM
9          Vehicle v
10     JOIN
11         Storage s ON s.id = 2
12     WHERE
13         v.id = 1 /* ID транспорта */
14 ),
```

Рисунок 2.33 – Определение начальной точки маршрута

Затем формируем таблицу из точек маршрута, т.е. из точек, в которых располагаются заказчики. Пользователь должен указать список заказов, которые должны быть выполнены. Реализация представлена на рис. 2.34. Затем реализуем матрицу расстояний между всеми точками. Реализация представлена на рис. 2.35.

```
/* Формируем точки маршрута */
RoutePoints AS (
    SELECT
        c.id AS point_id,
        c.latitude,
        c.longitude,
        'customer' AS point_type
    FROM
        OrderGeneral o
    JOIN
        Customer c ON o.fk_customer_id = c.id
    WHERE
        o.id IN (2, 4, 5, 7) /* Список ID заказов, которые необходимо доставить */
),
```

Рисунок 2.34 – Формирование точек маршрута

```
/* Вычисляем расстояния между всеми точками (включая склад) */
DistanceMatrix AS (
    SELECT
        rp1.point_id AS start_point,
        rp2.point_id AS end_point,
        ST_DistanceSphere(
            ST_MakePoint(rp1.longitude, rp1.latitude),
            ST_MakePoint(rp2.longitude, rp2.latitude)
        ) AS distance
    FROM (
        SELECT
            start_point AS point_id,
            start_latitude AS latitude,
            start_longitude AS longitude
        FROM
            VehicleStart
        UNION ALL
        SELECT
            point_id,
            latitude,
            longitude
        FROM
            RoutePoints
    ) rp1
    CROSS JOIN (
        SELECT
            start_point AS point_id,
            start_latitude AS latitude,
            start_longitude AS longitude
        FROM
            VehicleStart
        UNION ALL
        SELECT
            point_id,
            latitude,
            longitude
        FROM
            RoutePoints
    ) rp2
    WHERE
        rp1.point_id != rp2.point_id
),
```

Рисунок 2.35 – Формирование матрицы расстояний

В матрице расстояний будем выбирать самые короткие расстояния между соседними точками маршрута. Реализация нахождения оптимального маршрута представлена на рис. 2.36.

```
/* Итеративное построение маршрута */
OptimalRoute AS (
    SELECT
        dm.start_point,
        dm.end_point,
        dm.distance,
        ROW_NUMBER() OVER (PARTITION BY dm.start_point ORDER BY dm.distance ASC) AS rn
    FROM
        DistanceMatrix dm
)
```

Рисунок 2.36 – Построение оптимального маршрута

Ну и наконец, выведем полученный оптимальный маршрут. См. рис. 2.37.

```
/* Вывод маршрута */
SELECT
    or1.start_point,
    or1.end_point,
    or1.distance
FROM
    OptimalRoute or1
WHERE
    or1.rn = 1 -- Выбираем ближайшую точку для каждой стартовой точки
ORDER BY
    or1.start_point;
```

Рисунок 2.37 – Вывод полученного маршрута

Протестируем работу запроса. Зададим транспортное средство и список из заказов, которые оно должно развезти. Получим следующий ответ (см. рис. 2.38). Он описывает маршрут, состоящий из ребер наиболее оптимальной длины.

Data Output | Messages | Notifications

SQL

	start_point integer	end_point integer	distance double precision
1	2	7	3714.96958658
2	4	7	3706.77297165
3	5	2	8786.30989174
4	7	4	3706.77297165

Рисунок 2.38 – Тестирование запроса задачи коммивояжера

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была изучена предметная область логистики. Особое внимание уделялось задачам транспортной логистики. Для формального описания оптимизационных процессов были изучены различные методы математической оптимизации задач транспортной логистики. Отдельно были проанализированы существующие методы создания модели данных.

На основе изученной информации было принято решение о формировании реляционной модели данных. Анализ существующих СУБД показал целесообразность использования системы PostgreSQL. В качестве аспекта структуры модели данных были сформированы сущности (таблицы), а также оформлены связи между ними. Для поддержания аспекта манипуляции посредством создания SQL-запросов была реализована функциональность оптимизационной обработки логистических данных.

Помимо основных задач была также проведена работа по визуализации БД средствами инструмента графической визуализации pgModeler. Отдельно изучалось расширение для работы с геоданными – PostGIS.

Тестирование работы оптимизационных SQL-запросов показало корректность работы модели, что позволяет нам признать данную курсовую работу успешной.

Пожалуй, транспортная логистика – одна из ключевых областей современной экономики. А потому модель данных оптимизации процессов транспортной логистики еще долго будет оставаться востребованным инструментом экономистов и аналитиков.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Логистика: Учебник / Под ред. Б.А. Аникина: 3-е изд., перераб. и доп. – М.: Л69 ИНФРА-М, 2002. – 368 с.

2. Шумаев, В. А. Основы логистики: учеб. пособие / В. А. Шумаев. – М.: Юридический институт МИИТ, 2016. – 314 с.

3. Титов, Б. А. Транспортная логистика [Электронный ресурс]: электрон, учеб. пособие / Б. А. Титов; Минобрнауки России, Самар, гос. аэрокосм, ун-т им. С. П. Королева (нац. исслед. ун-т). – Самара, 2012.

4. Задачи транспортного типа: учебное пособие по дисциплине «Математическое моделирование»/ Ефимов Р.А., Иванова А.П. – М.: РУТ (МИИТ), Янус-К, 2023. – 112 с.

5. Сергеева Т.И. Базы данных: модели данных, проектирование, язык SQL: учеб. пособие / Т.И. Сергеева, М.Ю. Сергеев. Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2012. 233 с.

6. Defence Standard 00-60. Integrated Logistic Support. Part 3: Guidance for Application of Software Support / Ministry of Defence of Great Britain, London, 2004. 64p.

7. PostgreSQL 17.2 Documentation: [Электронный ресурс]. URL: https://www.postgresql.org/docs/current/index.html (Дата обращения: 10.12.2024)

8. PostgreSQL Database Modeler Documentation: [Электронный ресурс]. URL: https://pgmodeler.io/support/docs (Дата обращения: 10.12.2024)