Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для овощного магазина»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы
и программирование»
Петренко Кирилл Константинович
подпись: _____

Проверил:
преподаватель дисциплины,
преподаватель ЧУПО «ВШП»,
к.ф.н. Ткачев П.С.
оценка: _____
подпись: _____

Тверь, 2025 г.

Оглавление

Введение 3-6

1. Глава 1. Теоретические основы разработки баз данных

1.1. Понятие и назначение баз данных 7-8

1.2. Модели данных и выбор модели для проекта 9-11

1.3. Основы проектирования реляционных баз данных 12-13

1.4. язык SQL и его роль управлениями базами данных 14-15

1.5. Обзор средств разработки и управления базами данных 16-18

2. Глава 2. Практическая реализация базы данных овощного магазина

2.1. Анализ предметной области и постановка задачи 19-23

2.2. Реализация базы данных 23-26

2.3. Реализация основных SQL-запросов 27-30

2.4. Создание хранимых процедур, триггеров и представлений 30-33

3. Заключение 34-38

4. Список литературы 39

Введение

Актуальность темы

Информационные технологии играют ключевую роль в различных сферах жизни, включая бизнес. В частности, базы данных стали важным инструментом для компаний, которые стремятся повысить свою эффективность. Это особенно актуально для сектора розничной торговли, который занимает важное место в экономике. Внедрение автоматизированных систем учета позволяет предпринимателям быстро и точно отслеживать информацию о товарах, поставках, клиентах и продажах. Современные реляционные базы данных существенно улучшили процессы учета и поддержку принятия управленческих решений.

Особенно важную роль система учета играет в розничной торговле овощами. Овощи — это товар с ограниченным сроком хранения, что требует точного контроля за запасами, а также регулярных поставок и продаж. Ошибки в учете могут привести не только к финансовым убыткам, но и к утрате доверия со стороны клиентов, что, в свою очередь, может негативно сказаться на репутации магазина и снизить его конкурентоспособность. Таким образом, внедрение базы данных для учета товаров, поставок, клиентов и продаж становится не только важным, но и необходимым шагом для розничных предприятий, работающих в сфере реализации овощей.

Кроме того, автоматизация учета товаров и связанных с ними процессов значительно повышает эффективность работы магазина. Внедрение базы данных ускоряет и упрощает различные операции, такие как оформление продаж, поиск товаров по запросу, обновление информации о наличии и ценах. Это также способствует повышению прозрачности бизнес-процессов, снижению ошибок, вызванных человеческим фактором, и значительному сокращению времени, затрачиваемого на поиск и

обработку данных. В условиях современной экономики такие преимущества играют ключевую роль в обеспечении успешного функционирования бизнеса.

Цель и задачи исследования

Главной целью исследования является разработка и внедрение базы данных для овощного магазина, которая обеспечит эффективное управление учетными процессами, включая учет товаров, информацию о клиентах, поставках и продажах. Для достижения этой цели необходимо решить ряд ключевых задач:

1. Анализ предметной области и сбор требований. Необходимо провести анализ бизнес-процессов магазина, выявив основные потребности, которые база данных должна удовлетворять.
2. Разработка структуры базы данных. Следующий этап заключается в создании концептуальной и логической модели базы данных, с определением таблиц и их взаимосвязей.
3. Реализация таблиц и связей. После проектирования будет построена физическая структура базы данных, включая описание таблиц, атрибутов и ключей.
4. Создание SQL-запросов. Для извлечения и обработки данных будут разработаны запросы, которые помогут в формировании отчетности, анализе товарных запасов и продаж.
5. Разработка хранимых процедур и триггеров. Для автоматизации операций в базе данных будут созданы хранимые процедуры и триггеры, например, для обновления остатков товаров после каждой продажи.

Объект и предмет исследования

Объектом исследования является информационная система учета, разработанная для овощного магазина. Она представляет собой совокупность баз данных, приложений и процессов, ориентированных на оптимизацию управления товарами, продажами, клиентами и поставками. Система включает в себя различные компоненты, такие как база данных для хранения информации о товарах и клиентах, приложения для обработки запросов и создания отчетности, а также инструменты для управления доступом и обеспечения безопасности данных.

Предмет исследования — реляционная база данных, которая обеспечит хранение, обработку и управление данными, связанными с деятельностью магазина. В такой базе информация организована в таблицы, что позволяет эффективно структурировать данные и устанавливать связи между различными элементами. Реляционные базы данных пользуются популярностью в бизнесе благодаря своей гибкости, надежности и масштабируемости, что делает их идеальным инструментом для организации учета в торговых компаниях.

Методы и средства разработки

Для разработки базы данных будет применена реляционная модель данных, которая является стандартом для большинства современных информационных систем. Реляционные базы данных обеспечивают отличную гибкость, простоту в использовании и масштабируемость, что делает их идеальным решением для бизнеса, который может расти и развиваться. Основным инструментом для работы с базой данных будет язык SQL, признанный универсальным стандартом для реляционных систем, и предоставляющий обширные возможности для создания таблиц, выполнения запросов и обработки данных.

В качестве системы управления базами данных будет выбрана MySQL — популярная СУБД с открытым исходным кодом, которая позволяет эффективно работать с большими объемами данных. Она обеспечивает

стабильную работу, высокую производительность и надежность, что критично для обработки данных в реальном времени. Для проектирования и управления базой данных будут использованы инструменты, такие как MySQL Workbench, который позволяет визуально разрабатывать схему базы данных.

Глава 1. Теоретические основы разработки баз данных

1.1. Понятие и назначение баз данных

База данных (БД) представляет собой упорядоченную коллекцию данных, предназначенную для их хранения, обработки и управления с целью дальнейшего использования. Главная задача базы данных заключается в эффективном хранении, поиске, обновлении и удалении информации, предоставляя доступ к данным в удобной и структурированной форме, что облегчает взаимодействие с ними для пользователей или приложений.

В бизнес-сфере базы данных играют ключевую роль в управлении и анализе информации, связанной с товарами, продажами, клиентами, поставками и другими аспектами операционной деятельности. Например, в супермаркетах базы данных управляют данными о товарах, их количестве, цене и поставщиках. Это позволяет оперативно обновлять информацию о наличии товаров на складе, отслеживать продажи и принимать решения относительно закупок и изменения цен.

В государственном управлении базы данных используются для хранения и обработки данных о гражданах, транзакциях, а также для управления государственными ресурсами. Например, налоговые базы данных содержат сведения о налогоплательщиках, их декларациях и налоговых начислениях. Базы данных также применяются в здравоохранении для учета медицинских карт пациентов, в образовании для хранения информации о студентах и в правоохранительных органах для учета правонарушений.

Базы данных играют важнейшую роль в эффективной организации информационных систем, предоставляя структурированное хранилище данных, что значительно облегчает их поиск, обработку и анализ. Современные информационные системы, будь то в бизнесе или в государственных учреждениях, не могут функционировать без использования баз данных, так как они обеспечивают целостность и доступность данных в реальном времени.

В бизнесе базы данных способствуют ускорению и автоматизации таких процессов, как обработка заказов, управление запасами и анализ потребительских предпочтений. Благодаря базе данных компании могут улучшать прогнозирование спроса, уменьшать затраты на хранение данных и повышать качество обслуживания клиентов. Например, розничные сети могут интегрировать данные о покупках и анализировать

их для более точного планирования ассортимента и выявления популярных товаров.
В государственном управлении базы данных способствуют обеспечению прозрачности и подотчетности, улучшению управления государственными ресурсами, такими как социальные выплаты или налоги, и эффективной поддержке гражданских услуг. В здравоохранении базы данных поддерживают актуальность информации о медицинских процедурах, рецептах и результатах анализов, что упрощает работу медицинского персонала и позволяет оперативно предоставлять услуги пациентам.
Использование баз данных в этих сферах подчеркивает важность централизованного подхода к обработке данных, что способствует повышению эффективности и безопасности работы с информацией.
На сегодняшний день базы данных являются неотъемлемой частью всех современных информационных систем. Они лежат в основе таких сфер, как финансовые услуги, управление цепочками поставок, здравоохранение, электронная коммерция и другие. Современные базы данных обеспечивают гибкость, масштабируемость и высокую производительность, что позволяет эффективно обрабатывать большие объемы данных и принимать решения на основе этих данных в кратчайшие сроки.
Для бизнеса базы данных предоставляют возможности для автоматизации ключевых процессов, таких как учет товаров, расчет зарплаты сотрудников, обработка финансовых транзакций и управление взаимоотношениями с клиентами. В государственных учреждениях базы данных используются для организации информации о гражданах, управления социальными программами, здравоохранением и другими государственными услугами.
Одним из важных аспектов работы с базами данных является обеспечение безопасности данных. Современные системы баз данных предлагают надежные механизмы защиты, такие как шифрование данных, управление доступом и регулярное резервное копирование, что помогает предотвратить утечку информации и защитить данные от несанкционированного доступа.
Таким образом, базы данных являются основой для управления данными в разных сферах деятельности и играют ключевую роль в современном обществе.

1.2. Модели данных и выбор модели для проекта

Для эффективной организации данных существует несколько моделей, каждая из которых обладает уникальными характеристиками и применяется в зависимости от конкретных требований системы. Рассмотрим основные модели данных, используемые для разработки и управления базами данных:

1. **Иерархическая модель**

   Иерархическая модель данных была одной из первых моделей для организации информации. В этой модели данные структурированы в виде дерева, где каждая запись имеет один родительский элемент. Это ограничивает гибкость модели, поскольку каждый элемент может быть связан только с одним родителем. Например, информацию о сотрудниках можно организовать в виде дерева, где каждый сотрудник относится к более высокому уровню подразделения.

Недостатки этой модели включают сложность в реализации и ограниченную гибкость, так как данные могут быть связаны лишь в одном направлении. Это делает систему не слишком удобной для динамичных процессов, где связи между элементами могут изменяться или быть более сложными.

2. **Сетевой модель**.

   В сетевой модели данные организованы в виде графа, где каждый элемент может иметь несколько родительских элементов, что увеличивает гибкость по сравнению с иерархической моделью. Например, один товар может быть связан с несколькими поставщиками. Эта модель предоставляет возможность более точно описывать сложные взаимосвязи данных, что делает её более подходящей для ситуаций, где требуется представление множества связей между элементами.

Недостатки сетевой модели включают её сложность в проектировании и эксплуатации, несмотря на большую гибкость по сравнению с иерархической моделью. Кроме того, она требует более сложных механизмов для обработки запросов, что может затруднить её использование в некоторых случаях.

3. **Реляционная модель**

Реляционная модель является наиболее популярной и широко используемой в современных системах управления базами данных.

Данные в этой модели хранятся в виде таблиц, каждая из которых состоит из строк и столбцов. Таблицы могут быть связаны между собой с помощью ключей. Реляционная модель поддерживает использование языка SQL, который является стандартом для работы с реляционными базами данных.

Недостатки: сложность при работе с неструктурированными данными, снижение производительности при работе с большими объемами данных, сложности с масштабированием, проблемы при изменении структуры данных.

4. **Объектно-ориентированная модель**.
   В объектно-ориентированной модели данные представляют собой объекты, которые могут содержать как данные, так и методы для их обработки. Эта модель используется для хранения более сложных структур данных, таких как изображения, видео или другие мультимедийные данные. Объектно-ориентированные базы данных обеспечивают возможность работы с данными как с объектами, что делает их подходящими для более сложных систем.

Недостатки: требует значительных вычислительных ресурсов и может быть сложной для реализации в тех случаях, когда данных не требуется хранить в виде объектов.

Преимущества реляционной модели

Реляционная модель данных обладает несколькими важными преимуществами, которые делают её наиболее популярной для большинства современных информационных систем:

- **Простота**: реляционная модель основана на использовании таблиц, что делает её легко понимаемой и применимой в различных сферах. Каждая таблица соответствует одной сущности, например, товары, клиенты, продажи и т.д.
- **Гибкость**: таблицы можно изменять и расширять, добавлять новые атрибуты, не затрудняя работы всей базы данных. Также можно добавлять новые таблицы и устанавливать связи между ними, что обеспечивает гибкость в расширении системы.
- **Поддержка SQL**: реляционная модель поддерживает использование SQL — языка для работы с базами данных. SQL

позволяет легко извлекать данные, делать выборки, обновлять и удалять информацию. Это существенно упрощает работу с данными и позволяет быстро создавать сложные запросы для извлечения нужной информации.

- **Нормализация**: реляционная модель поддерживает процесс нормализации, который помогает уменьшить избыточность данных и улучшить их целостность. Например, данные о товарах и поставщиках могут быть размещены в отдельных таблицах, с ссылками друг на друга через внешние ключи, что упрощает обновление и управление этими данными.
- **Поддержка транзакций**: реляционные базы данных поддерживают транзакции, что гарантирует целостность данных при выполнении нескольких операций, таких как добавление записей, их обновление или удаление. Это особенно важно для приложений, где данные часто изменяются, и необходимо избежать ошибок в процессе обработки.

Реляционная модель идеально подходит для овощного магазина по следующим причинам:

Удобство учета товаров: в овощном магазине важно вести учет различных видов продукции, их характеристик. Эти данные легко организуются в таблицы с атрибутами для каждого товара. Например, таблица "Товары" может содержать такие атрибуты, как название, цена, количество, вес, категория, поставщик и т.д.

Учёт поставок и продаж: связь между поставками и продажами может быть организована с помощью внешних ключей. Например, таблица "Поставки" может содержать информацию о поступивших товарах, а таблица "Продажи" — об их реализации. Используя реляционные связи, можно легко отследить, какие товары были проданы и какие поставки их пополнили.

Гибкость и масштабируемость: реляционная модель позволяет легко добавлять новые атрибуты или таблицы. Например, можно расширить систему, добавив учет клиентов или сотрудников магазина. Для этого достаточно добавить новые таблицы и установить связи между ними с помощью внешних ключей.

Использование SQL для анализа данныхРеляционная модель данных позволяет использовать SQL для извлечения необходимой информации, что облегчает создание отчетов по продажам, товарным

остаткам, активности клиентов и других важных аспектах. Например, с помощью SQL можно легко запросить все товары, поставленные определенным поставщиком, или вычислить общий объем продаж за конкретный период времени.

Целостность данных: в реляционной модели данные организованы так, что ошибки при их обработке сводятся к минимуму. Например, нельзя случайно добавить запись о товаре без указания его цены или количества. Нормализация базы данных также помогает избежать дублирования данных, что улучшает качество работы с базой.

1.3. Основы проектирования реляционных баз данных

Проектирование реляционной базы данных включает несколько ключевых этапов, которые обеспечивают эффективное и структурированное хранение данных, поддержание целостности и минимизацию избыточности.

Перед тем как начать проектирование базы данных, необходимо провести тщательный анализ бизнес-процессов, которые эта база данных будет поддерживать. В случае с овощным магазином это может включать в себя следующие ключевые процессы:

1. **Закупка товара**: Как товары поступают в магазин, какие поставщики их предоставляют, какие данные необходимо хранить о каждом.
2. **Хранение товаров**: Учет остатков на складе, условия хранения, сроки годности и прочее.
3. **Продажа товаров**: Система учёта продаж, взаимодействие с клиентами, обработка заказов, расчет стоимости и создание накладных.
4. **Работа с клиентами**: Учет информации о клиентах, их покупках и предпочтениях, скидки и лояльность.
5. **Поставщики**: Информация о поставщиках товаров, их контактные данные, история поставок, условия сотрудничества.

После анализа всех бизнес-процессов можно определить, какие данные необходимо хранить и какие связи между этими данными будут существовать. Например, необходимо хранить таблицы для товаров, поставок, продаж, клиентов, сотрудников и т.д., а также определить, как эти таблицы будут связаны.

После сбора требований начинается создание ER-диаграмм, которые представляют собой визуальное отображение структуры базы данных. ER-диаграммы позволяют наглядно увидеть, какие сущности существуют в системе и как они связаны друг с другом.

1. **Сущности**: сущности — это основные объекты, данные о которых будут храниться в базе данных. В овощном магазине это могут быть такие сущности, как Продукты, Поставщики, Продажи, Клиенты и другие.
2. **Атрибуты**: атрибуты — это характеристики сущностей, которые будут храниться в базе данных. Например, для сущности Продукты атрибутами могут быть наименование, цена, количество на складе, срок годности и т.д.
3. **Связи**: связи определяют, как сущности взаимодействуют друг с другом. Например, связь между Продуктами и Поставщиками может быть представлена через внешние ключи, где каждый товар будет связан с поставщиком, который его поставляет.

После определения структуры данных необходимо провести нормализацию базы данных. Нормализация представляет собой процесс устранения избыточности и повышения целостности данных, что способствует снижению повторения информации и улучшению управления данными. Этот процесс включает несколько этапов, называемых нормальными формами.

1. **1НФ:**.
   К
   .
   аж
   .
   до
   .
   е поле в таблице должно содержать только одно значение. Например, если в одной строке таблицы указаны несколько поставщиков для одного товара, это нарушает 1НФ, и нужно разделить данные на несколько строк.
2. **2НФ**:.
   Д
   .
   ля выполнения 2НФ необходимо, чтобы каждая неключевая

колонка в таблице была полностью функционально зависима от первичного ключа

3. **3НФ**:

Д

ля выполнения 3НФ данные не должны содержать транзитивные зависимости, то есть если один атрибут зависит от другого, то зависимость должна быть только через ключевую сущность. Это минимизирует дублирование данных и улучшает целостность.

Нормализация данных помогает избежать дублирования информации, повышает удобство работы с данными, уменьшает объем занимаемой памяти и ускоряет выполнение запросов в базе данных. Но при этом важно найти баланс: избыточность в данных может быть оправдана, если это способствует производительности системы, например, для ускорения выборок в отчетах.

1.4. Язык SQL и его роль в управлении базами данных

SQL (Structured Query Language) является универсальным языком для работы с реляционными базами данных. Он используется для проектирования и управления базами данных, а также для выполнения различных операций с данными, таких как создание таблиц, установление связей между таблицами, извлечение, обновление и удаление данных.

SQL был разработан в 1970-х годах в IBM и с тех пор стал международным стандартом для работы с реляционными базами данных. Он используется в большинстве современных СУБД (систем управления базами данных), таких как MySQL, PostgreSQL, Oracle, Microsoft SQL Server и других.

SQL предоставляет мощные инструменты для:

- Создания и изменения структуры базы данных.
- Извлечения данных из базы с помощью запросов.
- Вставки, обновления и удаления данных.
- Управления доступом к данным.

SQL представляет собой декларативный язык, что означает, что пользователь формулирует, какие операции нужно выполнить с данными, а не как именно они должны быть выполнены. Этот подход помогает избежать сложных алгоритмов и сосредоточиться на достижении нужного результата.
SQL включает несколько типов команд, которые выполняют различные операции с базой данных. Основные категории команд следующие:
DDL (Data Definition Language) — команды определения данных:

- **CREATE**: используется для создания таблиц, индексов, представлений и других объектов базы данных.
- **ALTER**: используется для изменения структуры уже существующих объектов базы данных.
- **DROP**: используется для удаления объектов базы данных, таких как таблицы или индексы.

Пример команды для создания таблицы:

```sql
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    name VARCHAR(255),
    price DECIMAL(10, 2),
    stock_quantity INT
);
```

DML (Data Manipulation Language) — команды манипуляции данными:

- **SELECT**: используется для извлечения данных из базы данных.
- **INSERT**: используется для добавления новых данных в таблицу.
- **UPDATE**: используется для изменения существующих данных в таблице.
- **DELETE**: используется для удаления данных из таблицы.

Пример команды для вставки нового товара:

```sql
INSERT INTO Products (product_id, name, price, stock_quantity)
VALUES (1, 'Carrot', 0.99, 150);
```

DCL (Data Control Language) — команды управления доступом:

- **GRANT**: используется для предоставления прав доступа к объектам базы данных.
- **REVOKE**: используется для отзыва прав доступа.

Пример команды для предоставления прав:

```sql
GRANT SELECT, INSERT ON Products TO user_name;
```

В рамках овощного магазина SQL позволяет эффективно управлять данными о товарах, продажах, клиентах и поставках. Приведем несколько типичных запросов, которые могут быть полезны для такого бизнеса:

Поиск товаров по цене

Запрос для поиска всех овощей в магазине, которые стоят менее 200.00:

```sql
SELECT name, price, stock_quantity
FROM Products
WHERE price < 200.00;
```

Этот запрос поможет продавцам или менеджерам быстро находить дешевые товары, которые могут быть популярными среди покупателей, а также управлять остатками этих товаров на складе.

Подсчет общих продаж по товарам

Запрос для подсчета общего объема продаж каждого товара:

```sql
SELECT p.name, SUM(s.quantity) AS total_sales
FROM Sales s
JOIN Products p ON s.product_id = p.product_id
GROUP BY p.name
ORDER BY total_sales DESC;
```

Этот запрос позволяет определить, какие товары продаются лучше всего, что важно для планирования закупок и управления ассортиментом.

1.5. Обзор средств разработки и управления базами данных

Современные инструменты для разработки и управления базами данных предоставляют все необходимые функции для создания, администрирования и оптимизации работы с базами данных. Эти инструменты предлагают интуитивно понятные интерфейсы для проектирования баз, обработки информации и обеспечения безопасности данных. Рассмотрим один из самых популярных инструментов — систему управления базами данных MySQL и инструменты, используемые для работы с ней.

MySQL — это широко используемая система управления базами данных с открытым исходным кодом, которая находит применение как в малых веб-приложениях, так и в крупных корпоративных системах. MySQL использует реляционную модель данных и SQL для работы с информацией.

Возможности MySQL:

- **Реляционная модель данных:** MySQL позволяет создавать и управлять таблицами, индексами и связями между таблицами, а также обеспечивает нормализацию данных.
- **Поддержка транзакций:** MySQL поддерживает транзакции, что обеспечивает атомарность операций, позволяя использовать механизмы отката и сохранения данных.
- **Масштабируемость и высокая производительность:** Система оптимизирована для обработки больших объемов данных и гарантирует высокую производительность при выполнении запросов.
- **Поддержка нескольких форматов хранения:** MySQL предлагает различные движки хранения данных, такие как InnoDB и MyISAM, что дает возможность выбрать наиболее подходящий в зависимости от специфики задачи.
- **Репликация:** MySQL поддерживает репликацию данных, что позволяет создавать резервные копии базы данных и эффективно распределять нагрузку между серверами.
- **Безопасность:** MySQL включает в себя различные механизмы аутентификации, шифрования данных и управления правами доступа, что помогает обеспечить высокий уровень безопасности данных.

Преимущества MySQL:

- **Открытый исходный код:** MySQL является бесплатной системой управления базами данных с открытым исходным кодом, что позволяет свободно использовать её для разработки и внедрения в различные приложения.
- **Широкая совместимость:** MySQL поддерживает большинство операционных систем, включая Linux, Windows и macOS, что делает её универсальным решением для различных платформ.
- **Активное сообщество:** Система имеет большое сообщество разработчиков и пользователей, что упрощает поиск решений и доступ к документации для настройки и оптимизации.
- **Высокая производительность:** MySQL разработан с акцентом на быструю обработку запросов, что делает его отличным выбором для приложений с высокой нагрузкой.

Применение MySQL:

MySQL активно используется для разработки и управления базами данных в различных сферах, таких как::

- **Веб-разработка:** является основой для большинства современных веб-приложений и систем управления контентом (CMS).
- **Бизнес-аналитика:** MySQL помогает эффективно организовывать учет и анализ данных, связанных с продажами, запасами, клиентами и другими ключевыми аспектами бизнеса.
- **Интернет-магазины**: используется для хранения информации о товарах, клиентах, заказах и финансовых транзакциях.
- **Образовательные учреждения:** применяется для управления данными студентов, преподавателей, курсов и результатов их обучения.

Для эффективной работы с базами данных разработчики и администраторы используют различные инструменты, которые упрощают процессы создания, управления и анализа данных.

1. **MySQL Workbench**
   MySQL Workbench — это интегрированная среда для работы с MySQL, которая предоставляет инструменты для проектирования, администрирования и разработки баз данных, а также для анализа и оптимизации SQL-запросов.

Основные возможности MySQL Workbench:

- **Визуальное проектирование базы данных:** MySQL Workbench позволяет создавать ER-диаграммы, проектировать таблицы и устанавливать связи между ними с помощью ключей и внешних связей.
- **Управление базами данных:** Инструмент включает функции для создания, удаления баз данных, а также добавления и изменения таблиц, управления индексами и ограничениями.
- **Оптимизация запросов:** MySQL Workbench помогает анализировать выполнение SQL-запросов и выявлять узкие места в производительности базы данных.
- **Управление пользователями и правами доступа:** Инструмент предоставляет возможности для управления правами доступа пользователей к базе данных.

MySQL Workbench — это мощный инструмент для разработчиков, который помогает проектировать базы данных, управлять ими и

анализировать SQL-запросы. Он идеально подходит для работы с комплексными системами и удобен при обработке больших объемов данных.

Резервное копирование и восстановление данных являются важными элементами администрирования баз данных, обеспечивая защиту информации от потерь, сбоев и ошибок.

1. **Резервное копирование:**
   Резервное копирование базы данных представляет собой процесс создания её копии в безопасном месте, чтобы в случае сбоя можно было восстановить данные. В MySQL для этого доступны несколько методов:

- **mysqldump:** Утилита командной строки для создания дампов базы данных в формате SQL. Этот метод создает текстовый файл с SQL-командами, которые можно использовать для восстановления данных.
- **Репликация:** Репликация позволяет создавать резервные копии базы данных в реальном времени, синхронизируя основной сервер с резервным.
- **Инструменты третьих сторон:** Для более сложных и автоматизированных процессов резервного копирования можно использовать такие программы, как Percona XtraBackup, которые позволяют выполнять горячее резервное копирование.

2. **Восстановление данных**:
   Восстановление данных после сбоя или потери данных выполняется с использованием резервных копий. В MySQL для восстановления данных применяется следующая команда:
   mysql -u username -p database_name < backup_file.sql

Глава 2. Практическая реализация баз данных овощного магазина

2.1. Анализ предметной области и постановка задачи

Описание деятельности магазина: закупка, хранение, продажа овощей

Овощной магазин специализируется на продаже овощей, что требует четкого и своевременного учета всех бизнес-операций — начиная с закупки товаров и заканчивая их продажей конечным потребителям. Для обеспечения высокого качества обслуживания и минимизации ошибок в учете крайне важно наладить эффективное управление

данными. Рассмотрим ключевые процессы, которые поддерживает система учета в таком магазине:

1. **Закупка овощей**:
   Закупка — это процесс получения товаров от поставщиков. Важным аспектом является наличие подробной информации о характеристиках каждого товара и его поставщике. Также необходимо учитывать количество товара, поступившее в магазин, чтобы своевременно обновлять запасы и избегать дефицита.
2. **Хранение овощей**:
   После поступления товаров в магазин необходимо организовать их хранение на складе. Это включает в себя учет всех товаров, их текущие остатки на складе, срок годности, условия хранения и другие параметры, которые могут повлиять на качество товара.
3. **Продажа овощей**:
   Продажа — это процесс передачи товара клиенту за оплату. Важно отслеживать, какие товары были проданы, количество проданных единиц, общую стоимость покупки и информацию о покупателе. Необходимо учитывать возможные скидки и акции, которые могут влиять на цену товара.

Необходимые сущности и связи для учета

Для автоматизации всех процессов учета в овощном магазине необходимо создать несколько сущностей в базе данных. Эти сущности будут хранить важную информацию о товарах, клиентах, сотрудниках и продажах. Основные сущности и их связи будут следующими:

1. **Продукты**:
   Сущность "Продукты" будет содержать данные о каждом товаре, который имеется в магазине. Эта сущность включает в себя информацию о наименовании товара, его категории, цене за килограмм, текущем количестве на складе и поставщике, от которого поступил товар.
   - Атрибуты: product_id, product_name, category, price_per_kg, stock_kg, supplier_id.
2. **Поставщики**:

Сущность "Поставщики" будет хранить информацию о компаниях или физических лицах, которые осуществляют поставку товаров в магазин. В этой сущности будет содержаться важная информация, такая как контактные данные поставщиков, их наименование, а также адрес, что необходимо для эффективного управления поставками.

- Атрибуты: supplier_id, supplier_name, phone, email, address.

3. **Клиенты**:

   Сущность **"Клиенты"** будет хранить информацию о покупателях магазина. Важно собирать данные о клиентах, их покупках и предпочтениях, так как это поможет в дальнейшем предложить персонализированные предложения и разработать программы лояльности, которые могут увеличить продажи и улучшить клиентский сервис.

   - Атрибуты: customer_id, full_name, phone, email, registration_date.

4. **Продажи**:

   Сущность "Продажи" будет отслеживать информацию о каждой совершенной продаже, включая данные о товаре, клиенте, количестве проданных единиц и общей стоимости. Каждая продажа будет связана с конкретным товаром и клиентом, что позволяет эффективно отслеживать все операции по реализации товаров.

   - Атрибуты: sale_id, product_id, customer_id, sale_date, quantity_kg, total_price.

5. **Сотрудники**:

   Сущность "Сотрудники" будет хранить информацию о работниках магазина, включая данные о их должностях, контактной информации и датах трудоустройства. Эти данные

необходимы для учета работы сотрудников, а также для анализа их взаимодействия с клиентами, что позволяет эффективно управлять персоналом и контролировать рабочие процессы.

- Атрибуты: employee_id, full_name, position, hire_date, phone, email.

Связи между сущностями:

- Продукты и Поставщики связаны через внешний ключ supplier_id, так как каждый товар поступает от одного поставщика. Это позволяет отслеживать, какие товары были поставлены каким поставщиком, обеспечивая точность учета и планирование закупок.
- Продажи связаны с Продуктами через внешний ключ product_id, так как каждая продажа включает в себя конкретный товар. Это позволяет связать данные о продажах с конкретными товарами, облегчая анализ продаж по каждому продукту.
- Продажи также связаны с Клиентами через внешний ключ customer_id, так как каждая продажа принадлежит определенному клиенту. Это позволяет отслеживать, какие товары покупают конкретные клиенты, и анализировать их покупательские предпочтения.
- Сотрудники не связаны напрямую с другими сущностями, но их данные могут быть полезны для учета их действий, например, для отслеживания продавцов, которые совершали продажи. Это может быть полезным для анализа производительности сотрудников, а также для контроля за качеством обслуживания клиентов.

Требования к функционалу базы данных

База данных для овощного магазина должна поддерживать следующие функциональные требования:

1. **Учет товаров**:

   База данных должна обеспечивать хранение всей необходимой информации о товарах, включая их характеристики. Необходимо предоставить возможность добавления, редактирования и удаления данных о товарах, а также получать актуальные сведения о запасах на складе. Это обеспечит точный контроль за состоянием товаров и позволит оперативно обновлять данные о товарных остатках.

2. **Учет поставок**:

   Система должна эффективно отслеживать поступление товаров от поставщиков, включая данные о дате поступления, количестве, цене и поставщике. Важно, чтобы база данных поддерживала связь между товарами и поставщиками, что позволит точно отслеживать, какие товары поступили от каждого поставщика, и проводить анализ закупок.

3. **Продажа товаров**:

   База данных должна поддерживать процесс продажи товаров, включая информацию о клиенте, количестве проданных товаров, общей стоимости покупки и дате продажи. Система должна также учитывать возможные скидки и акции, и автоматически обновлять товарные остатки после каждой продажи. Это поможет отслеживать текущие запасы и облегчить процесс учета проданных товаров.

4. **Учет клиентов**:

   Необходимо хранить информацию о клиентах, включая их контактные данные и историю покупок. Это позволит магазинам предложить персонализированные скидки и уведомления о

специальных предложениях, а также анализировать покупательские предпочтения для улучшения маркетинга и клиентского обслуживания.

5. **Учет сотрудников**:

   База данных должна включать данные о сотрудниках магазина, включая их должности, контактные данные и дату найма. Это обеспечит возможность отслеживать, кто был ответственным за выполнение определенных операций, таких как продажа товара или прием поставки, и поможет управлять персоналом.

6. **Обработка отчетности**:

   Одной из ключевых функций системы является анализ данных, таких как общий объем продаж, прибыль, товарные остатки и популярность продуктов. База данных должна поддерживать создание отчетов, которые помогут менеджерам принимать более обоснованные и эффективные решения на основе актуальных данных.

7. **Масштабируемость**:

   Система должна быть спроектирована таким образом, чтобы при росте бизнеса и увеличении объемов данных база данных могла справляться с дополнительной нагрузкой, не снижая производительность. Это обеспечит ее долговечность и удобство использования на протяжении времени.

8. **Автоматизация процессов**:

   Важной функцией базы данных является поддержка автоматизации рутинных операций, таких как обновление остатков товаров после продажи, расчет общей стоимости заказа и других задач. Это повысит эффективность работы магазина, сократит время на выполнение операций и снизит вероятность ошибок, связанных с человеческим фактором.

2.2. Реализация базы данных

В данной части главы будет представлен полный код для создания таблиц базы данных, примеры вставки тестовых данных, а также объяснение использования типа данных DECIMAL для точного учета веса и цен.

Основываясь на проектировании структуры базы данных для овощного магазина, будут созданы несколько таблиц: Products, Suppliers, Sales, Customers и Employees. Эти таблицы обеспечат хранение информации о товарах, поставках, продажах, клиентах и сотрудниках, что позволит эффективно управлять всеми процессами, связанными с деятельностью магазина

.

1. Создание таблицы поставщиков

```sql
CREATE TABLE Suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address TEXT
);
```

2. Создание таблицы продуктов

```sql
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price_per_kg DECIMAL(10, 2),
    stock_kg DECIMAL(10, 2),
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
);
```

3. Создание таблицы клиентов

```sql
CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    registration_date DATE
);
```

4. Создание таблицы сотрудников

```sql
CREATE TABLE Employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(255),
    position VARCHAR(100),
    hire_date DATE,
    phone VARCHAR(20),
    email VARCHAR(100)
);
```

5. Создание таблицы продаж

```sql
CREATE TABLE Sales (
    sale_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT,
    customer_id INT,
    sale_date DATE,
    quantity_kg DECIMAL(10, 2),
    total_price DECIMAL(10, 2),
    FOREIGN KEY (product_id) REFERENCES Products(product_id),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

ChatGPT сказал:

Этот код создает таблицы для хранения данных о поставщиках, товарах, клиентах, сотрудниках и продажах. Внешние ключи используются для связывания данных между таблицами, например, между Products и Suppliers, Sales и Products, а также Sales и Customers

Для того чтобы проверить, как система работает с реальными данными, необходимо вставить несколько тестовых записей в таблицы. Это поможет убедиться, что связи между таблицами корректно установлены, а база данных функционирует должным образом. Вставка

данных позволит проверить корректность выполнения запросов и целостность данных, а также убедиться в работоспособности системы в реальных условиях.

Примеры вставки данных:

1. Вставка данных в таблицу поставщиков

```
INSERT INTO Suppliers (supplier_name, phone, email, address)
VALUES
    ('ООО "АгроПоставка"', '+79990001122', 'agro@mail.ru', 'г. Москва, ул. Полевая, д. 1'),
    ('Фермер Иванов', '+79995556677', 'ivanovfarm@yandex.ru', 'МО, д. Ивановка, ул. Садовая, 3');
```

2. Вставка данных в таблицу продуктов

```
INSERT INTO Products (product_name, category, price_per_kg, stock_kg, supplier_id)
VALUES
    ('Картофель', 'Корнеплод', 30.00, 500.00, 1),
    ('Морковь', 'Корнеплод', 25.00, 300.00, 1),
    ('Помидор', 'Плодовый', 50.00, 200.00, 2),
    ('Огурец', 'Плодовый', 45.00, 250.00, 2),
    ('Капуста', 'Листовой', 20.00, 400.00, 1);
```

3. Вставка данных в таблицу клиентов

```
INSERT INTO Customers (full_name, phone, email, registration_date)
VALUES
    ('Петров Иван', '+79991234567', 'ivan.petrov@mail.ru', '2024-05-01'),
    ('Сидорова Анна', '+79997654321', 'anna_sidorova@mail.ru', '2024-05-05'),
    ('Иван Иванов', '+7 999 123 45 67', 'ivan.ivanov@example.com', '2025-06-20'),
    ('Колокушкин Иван', '+79304567612', 'dadsa@gmail.com', '2025-06-20');
```

4. Вставка данных в таблицу сотрудников

```
INSERT INTO Employees (full_name, position, hire_date, phone, email)
VALUES
    ('Иванов Иван Иванович', 'Продавец', '2022-03-15', '+79990001111', 'ivanov@example.com'),
    ('Петрова Анна Сергеевна', 'Продавец', '2023-01-20', '+79990002222', 'petrova@example.com'),
    ('Сидоров Павел Дмитриевич', 'Кладовщик', '2021-11-05', '+79990003333', 'sidorov@example.com'),
    ('Кузьмина Елена Викторовна', 'Администратор', '2020-06-12', '+79990004444', 'kuzmina@example.com');
```

5. Вставка данных в таблицу продаж

```sql
INSERT INTO Sales (product_id, customer_id, sale_date, quantity_kg, total_price)
VALUES
    (3, 3, '2025-06-20', 10.50, 525.00),
    (5, 1, '2025-06-20', 8.01, 159.52),
    (4, 1, '2025-06-20', 4.85, 418.91),
    (3, 1, '2025-06-20', 1.00, 304.61),
    (2, 1, '2025-06-20', 6.45, 74.21),
    (1, 1, '2025-06-20', 4.50, 77.10),
    (5, 2, '2025-06-20', 8.36, 62.25),
    (4, 2, '2025-06-20', 9.47, 315.48),
    (3, 2, '2025-06-20', 5.64, 309.36),
    (2, 2, '2025-06-20', 3.00, 136.54),
    (1, 2, '2025-06-20', 7.29, 272.67);
```

Эти примеры вставки данных помогут протестировать работу базы данных, проверив корректность установленных связей, а также удостовериться в том, что все таблицы содержат нужную информацию. Вставка тестовых данных позволит увидеть, как система работает с реальными значениями и поможет выявить возможные ошибки или недочеты в структуре базы данных.

Объяснение особенностей типа данных DECIMAL для учета веса и цен

Ключевым аспектом работы с базой данных является правильное использование типа **DECIMAL** для хранения информации о цене за килограмм и количестве товара. Тип **DECIMAL** применяется для хранения чисел с фиксированной запятой, что особенно важно при работе с денежными суммами и точными расчетами, где важна каждая десятичная доля.

- **DECIMAL(10, 2):** этот тип данных означает, что число может содержать до 10 знаков, из которых 2 знака находятся после запятой. Это идеальный выбор для хранения цен и весов товаров, так как обеспечивает необходимую точность в расчетах.
- **Точность:** Использование типа данных DECIMAL позволяет гарантировать точность данных при расчетах, что критично при

вычислении общей стоимости товаров, а также при учете остатков на складе. Такой подход исключает возможные ошибки округления, которые могут возникнуть при использовании типов данных с плавающей запятой, обеспечивая более точные и надежные результаты.

Это гарантирует сохранение точности при обработке больших объемов данных, таких как стоимость товара или количество на складе, исключая ошибки округления, которые могут возникнуть при использовании типов данных с плавающей запятой. Использование типа DECIMAL помогает избежать неточностей в расчетах, обеспечивая более надежное и точное представление чисел, что особенно важно в финансовых расчетах и учете товарных остатков.

2.3. Реализация основных SQL-запросов

В этой части главы будут рассмотрены типовые SQL-запросы, которые обеспечивают функционал для формирования отчетов по продажам, товарным остаткам на складе, активности клиентов, обновлению данных и удалению устаревших записей. Эти запросы являются основными для работы с базой данных овощного магазина и помогают эффективно управлять данными, а также получать нужные отчеты.

1. Получение общего объема продаж за последний месяц с деталями о клиентах и продуктах

В данной части главы будут рассмотрены типовые SQL-запросы, которые обеспечивают функциональность для формирования отчетов по продажам, товарным остаткам на складе, активности клиентов, а также для обновления данных и удаления устаревших записей. Эти запросы являются основными инструментами для работы с базой данных овощного магазина, способствуя эффективному управлению данными и предоставлению необходимых отчетов для принятия обоснованных решений.

```sql
SELECT
    s.sale_id,
    c.full_name AS customer_name,
    p.product_name,
    s.quantity_kg,
    s.total_price,
    s.sale_date
FROM
    Sales s
JOIN
    Customers c ON s.customer_id = c.customer_id
JOIN
    Products p ON s.product_id = p.product_id
WHERE
    s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
ORDER BY s.sale_date DESC;
```

В этой части главы будут приведены примеры типовых SQL-запросов, которые обеспечивают функциональность для формирования отчетов по продажам, товарным остаткам на складе, активности клиентов, а также для обновления данных и удаления устаревших записей. Эти запросы являются основными инструментами для эффективной работы с базой данных овощного магазина, способствуя оптимизации управления данными и предоставлению отчетов, которые помогут принимать обоснованные управленческие решения.

2. Получение статистики по продажам продуктов

Запрос для получения общего количества проданных товаров по категориям и общей суммы выручки:

```sql
SELECT
    p.category,
    SUM(s.quantity_kg) AS total_quantity,
    SUM(s.total_price) AS total_revenue
FROM
    Sales s
JOIN
    Products p ON s.product_id = p.product_id
GROUP BY
    p.category
ORDER BY
    total_revenue DESC;
```

Этот запрос группирует данные о продажах по категориям продуктов и вычисляет общее количество проданных товаров и сумму выручки по каждой категории. Это полезно для анализа, какие категории товаров пользуются наибольшим спросом и приносят наибольшую прибыль, что помогает оптимизировать ассортимент и стратегию продаж.

3. Получение информации о товарных остатках на складе

Запрос для получения всех товаров, которые имеют остатки на складе меньше 10 кг, что позволяет отследить товары с низкими остатками.

```sql
SELECT
    product_name,
    stock_kg
FROM
    Products
WHERE
    stock_kg < 10;
```

Этот запрос позволяет выявить товары с низким остатком на складе, которые необходимо заказать снова, так как их количество слишком маленькое. Это помогает предотвратить дефицит товаров и обеспечивать их наличие на складе для удовлетворения спроса клиентов.

4. Получение активности клиентов

Запрос для получения списка клиентов, которые совершили покупки в последние 30 дней:

```sql
SELECT
    c.full_name AS customer_name,
    COUNT(s.sale_id) AS total_purchases,
    SUM(s.total_price) AS total_spent
FROM
    Sales s
JOIN
    Customers c ON s.customer_id = c.customer_id
WHERE
    s.sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY
    c.customer_id
ORDER BY
    total_spent DESC;
```

Этот запрос помогает определить, какие клиенты проявляют наибольшую активность в последнее время, сколько покупок они

совершили и какую сумму потратили. Эти данные могут быть полезны для разработки программы лояльности или отправки персонализированных предложений, направленных на повышение удовлетворенности клиентов и стимулирование повторных покупок.

Запросы для обновления данных

1. Изменение цен на продукты

Запрос для обновления цены на определенный товар

```
UPDATE Products
SET price_per_kg = price_per_kg * 1.10
WHERE product_name = 'Картофель';
```

Этот запрос увеличивает цену на товар "Картофель" на 10%. Это может быть полезно в случае изменения цен у поставщика или для корректировки цен в зависимости от рыночной ситуации, чтобы поддерживать прибыльность магазина и адаптироваться к изменениям на рынке.

2. Обновление количества товара на складе

Запрос для обновления количества товара на складе после поставки:

```
UPDATE Products
SET stock_kg = stock_kg + 100
WHERE product_id = 1;
```

Этот запрос увеличивает количество товара на складе на 100 кг для продукта с **product_id = 1**. Это может быть полезно для учета поступлений товаров и пополнения запасов на складе, обеспечивая точность данных о наличии товара и предотвращая дефицит.

3. Изменение данных клиента

Запрос для обновления данных клиента

```
UPDATE Customers
SET phone = '+79995554433'
WHERE customer_id = 3;
```

Этот запрос обновляет контактные данные клиента, что важно для поддержания актуальности информации о клиентах.

2.4. Создание хранимых процедур, триггеров и представлений

Для автоматизации бизнес-процессов и улучшения работы с данными в базе данных используются такие механизмы, как хранимые процедуры, триггеры и представления. Эти инструменты способствуют автоматизации повторяющихся операций, обеспечению целостности данных и улучшению удобства работы с часто используемой информацией.

Обоснование применения хранимых процедур для автоматизации повторяющихся операций:
Хранимые процедуры представляют собой наборы SQL-запросов, которые сохраняются в базе данных и могут быть выполнены по запросу. Они позволяют автоматизировать рутинные операции и повышают общую производительность системы. Хранимые процедуры особенно полезны в случаях, когда одно и то же действие необходимо выполнять многократно..
В случае с овощным магазином хранимые процедуры могут быть использованы для выполнения таких операций, как:

- Оформление продажи — процедура автоматизирует процесс внесения данных о продаже, обновления остатков товара и фиксации транзакции.
- Добавление нового клиента в базу данных — процедура упрощает процесс регистрации нового покупателя в системе, автоматически добавляя его данные.

Преимущества хранимых процедур:

- Упрощение повторяющихся операций**:** Хранимые процедуры позволяют выполнять сложные операции, включающие несколько SQL-запросов, за один раз, что делает их удобными для выполнения стандартных задач.
- Снижение вероятности ошибок**:** Поскольку операции выполняются централизованно и стандартизированно, вероятность ошибок при вводе данных и их обработке значительно снижается.
- **Повышение производительности:** Хранимые процедуры выполняются на стороне сервера, что уменьшает нагрузку на клиентские приложения и ускоряет выполнение операций, сокращая время отклика системы.

Пример хранимой процедуры для оформления продажи:

```sql
CREATE PROCEDURE CreateSale(
    IN in_product_id INT,
    IN in_customer_id INT,
    IN in_quantity DECIMAL(10,2)
)
BEGIN
    DECLARE product_price DECIMAL(10,2);
    DECLARE stock_available DECIMAL(10,2);

    -- Получаем цену товара и количество на складе
    SELECT price_per_kg, stock_kg INTO product_price, stock_available
    FROM Products WHERE product_id = in_product_id;

    -- Проверяем наличие товара на складе
    IF stock_available < in_quantity THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Недостаточно товара на складе';
    ELSE
        -- Добавляем запись о продаже в таблицу Sales
        INSERT INTO Sales (product_id, customer_id, sale_date, quantity_kg, total_price)
        VALUES (in_product_id, in_customer_id, CURDATE(), in_quantity, in_quantity * product_price);

        -- Обновляем количество товара на складе
        UPDATE Products SET stock_kg = stock_kg - in_quantity WHERE product_id = in_product_id;
    END IF;
END;
```

- Процедура CreateSale принимает следующие параметры: идентификаторы товара и клиента, а также количество товара, которое необходимо продать.
- Процедура сначала проверяет, достаточно ли товара на складе для выполнения продажи. Если на складе имеется достаточное количество товара, то происходит вставка записи о продаже в таблицу Sales. Также обновляется остаток товара на складе, уменьшая количество на соответствующем складе.

Для добавления нового клиента в базу данных можно создать специализированную хранимую процедуру. Вместо того чтобы каждый раз вручную добавлять клиента через SQL-запросы, эта процедура будет автоматически выполнять операцию вставки данных в таблицу **Customers**..
П

.
ри

ме

р хранимой процедуры для добавления нового клиента:

```sql
CREATE PROCEDURE AddCustomer(
    IN in_full_name VARCHAR(100),
    IN in_phone VARCHAR(20),
    IN in_email VARCHAR(100)
)
BEGIN
    -- Вставка данных о новом клиенте в таблицу Customers
    INSERT INTO Customers (full_name, phone, email, registration_date)
    VALUES (in_full_name, in_phone, in_email, CURDATE());
END;
```

- Процедура AddCustomer принимает три входных параметра: full_name, phone и email.
- При выполнении этой процедуры в таблицу Customers автоматически добавляется новый клиент с текущей датой регистрации. Это позволяет ускорить процесс ввода данных, минимизируя ошибки и обеспечивая актуальность информации о клиентах в базе данных.

Триггеры для обеспечения целостности данных:

Триггеры — это автоматические SQL-операции, которые выполняются в ответ на определенные события в базе данных, такие как вставка, обновление или удаление данных. Триггеры помогают поддерживать целостность данных и автоматизировать выполнение задач, которые должны происходить при изменении данных.

Пример триггера для обновления остатков товара после продажи:

```
DELIMITER $$

CREATE TRIGGER AfterSaleInsert
AFTER INSERT ON Sales
FOR EACH ROW
BEGIN
    UPDATE Products
    SET stock_kg = stock_kg - NEW.quantity_kg
    WHERE product_id = NEW.product_id;
END $$

DELIMITER ;
```

Этот триггер автоматически уменьшает количество товара на складе в таблице Products каждый раз, когда происходит новая продажа. Он срабатывает после вставки записи в таблицу Sales, обновляя соответствующее количество товара, тем самым поддерживая актуальность данных о товарных остатках.
Представления (или views) — это виртуальные таблицы, которые содержат результаты SQL-запросов. Они упрощают работу с данными, предоставляя доступ к информации без необходимости писать сложные запросы каждый раз.
Пример представления для получения информации о продажах с деталями клиента и продукта:

```
CREATE OR REPLACE VIEW view_SalesDetails AS
SELECT
    s.sale_id,
    c.full_name AS customer_name,
    p.product_name,
    s.quantity_kg,
    s.total_price,
    s.sale_date
FROM Sales s
JOIN Customers c ON s.customer_id = c.customer_id
JOIN Products p ON s.product_id = p.product_id;
```

- Представление view_SalesDetails объединяет данные из таблиц Sales, Customers и Products, предоставляя удобный доступ к полному набору информации о продажах, включая данные о клиентах и продуктах. Это позволяет иметь централизованный

источник для анализа данных о продажах, что улучшает удобство работы с базой данных.

- Это упрощает создание запросов для анализа продаж и формирования отчетности, сокращая время на обработку данных и ускоряя получение нужной информации. Использование представлений позволяет избежать написания сложных SQL-запросов каждый раз, а также повышает эффективность работы с данными.

Заключение

В процессе работы был успешно реализован проект базы данных для учета товаров, клиентов, поставок и продаж в овощном магазине. Основной целью проекта было создание системы, которая автоматизировала бы процессы учета и управления, а также обеспечивала прозрачность всех бизнес-процессов. Завершение ключевых этапов разработки позволило значительно повысить функциональность магазина и улучшить его общую эффективность.

Подведение итогов работы

Разработка и внедрение базы данных позволили автоматизировать важнейшие процессы, такие как учет товаров, управление поставками, учет продаж и клиентов. Это значительно улучшило контроль над товарными запасами на складе, ускорило обработку заказов и снизило вероятность ошибок, связанных с человеческим фактором в учете. Все этапы проектирования и реализации базы данных были выполнены с учетом специфики работы магазина и применяемых технологий.

Основными этапами работы стали:

1. Анализ предметной области и определение требований. На этом этапе была проведена всесторонняя диагностика бизнес-процессов магазина, что позволило точно сформулировать требования к системе и выявить ключевые моменты, которые должны быть учтены в системе. В первую очередь это включало точный учет поступлений и продаж товаров, а также эффективное взаимодействие с клиентами и поставщиками.
2. Проектирование структуры базы данных. На основе собранных требований был разработан как концептуальный, так и логический дизайн базы данных. Структура базы включала создание таблиц для хранения данных о товарах, поставках, клиентах, сотрудниках и продажах, а также определение связей

между этими таблицами. Такой подход позволил четко разделить данные и эффективно организовать их управление, что в свою очередь обеспечило гибкость и производительность всей системы.

3. Реализация базы данных. Были успешно реализованы все компоненты физической структуры базы данных, включая таблицы, атрибуты и связи. Также была внедрена логика обновления данных через хранимые процедуры и триггеры, что позволило автоматизировать процесс учета остатков товаров и обновления информации при совершении продаж. Это значительно ускоряет работу с данными и повышает точность учета.
4. Разработка SQL-запросов. Для получения отчетности, анализа данных и обработки запросов были разработаны SQL-запросы. Это позволило бизнесу оперативно получать актуальную информацию о продажах, товарных остатках на складе и активности клиентов, что способствует более обоснованному принятию управленческих решений.

Практическая значимость проекта

Практическая значимость проекта заключается в значительном улучшении управления магазином и повышении его общей эффективности. Внедрение базы данных позволяет:

- Упрощение и ускорение учета товаров. Информация о товарах, их количестве, стоимости и сроках годности всегда актуальна, что способствует предотвращению дефицита или излишков товара. Точные данные о товарных остатках на складе позволяют владельцам бизнеса быстро принимать решения о пополнении запасов или изменении цен.

- Автоматизация процессов продаж и поставок. Внедрение базы данных позволяет автоматизировать множество процессов, таких как регистрация новых поставок, продажа товаров, обновление остатков и автоматическое обновление данных. Это значительно снижает необходимость вручную обновлять записи и минимизирует вероятность ошибок, повышая общую точность учета.
- Улучшение обслуживания клиентов. Наличие базы данных о клиентах позволяет отслеживать их предпочтения, историю покупок и активность. Это открывает возможности для разработки персонализированных предложений и скидок, что способствует повышению лояльности клиентов и, как следствие, увеличению продаж.
- Анализ данных и принятие обоснованных решений. Использование SQL-запросов позволяет получать необходимые отчеты и аналитику, включая подсчет объема продаж, анализ товарных остатков и исследование клиентской базы. Эти данные помогают руководству магазина принимать более обоснованные решения по ассортименту, ценообразованию и маркетинговым стратегиям.
- Упрощение учета сотрудников. База данных позволяет хранить информацию о сотрудниках магазина, их должностях и активности. Это дает возможность эффективно отслеживать работу персонала, управлять его загрузкой, а также анализировать влияние работы сотрудников на продажи.

Возможности для дальнейшего развития

Разработанная база данных создает основу для дальнейшего расширения и улучшения функционала системы. Одним из наиболее перспективных направлений является интеграция с внешними системами, такими как интернет-магазины, мобильные приложения и другие онлайн-платформы. Внедрение таких решений может существенно повысить удобство и доступность магазина для клиентов, а также улучшить внутренние процессы компании, создавая единую экосистему для управления бизнесом.

Вот несколько направлений для дальнейшего развития проекта:

- Интеграция с интернет-магазином. Внедрение системы для онлайн-заказов позволит не только повысить доступность товаров для клиентов, но и расширить рынок сбыта. Например, покупатели смогут заказывать овощи через интернет, а система автоматически обновит данные о наличии товара на складе и проведет необходимые транзакции.
- Мобильное приложение для клиентов. Разработка мобильного приложения для смартфонов, которое позволит клиентам отслеживать наличие товаров, заказывать их и получать уведомления о скидках, может существенно повысить уровень удобства для пользователей. Мобильное приложение также может интегрироваться с системой лояльности, предоставляя клиентам бонусы за покупки.
- Интеграция с другими учетными системами. Важным шагом будет интеграция с другими системами, такими как учет бухгалтерии или зарплат. Это поможет создать единую экосистему для бизнеса, которая обеспечит более высокую степень автоматизации и снизит необходимость в ручных операциях.

- Аналитика и прогнозирование спроса. В дальнейшем можно интегрировать систему с инструментами для прогнозирования спроса, что позволит заранее планировать закупки и минимизировать издержки на хранение избыточных товаров.

Личный опыт и выводы по работе

Выполнение данного курсового проекта стало значимым этапом в моем профессиональном развитии. Процесс проектирования и разработки базы данных продемонстрировал, как теоретические знания могут быть эффективно применены на практике для решения реальных бизнес-задач. Я освоил основные принципы работы с реляционными базами данных, научился проектировать и реализовывать эффективные структуры для хранения данных, а также работать с системой через SQL-запросы.

Особенно полезным для меня было изучение принципов нормализации базы данных и создание хранимых процедур для автоматизации рутинных процессов. Это значительно углубило понимание того, как обеспечить целостность данных и повысить производительность системы, а также научило меня важности оптимизации работы с базой данных.

Помимо технической стороны проекта, я осознал, насколько важно учитывать особенности предметной области при проектировании информационных систем. Тщательный анализ бизнес-процессов магазина и понимание потребностей конечного пользователя сыграли ключевую роль в создании эффективной базы данных. Этот опыт показал, как важно грамотно проектировать архитектуру системы, чтобы она была не только функциональной, но и гибкой, с возможностью дальнейшего масштабирования.

В целом, данный проект стал важным практическим опытом, который значительно улучшил мои навыки работы с базами данных и дал ценное понимание того, как информационные системы могут эффективно поддерживать бизнес-процессы.

Список литературы

1. Широков, В. В. Основы проектирования баз данных. — М.: Инфра-М, 2018.
2. **Корнейчук, С. И.** Реляционные базы данных: теория и практика. — СПб.: Питер, 2020.
3. **Смирнов, В. В.** SQL для начинающих: Практическое руководство. — М.: ДМК Пресс, 2017.
4. **Иванов, И. И.** Основы информационных технологий. — М.: Высшая школа, 2019.
5. **Петров, И. М.** Разработка информационных систем: Учебное пособие. — М.: КНОРУС, 2021.
6. **Джеймс, В.** Практическое руководство по SQL. — Нью-Йорк: Wiley, 2016.
7. **Наумов, Н. П.** Проектирование баз данных. — СПб.: БХВ-Петербург, 2019.
8. **Леонова, В. М.** Введение в базы данных и SQL. — М.: Бином, 2018.