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

Актуальность темы

В современном мире информационные технологии играют ключевую роль в различных сферах жизни, и бизнес не является исключением. Внедрение и использование баз данных становятся важными составляющими успеха для организаций, стремящихся повысить эффективность своей деятельности. Это особенно актуально для розничной торговли, которая представляет собой важный сектор экономики. Применение автоматизированных систем учета позволяет предпринимателям быстро и точно отслеживать информацию о товарах, поставках, клиентах и продажах. Существующие на сегодняшний день реляционные базы данных позволяют значительно повысить качество учета и оперативность принятия управленческих решений.

В частности, в розничной торговле овощами система учета играет особенно важную роль. Овощи, как товар с ограниченным сроком хранения, требуют точного контроля запасов, а также регулярных поставок и продаж. Ошибки в учете могут привести не только к финансовым потерям, но и к потере доверия со стороны клиентов, что в свою очередь может сказаться на репутации магазина и снизить его конкурентоспособность. Таким образом, внедрение базы данных для учета продукции, поставок, клиентов и продаж становится не только актуальным, но и жизненно необходимым для розничных торговых предприятий, занимающихся реализацией овощей.

Кроме того, автоматизация учета товаров и процессов, связанных с их продажей, позволяет повысить эффективность работы магазина. Внедрение базы данных способствует ускорению и упрощению всех операций, таких как оформление продаж, поиск товаров по запросу, обновление информации о наличии и ценах. Это также позволяет повысить прозрачность бизнес-процессов, избежать ошибок, связанных с

человеческим фактором, и существенно снизить затраты времени на поиск и обработку данных. В условиях современной экономики такие преимущества являются решающими для успешного функционирования бизнеса.

Цель и задачи исследования

Целью данной работы является разработка и внедрение базы данных для овощного магазина, которая обеспечит эффективное ведение учета продукции, клиентов, поставок и продаж. Для достижения этой цели необходимо решить несколько ключевых задач:

1. **Анализ предметной области и определение требований.** Необходимо исследовать существующие бизнес-процессы овощного магазина, выявить основные нужды и требования, которые должна удовлетворять база данных.
2. **Проектирование структуры базы данных**. Следующим шагом полно создание концептуальной и логической модели базы данных, включающей таблицы и взаимосвязи между ними.
3. **Реализация таблиц и связей**. На основе проектирования хватит осуществлено создание физической структуры базы данных, коия включает описание таблиц, атрибутов и ключей.
4. **Разработка SQL-запросов**. Необходимыми станут запросы для извлечения и обработки данных, которые будут применяться для формирования отчетности, анализа продаж и товаров.
5. **Создание хранимых операцй и триггеров**. Для автоматизации некоторых операций в базе данных станут созданы хранимые процедуры и триггеры, например, для обновления остатков товара позже продажи.

Объект и предмет исследования

Объектом исследования считается информационная система учета для овощного магазина, представляющая внешне совокупность баз данных,

приложений и процессов, направленных на эффективное регулирование товарами, продажами, клиентами и поставками. Эта система включает в себе различные компоненты, такие как базу данных для хранения информации о товарах и посетителях, приложения для обработки запросов и отчетности, а также средства для управления доступом и защищенности. Предметом исследования является **реляционная база данных**, коия будет обеспечивать хранение, обработку и управление данными, связанными с работой магазина. Реляционная база данных использует таблицы для представления этих и связи между ними, что является удобным и эффективным приемом хранения информации. Реляционные базы данных широко применяются в бизнесе благодаря своей гибкости, надежности и возможностям для масштабирования, что делает их безукоризненными для учета в магазинах.

Методы и средства разработки

Для разработки базы этих будет использоваться реляционная модель данных, которая считается стандартом для большинства современных информационных систем. Реляционные базы этих обеспечивают гибкость, простоту и возможность масштабирования, что особенно чванно для бизнеса, который может расти и развиваться. Основным инструментом для существа и управления базой данных будет язык **SQL.** SQL является стереотипом для работы с реляционными базами данных и предоставляет множество инструментов для существа таблиц, выполнения запросов и работы с данными.

В качестве системы управления базами этих будет использоваться **MySQL**—популярное СУБД с открытым начальным кодом, которые позволяют эффективно работать с большими размерами данных. Они обеспечивают стабильную работу, высокую производительность и надежность информации, что важно для работы с данными в реальном времени. Для проектирования и управления базой этих будут

использованы такие инструменты, как **MySQL Workbench** для зрительного проектирования схемы базы данных

Глава 1. Теоретические основы исследования баз данных

1.1. Понятие и назначение баз данных

База данных (БД) — это структурированная гербарий данных, которая предназначена для хранения, обработки и управления информацией с целью последующего использования. Основной задачей базы данных является результативное хранение, поиск, обновление и удаление данных. Она предоставляет достижимость пользователям или программам хранить и извлекать данные в удобной для обработки форме.

В бизнесе базы этих широко используются для управления и анализа информации, связанной с продажами, резервами, клиентами, поставками и другими аспектами бизнеса. Например, системы учета запасов товаров в супермаркете является базой этих, где хранятся сведения о товарах, их количестве, стоимости, поставщиках и т.д. Это разрешает оперативно обновлять информацию о наличии товаров и отслеживать продажи, помогая принять верные решения по заказу новых поставок или изменению цен.

В государственном управлении базы этих применяются для хранения и обработки информации о гражданах, транзакциях, управлении государственными источниками. Примером может служить база данных налоговой службы, в коей содержится информация о налогоплательщиках, их декларациях и начислениях налогов. Базы этих также используются в сфере здравоохранения для учета медицинских автомобиль пациентов, в образовании для хранения данных студентов и в правоохранительных органах для учета преступлений.

Базы данных играют ключевую роль в эффективной организации информационных систем. Они дают удобное и структурированное хранилище для данных, что значительно упрощает поиск товара, обработку и анализ информации. Современные информационные системы, будьте то в бизнесе, в государственных учреждениях или в любой другой сфере, не имеют все шансы функционировать без использования баз данных, так как они обеспечивают целостность и доступность этих в реальном времени.

В бизнесе базы данных позволяют ускорить и автоматизировать процессы, эти как обработка заказов, управление запасами и анализ покупательских предпочтений. Используя базы этих, компании могут улучшить прогнозирование спроса, сократить протори на хранение данных и повысить качество обслуживания клиентов. Например, розничные сети

имеют все шансы интегрировать данные о покупках и анализировать их для более точного планирования перечня и выявления популярных товаров.

В государственном управлении базы этих помогают обеспечить прозрачность и подотчетность, улучшить управление государственными источниками, такими как социальные выплаты или налоги, а также эффективно поддерживать предложения для граждан. Например, в сфере здравоохранения базы данных поддерживают значительность информации о медицинских процедурах, рецептах, результатах анализов, что делает легче работу медицинского персонала и позволяет своевременно предоставлять предложения пациентам.

Применение баз данных в этих сферах показывает, как торжественно иметь централизованную систему для работы с данными, что способствует возрастанию общей эффективности и безопасности в работе с информацией.

Сегодня базы этих стали неотъемлемой частью всех современных информационных систем. Они работают основой для работы в таких областях, как финансовые услуги, регулирование цепочками поставок, здравоохранение, электронная коммерция и многие иные. Современные базы данных предлагают гибкость, масштабируемость и высокую производительность труда, что позволяет эффективно работать с большими объемами данных и мгновенно принимать решения на основе этих данных.

Для бизнеса базы этих предоставляют возможность автоматизации ключевых процессов, таких как воинский учет товаров, расчет зарплаты сотрудников, обработка финансовых транзакций и регулирование отношениями с клиентами. В государственных структурах базы данных применяются для организации информации о гражданах, управлении социальными программами, здравоохранении и иных государственных услугах.

Важным аспектом является также обеспечивание безопасности данных. Современные базы данных обеспечивают верные механизмы защиты, такие как шифрование данных, управление доступом и регулярное резервное фотографическое копирование, что позволяет предотвратить утечку информации и защитить данные от несанкционированного доступа.

Таким образом, базы этих обеспечивают основу для всех видов деятельности, связанных с управлением данными, и играют критически важную роль в современном обществе.

1.2. Модели этих и выбор модели для проекта

Для организации данных в информационных системах применяется несколько типов моделей, каждая из которых имеет собственные особенности и области применения. Рассмотрим основные типы моделей этих:

1. **Иерархическая модель**
   Иерархическая модель была одной из первых моделей для организации этих. В этой модели данные представляются в виде дерева, где вся запись имеет один родительский элемент. Это ограничивает пластика модели, так как каждый элемент может быть связан лишь с одним родителем. Например, информация о сотрудниках может быть организована как деревья, где каждый сотрудник является частью более высокоуровневого подразделения.

Недостатки: хитроумие в реализации и ограниченная гибкость — данные могут быть связаны лишь в одном направлении, что делает систему не слишком удобной для динамичных процессов.

2. **Сетевой образець**
   В сетевой модели данные представлены в виде графа, где любой элемент может иметь несколько родительских элементов, что повышает гибкость по сравнению с иерархической моделью. Например, один опт может быть связан с несколькими поставщиками. Эта модель разрешает более точно описывать сложные взаимосвязи данных.

Недостатки: хоть эта модель более гибкая, она все еще сложна в проектировании и эксплуатации. Также эта моделирование требует более сложных механизмов для обработки запросов.

3. **Реляционная моделированиеь**
   самая популярная и широко используемая модель в современных СУБД. Данные сберегаются в виде таблиц, где каждая таблица состоит из строк и столбцов. Таблицы имеют все шансы быть связаны между собой через ключи. Реляционная образец поддерживает использование SQL, который является стандартом для работы с реляционными базами этих.

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

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

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

Преимущества реляционной модели

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

- **Простота**: реляционная образец основана на использовании таблиц, что делает её легко понимаемой и применимой в разных сферах. Каждая таблица соответствует одной сущности, примерно, товары, клиенты, продажи и т.д.
- **Гибкость**: таблицы можно изменять и расширять, добавлять свежие атрибуты, не затрудняя работы всей базы данных. Также разрешается добавлять новые таблицы и устанавливать связи между ними, что гарантирует гибкость в расширении системы.
- **Поддержка SQL**: реляционная модель поддерживает использование информации SQL — языка для работы с базами данных. SQL позволяет легко извлекать эти, делать выборки, обновлять и удалять информацию. Это существенно упрощает работу с данными и дозволяет быстро создавать сложные запросы для извлечения нужной информации.
- **Нормализация**: реляционная моделирование поддерживает процесс нормализации, который помогает уменьшить излишек данных и

улучшить их целостность. Например, данные о товарах и подрядчиках могут быть размещены в отдельных таблицах, с ссылками корешок на друга через внешние ключи, что упрощает обновление и регулирование этими данными.

- **Поддержка транзакций**: реляционные базы этих поддерживают транзакции, что гарантирует целостность данных при выполнении нескольких операций, этих как добавление записей, их обновление или удаление. Это особенно важно для приложений, где условия часто изменяются, и необходимо избежать ошибок в процессе обработки.

Реляционная модели идеально подходит для овощного магазина по следующим причинам:

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

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

Гибкость и масштабируемость: реляционная модели позволяет легко добавлять новые атрибуты или таблицы. Например, позволяется расширить систему, добавив учет клиентов или сотрудников торгового центра. Для этого достаточно добавить новые таблицы и установить взаимосвязи между ними с помощью внешних ключей.

Использование SQL для обзора данных: реляционная модель предоставляет возможность использовать SQL для извлечения информации, что дозволяет создавать отчеты по продажам, остаткам товаров на складе, активности заказчиков и т.д. Например, с помощью SQL можно быстро запросить все товары, коие поступили от конкретного поставщика, или подсчитать общий объем продаж за определение период.

Целостность данных: в реляционной модели данные организованы так, что промахи при их обработке сводятся к минимуму. Например, нельзя случайно прибавить запись о товаре без указания его цены или

количества. Нормализация базы этих также помогает избежать дублирования данных, что улучшает качества работы с базой.

1.3. Основы проектирования реляционных баз данных

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

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

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

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

После сбора притязаний начинается создание ER-диаграмм, которые представляют собой зрительное отображение структуры базы данных. ER-диаграммы позволяют описательно увидеть, какие сущности существуют в системе и как они связаны дружок с другом.

1. **Сущности**: сущности — это основные объекты, данные о коих будут храниться в базе данных. В овощном магазине это имеют все шансы быть такие сущности, как Продукты, Поставщики, Продажи, Клиенты и иные.

2. **Атрибуты**: атрибуты — это характеристики сущностей, которые будут сберегаться в базе данных. Например, для сущности Продукты атрибутами имеют все шансы быть наименование, цена, количество на складе, срок годности и т.д.
3. **Связи**: взаимосвязи определяют, как сущности взаимодействуют друг с другом. Например, связи между Продуктами и Поставщиками может быть представлена чрез внешние ключи, где каждый товар будет связан с генпоставщиком, который его поставляет.

После того как структуры данных отнесены, необходимо провести нормализацию базы данных. Нормализация — это процессия устранения избыточности и улучшения целостности данных, который может помочь уменьшить дублирование данных и улучшить управление данными. Нормализация осуществляется в каплю этапов, которые называются нормальными формами:

1. **1НФ:**
   Каждое ферония в таблице должно содержать только одно значение. Например, коли в одной строке таблицы указаны несколько поставщиков для одного товара, это не соблюдает 1НФ, и нужно разделить данные на несколько строк.
2. **2НФ**:
   Для выполнения 2НФ нужно будет, чтобы каждая неключевая колонка в таблице была сполна функционально зависима от первичного ключа
3. **3НФ**:
   Для выполнения 3НФ данные не обязаны содержать транзитивные зависимости, то есть если один реквизит зависит от другого, то зависимость должна быть только вследствие ключевую сущность. Это минимизирует дублирование данных и улучшает синтез.

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

1.4. Язык SQL и его ипостась в управлении базами данных

SQL (Structured Query Language) — это стандартный формат язык для управления реляционными базами данных, который применяется для работы с данными, их структуры, запросами и правами доступа. SQL считается основным инструментом для взаимодействия с базой данных, и он используется для существа, изменения и извлечения данных, а также для настройки доступа к этим.

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

SQL предоставляет мощные инструменты для:

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

SQL является декларативным языком, что обозначает, что пользователь задает, **что** нужно сделать с данными, а не **как** это нужно сделать. SQL-подход позволяет избежать сложных алгоритмов и сосредоточиться на эффектах.

SQL включает несколько типов команд, которые выполняют всевозможные операции с базой данных. Основные категории команд последующие:

DDL (Data Definition Language) — команды определения данных:

- **CREATE**: применяется для создания таблиц, индексов, представлений и других объектов базы этих.
- **ALTER**: используется для изменения структуры уже существующих объектов базы этих.
- **DROP**: используется для удаления объектов базы данных, этих 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**: применяется для извлечения данных из базы данных.
- **INSERT**: используется для добавления свежих данных в таблицу.
- **UPDATE**: используется для изменения существующих этих в таблице.
- **DELETE**: используется для удаления данных из таблицы.

Пример команды для вставки свежего товара:

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

DCL (Data Control Language) — команды управления доступом:

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

Пример команды для предоставления прав:

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

В контексте овощного торгового центра SQL позволяет эффективно работать с данными о товарах, продажах, посетителях и поставках. Рассмотрим несколько типовых запросов, которые имеют все шансы быть полезны в таком бизнесе:

Поиск товаров по группы и цене

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

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

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

Подсчет единых продаж по товарам

Запрос для подсчета общего объема продаж любого товара:

```
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 гораздо лучше всего, что важно для планирования закупок и управления ассортиментом.

1.5. Обзор средств исследования и управления базами данных

Современные средства разработки и управления базами этих позволяют создавать, поддерживать и оптимизировать работу с базами этих. Они предоставляют удобные инструменты для проектирования, работы с данными, а тоже.одновременно обеспечения безопасности и целостности. Рассмотрим основные средства исследования и управления базами данных на примере популярной СУБД **MySQL** и инструментов, применяемых для работы с ней.

MySQL — это одна из самых популярных и широко применяемых систем управления базами данных с открытым исходным кодом. Она просторно применяется в различных областях: от малых веб-приложений до крупных корпоративных систем. MySQL поддерживает реляционную модельный данных и использует язык SQL для работы с данными.

Возможности MySQL:

- **Реляционная модели данных**: MySQL поддерживает создание и управление таблицами, индексацией, взаимосвязями между таблицами, нормализацией данных.
- **Поддержка транзакций**: MySQL поддерживает транзакции, что гарантирует атомарность операций, и позволяет использовать механизмы отката и сбережения данных.
- **Масштабируемость и производительность**: MySQL оптимизирован для работы с огромными объемами данных и обладает высокой производительностью при обработке запросов.
- **Множественные форматы сбережениия**: MySQL поддерживает несколько движков хранения данных, этих как InnoDB и MyISAM, что позволяет выбирать оптимальный движок в зависимости от притязаний.
- **Поддержка репликации**: MySQL поддерживает репликацию данных, что разрешает создавать резервные копии базы данных и распределять нагрузку промежду серверами.
- **Безопасность**: MySQL поддерживает различные методы аутентификации, кодирование данных и управление правами доступа, что помогает обеспечивать экономическая безопасность данных.

Преимущества MySQL:

- Открытый исходный код: MySQL считается бесплатной СУБД с открытым исходным кодом, что делает её легкодоступной для разработки и использования в разных приложениях.
- Широкая поддержка: MySQL поддерживается на основной массе операционных систем, включая Linux, Windows, macOS и иные.
- Большое сообщество: Существует активное сообщество разработчиков и юзеров, что облегчает поиск решений и документацию по вопросам настройки и оптимизации.
- Производительность: MySQL оптимизирован для стремительной обработки запросов, что делает её хорошим выбором для высоконагруженных приложений.

Применение MySQL:

MySQL применяется для создания и управления базами данных в таких областях, как:

- **Веб-разрабатывание:** используется в большинстве современных веб-приложений и CMS.
- **Бизнес-аналитика:** MySQL может помочь организовать эффективный учет и анализ данных о продажах, резервах, клиентах и других аспектах бизнеса.
- **Интернет-магазины:** для сохранения информации о товарах, клиентах, заказах и транзакциях.
- Образовательные учреждения: для управления данными о студентах, преподавателях, курсах и эффектах.

Для удобной работы с базами данных, разработчики и администраторы баз этих используют различные инструменты, которые облегчают создание, регулирование и анализ данных.

1. **MySQL Workbench**
   MySQL Workbench — это интегрированная среда обитания для работы с MySQL. Она предоставляет набор инструментов для проектирования, администрирования и исследования баз данных, а также для анализа и оптимизации SQL-запросов.

Основные вероятности MySQL Workbench:

- **Визуальное проектирование базы данных**: MySQL Workbench дозволяет создавать ER-диаграммы, проектировать таблицы, связывать их с помощью ключей и наружных связей.

- **Управление базами данных**: инструмент предоставляет функции для существа и удаления баз данных, добавления и изменения таблиц, управления индексами и лимитированиями.
- **Оптимизация запросов:** MySQL Workbench позволяет анализировать выполнение обещаний SQL-запросов и находить узкие места в производительности базы этих.
- **Управление пользователями и правами доступа**: инструмент позволяет рулить правами доступа пользователей к базе данных.

MySQL Workbench считается идеальным инструментом для разработчиков, которым нужно проектировать базы этих, управлять ими и анализировать запросы. Он подходит для сложных систем и комфортен в использовании при работе с большими объемами данных.

Резервное фотографическое копирование и восстановление данных — важная часть администрирования баз данных, коия помогает защитить данные от потерь, сбоев и ошибок.

1. **Резервное копирования:**
   Резервное копирование базы данных заключается в создании её копии в не опасном месте для того, чтобы в случае сбоя можно было реконструировать данные. В MySQL существуют несколько способов резервного копирования:
- **mysqldump**: программа командной строки для создания дампов базы данных в формате SQL. Этот способы создаёт текстовый файл с SQL-командами, которые можно применять для восстановления данных.
- **Репликация**: использование репликации позволяет делать резервные копии базы данных в реальном времени, синхронизируя основная и резервный серверы.
- **Инструменты третьих сторон**: для более трудных и автоматизированных сценариев используются различные программы, например, **Percona XtraBackup** для жгучего резервного копирования.

2. **Восстановление данных:**
   Восстановление данных постфактум сбоя или потери данных осуществляется с помощью резервных копий. В MySQL для поправления используется команда:
   mysql -u username -p database_name < backup_file.sql.

Глава 2. Практическая распродажа базы данных овощного магазина

2.1. Анализ предметной области и художественная постановка задачи

Описание деятельности магазина: закупка, хранение, торговля овощей

Овощной магазин занимается торговлей овощами, что настятельно просит точного и своевременного учета всех процессов — от закупки товаров до их продажи финальным потребителям. Важно наладить эффективный учет и управление данными, ради минимизировать ошибки и обеспечить высокий уровень обслуживания посетителей. Рассмотрим ключевые процессы, которые поддерживает система учета в овощном торговом центре:

1. **Закупка овощей:**
   Закупка — это процесс получения товаров от подрядчиков. Для каждого товара важна информация о его характеристиках и поставщике. Также потребно отслеживать количество товара, поступившее в магазин, чтобы вовремя пополнять запасы и избегать дефицита.
2. **Хранение овощей:**
   После поступления товара в магазинный необходимо организовать его хранение на складе. Это включает в себя диспансерный учет всех товаров, их текущие остатки на складе, срок годности, условия труда хранения и другие параметры, влияющие на качество товара.
3. **Продажа овощей:**
   Продажа — это процедура, при котором товар передается клиенту за оплату. Важно отслеживать, какие товары были реализованы, количество проданных единиц, общую стоимость покупки и информацию о потребителе. Также необходимо учитывать возможные скидки и акции, коие могут влиять на цену товара.

Необходимые сущности и взаимосвязи для учета

Для автоматизации всех процессов учета в овощном торговом центре требуется создание нескольких сущностей в базе данных. Эти сути будут содержать важную информацию о товарах, клиентах, работниках и продажах. Основные сущности и их связи следующие:

1. **Продукты:**
   Сущность "Продукты" хватит хранить информацию о каждом товаре, который есть в торговом центре.

Это включает наименование товара, категорию, цену за килограмм, макроколичество на складе и поставщика.

- Атрибуты: product_id, product_name, category, price_per_kg, stock_kg, supplier_id.

2. **Поставщики**:

   Сущность "Поставщики" станет хранить данные о компаниях или физических лицах, которые поставляют товары в магазины. Необходимо хранить контактную информацию о поставщиках, а также их местоположение.

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

3. **Клиенты**:

   Сущность "Клиенты" довольно содержать информацию о покупателях магазина. Важно хранить данное о клиентах, их покупках и предпочтениях. Это поможет в будущем предложить заказчикам персонализированные предложения и создать программы лояльности.

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

4. **Продажи**:

   Сущность "Продажи" довольно отслеживать данные о каждой продаже: товар, клиент, объем и общая стоимость. Каждая продажа будет связана с товаром и заказчиком.

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

5. **Сотрудники**:

   Сущность "Сотрудники" достаточно хранить информацию о работниках магазина.Эти данные необходимы для учета работы работников и их взаимодействия с клиентами.

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

Связи посередь сущностями:

- Продукты и Поставщики связаны через внешний электроключ supplier_id, так как каждый товар поступает от одного поставщика.
- Продажи связано с Продуктами посредством внешний ключ product_id.так как каждая продажа включает взаперти товар.
- Продажи также связано с Клиентами через внешний накопитель ключ customer_id. Так как каждая продажа принадлежит определенному посетителю.
- Сотрудники не связано напрямую с другими сущностями, но могут быть пригодны для учета действий сотрудников, например, для отслеживания продавцов, коие совершали продажи.

Требования к функционалу базы данных

База этих для овощного магазина должна поддерживать следующие функциональные притязании:

1. **Учет товаров**:

   База данных должна позволять беречь информацию о всех товарах, их характеристиках. Необходимо иметь право добавлять, редактировать и удалять данные о товарах, а также получать информацию о нынешних остатках на складе.

2. **Учет поставок**:

   Важно вести воинский учет поступающих товаров от поставщиков, включая дату поступления, макроколичество, цену и поставщика. База данных должна поддерживать связи между товарами и

поставщиками, что позволяет отслеживать, какие товары поступили от какого генпоставщика.

3. **Продажа товаров**:

   База данных должна поддерживать процедура продажи товаров, включая информацию о клиенте, количестве товара, всеобщей стоимости покупки и дате продажи. Также должна быть шанс учета скидок и акций. Система должна автоматически обновлять остатки товара следом каждой продажи.

4. **Учет клиентов**:

   Необходимо хранить информацию о посетителях, включая их контактные данные и историю покупок. Это поможет торговым центрам предложить индивидуальные скидки, уведомления о скидках, а также наиболее точно анализировать покупательские предпочтения.

5. **Учет сотрудников**:

   База этих должна поддерживать информацию о сотрудниках магазина, включая их должности, контактные данное и дату найма. Это позволит отслеживать, кто был ответственным за определенные операции, эти как продажа товара или прием поставки.

6. **Обработка отчетности**:

   Важной частично работы магазина является анализ данных, таких как общий ресурс объем продаж, прибыль, товарные остатки и популярность продуктов. База этих должна поддерживать запросы для создания отчетов, которые станут помогать менеджерам принимать обоснованные решения.

7. **Масштабируемость**:

   Система обязана быть масштабируемой, чтобы в случае роста бизнеса и повышения объемов данных база данных могла

справиться с дополнительными нагрузками без издержки производительности.

8. **Автоматизация процессов**:

   Важно, чтобы база этих поддерживала автоматизацию операций, таких как обновление остатков товаров в последствии продажи, расчет общей стоимости заказа и т.д. Это поможет увеличить эффективность работы магазина и снизить вероятность ошибок.

2.2. Реализация базы этихх

В этой части главы будет представлен полный код для существа таблиц базы данных, примеры вставки данных для испытания, а также объяснение использования типа данных DECIMAL для учета веса и цен.

Полный код существа таблиц

На основе проектирования структуры базы данных для овощного торгового центра, мы создаем несколько таблиц: Products, Suppliers, Sales, Customers и Employees. Эти таблицы дадут хранение данных о товарах, поставках, продажах, клиентах и работниках.

1. Создание таблицы поставщиков

```
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 заказчиковв

```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)
);
```

Этот код создаёт таблицы для сохранения данных о поставщиках, товарах, клиентах, сотрудниках и продажах. Внешние ключи (FOREIGN KEY) применяются для связывания данных, например, между **Products** и **Suppliers**, **Sales** и **Products**, а вдобавок **Sales** и **Customers**.

Примеры вставки данных для тестирования

Для испытания базы данных необходимо вставить несколько примеров этих в таблицы. Это позволит проверить, как система работает с реальными данными и удостовериться, что все связи между таблицами корректно установлены.

Примеры вставки этих:

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 этих 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

```sql
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 этих 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 вам протестировать работу базы этих, проверить корректность связей и убедиться, что все таблицы содержат необходимую информацию.

Объяснение особенностей типа данных DECIMAL для учета веса и цен

Важной частично базы данных является правильное использование DECIMAL для сохранения данных о цене за килограмм и количестве товара. Тип DECIMAL применяется для хранения чисел с фиксированной запятой, что важно при работе с наличными средствами и точными расчетами.

- **DECIMAL(10, 2)**: означает, что число может содержать до 10 символов, 2 из которых — после запятой. Это идеально подходит для хранения цен и весов с нужной точностью.

- **Точность**: Использование DECIMAL гарантирует точность этих, что необходимо при расчете общей стоимости товаров, а также для учета остатков на складе.

Это обеспечивает, что даже при обработке больших объемов данных, таких как лэндинг товара или количество на складе, будут соблюдены все точности, кроме ошибки округления, которые могут возникнуть при использовании типов этих с плавающей запятой

2.3. Реализация основных SQL-запросов

В этой части руководители мы рассмотрим типовые SQL-запросы, которые обеспечивают функционал для приобретения отчетов по продажам, остаткам на складе, активности клиентов, обновления этих и удаления устаревших записей. Эти запросы являются основными для работы с базой этих овощного магазина и помогают эффективно управлять данными и получать требуемые отчеты.

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

Этот запрос позволяет обрести подробную информацию о продажах за последний месяц, включая эти о клиентах, продуктах и количестве проданных товаров.

```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;
```

Этот интерпелляция извлекает данные о всех продажах, сделанных в последние 30 день, включая информацию о продукте, количестве, стоимости и имени клиента. Он полезен для анализа текущих продаж и активности клиентов.

2. Получение статистики по продажам продуктов

Запрос для приобретения общего количества проданных товаров по категориям и общей суммы спасения:

```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 продажи по категориям продуктов и суммирует эти о количестве проданных товаров и выручке по каждой категории. Это надо для анализа, какие категории товаров более востребованы и навевают наибольшую прибыль.

3. Получение информации о товарных остатках на складе

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

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

Этот запрашивание помогает увидеть товары, которые нужно заказать вторично, так как их количество на складе слишком маленькое. Это полезно для предотвращения недобора товаров.

4. Получение активности клиентов

Запрос для получения перечня 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 запрашивание позволяет увидеть, какие клиенты наиболее активны в последнее лейдейс, сколько покупок они сделали и сколько потратили. Это может быть благотворно для создания программы лояльности или отправки персонализированных предложений.

Запросы для обновления этихх

1. Изменение цен на продукты

Запрос для обновления цены на определенный зоотоварр

```sql
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 имеет возможность быть полезно в случае изменения цен у поставщика или для корректировки цен на основе рыночной обстановки.

2. Обновление количества товара на складе

Запрос для обновления числа товара на складе после поставки:

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

Этот запрос преумножает количество товара на складе на 100 кг для продукта с product_id = 1. Это может быть питательно для учета поставок и пополнения запасов на складе.

3. Изменение этих клиента

Запрос для обновления данных клиента

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

Этот задание обновляет контактные данные клиента, что важно для поддержания актуальности информации о заказчиках.

2.4. Создание хранимых процедур, триггеров и представлений

Для автоматизации коммерция-процессов и улучшения работы с данными в базе данных применяются такие механизмы, как хранимые процедуры, триггеры и представления. Эти инструменты могут помочь автоматизировать повторяющиеся операции, обеспечивать целостность данных и делать лучше удобство работы с часто используемой информацией.

Обоснование применения хранимых процедур для автоматизации повторяющихся операций

Хранимые упражнения — это наборы SQL-запросов, которые сохраняются в базе данных и имеют все шансы быть выполнены по запросу. Они помогают автоматизировать рутинные операции и усовершенствовать производительность системы. Хранимые процедуры удобны, когда одну и то же действие нужно выполнять многократно.

В случае овощного торгового центра хранимые процедуры могут быть использованы для таких операций, как:

- **Оформление продажи**.
- **Добавление свежего клиента в базу данных.**

Преимущества хранимых процедур:

- Упрощение повторяющихся операций: Процедуры дозволяют выполнять сложные операции с несколькими запросами за один раз.
- Уменьшение вероятности промахов: Так как операция выполняется в одном месте, вероятность ошибок при вводе этих снижается.
- Повышение производительности: Процедуры выполняются на стороне сервера, что разрешает уменьшить нагрузку на клиентские приложения и ускорить выполнение.

Пример **хранимой упражнения для оформления продажи**:

```
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 параметры: личные номера товара и клиента, а также количество товара для продажи.
- Она проводит проверку, есть ли достаточное количество товара на складе, и если товара сносно, выполняется вставка записи о продаже в таблицу **Sales**, а тоже.одновременно обновление остатка на складе.

Для добавления нового клиента в базу этих можно создать специальную хранимую процедуру. Вместо того для каждый раз вручную добавлять нового клиента через SQL-запросы, эта операция будет выполнять операцию вставки данных в таблицу Customers.

Пример хранимой упражнения 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 данной процедуры в таблицу **Customers** добавляется новый клиент с нынешней датой регистрации

Триггеры для поддержания целостности данных

Триггеры — это механические SQL-операции, которые выполняются в ответ на определенные события в базе этих, такие как вставка, обновление или удаление данных. Триггеры могут помочь поддерживать целостность данных, например, обновлять остатки товаров постфактум продажи или проверять условия перед вставкой данных.

Пример триггера для обновления остатков товара попозже продажи:

```sql
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 весь раз, когда происходит новая продажа

Представления для удобного отображения убыстренно используемой информации

Представления (или **views**) — это виртуальные таблицы, коие содержат результаты запросов. Они помогают упростить работу с данными, предоставляя путь к информации без необходимости писать сложные SQL-запросы.

Пример представления для приобретения информации о продажах с деталями клиента и продукта:

```sql
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 соединяет данные из таблиц Sales, Customers и Products, предоставляя выигрышный доступ к полному набору информации о продажах, включая условия о клиентах и продуктах.
- Это позволяет упростить запросы для анализа продаж и отчетности, уменьшая время на обработку и получение нужной информации.

Заключение

В ходе исполнения курсового проекта была успешно разработана база этих для учета товаров, клиентов, поставок и продаж в овощном торговом центре. Основной целью работы было создание эффективной информационной системы, способной автоматизировать процессы учета и управления торговым центром, а также повысить прозрачность бизнес-процессов. Для достижения данной цели были выполнены ключевые задачи, что позволило необходимо улучшить функциональность магазина.

Подведение итогов работы

Разработка и имплантация базы данных позволили автоматизировать основные процессы, включая устройства учет товаров, управление поставками, учет продаж и посетителей. Это существенно улучшает контроль над запасами на складе, ускоряет обработку заявок и снижает вероятность ошибок в учете, связанных с человеческим моментом. Все этапы проектирования и реализации базы данных были сделаны с учетом требований, специфики работы магазина и технологий, применяемых для разработки.

Основными этапами работы стали:

1. Анализ предметной области и аппелляционное определение требований. На этом этапе была проведена тщательная диагностирование процессов работы магазина, что позволило точно сформулировать притязании к системе и выявить ключевые моменты, которые система обязана учитывать. Прежде всего, это точный учет поступлений и продаж товаров, а тоже.одновременно взаимодействие с клиентами и поставщиками.
2. Проектирование структуры базы этих. На основе собранных требований был разработан концептуальный и логический проектирование базы данных. Структура включала создание таблиц для товаров, поставок, посетителей, сотрудников и продаж, а также

описание связей между ними. Это обеспечило отчетливое разделение данных и их эффективное управление.

3. Реализация базы этих. Было выполнено создание физической структуры базы этих, которая включает таблицы, атрибуты и связи. Также была продана логика обновления данных через хранимые процедуры и триггеры, что дозволило автоматизировать процесс учета остатков товаров и обновления этих при совершении продаж.
4. Разработка SQL-запросов. Реализованы SQL-запросы для приобретения отчетности, анализа данных и обработки запросов. Это позволило бизнесу практически сразу получать информацию о продажах, остатках товаров на складе и интенсивных клиентах.

Практическая значимость проекта

Практическая значимость плана заключается в существенном улучшении управления магазином и повышения его производительности. Внедрение базы данных позволяет:

- Упрощение и ускорение учета товаров. Теперь информации о товарах, их количестве, стоимости и сроках годности всегда востребована, что помогает избежать дефицита или излишков товаров. Точные эти о товарах и остатках на складе позволяют предпринимателям оперативно брать на себя решения о пополнении запасов или изменения цен.
- Автоматизация процессов продаж и поставок. Внедрение базы этих позволяет автоматизировать многие процессы, включая регистрацию свежих поставок, продажу товаров, обновление остатков и автоматическую исправление данных. Это избавляет сотрудников от необходимости вручную обновлять записи и минимизирует вероятность ошибок.
- Улучшение обслуживания клиентов. Наличие базы этих о клиентах позволяет отслеживать их предпочтения, историю

покупок и действенность. Это дает возможность для разработки персонализированных предложений и скидок, увеличения лояльности клиентов и, как следствие, роста продаж.

- Анализ этих и принятие обоснованных решений. SQL-запросы позволяют получать нужные отчеты и аналитику, такую как подсчет объема продаж, разбирание товарных остатков, исследование клиентской базы. Эти данные могут помочь управленцам магазина принимать более обоснованные решения по перечню, ценообразованию и маркетинговым стратегиям.
- Упрощение учета сотрудников. База этих позволяет хранить информацию о сотрудниках магазина, их должностях и активности. Это выдает возможность эффективно отслеживать работу персонала и управлять его загрузкой, а вдобавок учитывать влияние работы сотрудников на продажи.

Возможности для последующего развития

Разработанная база данных предоставляет 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 значительно повысить удобство и доступность магазина для клиентов, а также улучшить процессы внутри самой компании. 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

Для меня выполнение данного курсового проекта стало важным этапом в профессиональном развитии. Процесс проектирования и разработки базы данных показал, как теоретические знания применяются на практике для решения реальных задач. Я освоил ключевые аспекты работы с реляционными базами данных, научился проектировать и реализовывать эффективные структуры для хранения данных, а также взаимодействовать с системой через SQL-запросы.

Особенно ценным было изучение принципов нормализации базы данных и создание хранимых процедур для автоматизации процессов. Это

позволило углубить понимание того, как обеспечить целостность данных и повысить производительность системы.

Помимо технической части, проект показал, как 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. Я также убедился, как важно грамотно проектировать архитектуру системы, чтобы она была гибкой и масштабируемой.

В целом, данный проект стал важным практическим опытом

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.