****

**Частное учреждение профессионального образования**

**«Высшая школа предпринимательства»**

**(ЧУПО «ВШП»)**

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для магазина по продаже бытовой техники»

Выполнил:

студент 3-го курса специальности

09.02.07 «Информационные системы и программирование»  
Гаврин Кирилл Андреевич

подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

преподаватель дисциплины,  
преподаватель ЧУПО «ВШП»,  
к.ф.н. Ткачев П.С.

оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание**

Введение……………………………………………………........................................3

Глава 1. Теория о разработки базы данных……………………………………….5

1.1 Анализ предметной области..…………………………………………………..5

1.2 Анализ существующий решений ………………………………………………6

1.3 Требования для базы данных ………………………………………………….7

1.4 Выбор СУБД и проектирование схемы базы данных………………………...8

Глава 2. Разработка базы данных…………………………………………………11

2.1 Таблицы и типы значений……………………………………………………..11

2.2 Хранимые процедуры………………………………………………………….15

2.3 Функции………………………………………………………………………...16

2.4 Триггеры………………………………………………………………………..17

2.5 Представление таблиц……………………………………..................................18

2.6 Создание ролей и прав…………………………………………………………18

2.7 Типовые запросы……………………………………………………………….19

Заключение…………………………………………………………..........................23

Список источников……………………………………………………………........24

Приложение 1. Хранимая процедура добавление нового клиента……………...25

Приложение 2. Хранимая процедура вычисления общего дохода за месяц…...25

Приложение 3. Представление получение всех заказов клиентов и их сумма….25

Приложение 4. Ссылка на репозиторий…………………………………………...26

Приложение 5. Антиплагиат……………………………………………………….27

**Введение**

В современном обществе наблюдается не только быстрое развитие технологий, но и стремительное увеличение объемов данных, которые должны быть эффективно управляемы для успешной деятельности бизнеса. В контексте этого развития особенно актуальным является вопрос создания и управления базами данных для компаний, осуществляющих продажу товаров в различных сферах розничной торговли. Постоянно возрастающий объем данных и потребностей потребителей требует от компаний внимательного и эффективного управления всеми аспектами бизнеса.

Актуальность данной темы обусловлена не только ростом объемов продаж, но и увеличением конкуренции в сфере розничной торговли. Современные потребители ставят все более высокие требования к качеству обслуживания и скорости поставки товаров, что требует от компаний разработки эффективных систем управления продажами, складским учетом, управлением закупками и качеством обслуживания клиентов. Создание хорошо организованной базы данных, способной управлять информацией о товарах, заказах, поставках, клиентах и других аспектах бизнеса, является одним из приоритетных направлений развития современного розничного бизнеса в различных отраслях торговли.

Рост конкуренции в розничной торговле и увеличение требований потребителей к качеству обслуживания делают необходимой разработку эффективных баз данных для управления продажами в различных сферах торговли. Таким образом, создание и использование баз данных для управления продажами становится актуальной задачей для современных компаний в контексте конкурентной среды и новых потребностей потребителей.

**Цель**

Целью данной курсовой работы является разработка базы данных для магазина бытовой техники, которая предназначена для оптимизации управления товарами, заказами и клиентской информацией.

**Задачи**

Для достижения этой цели необходимо выполнить нижеперечисленные задачи:

1. Провести анализ требований к базе данных магазина бытовой техники
2. Спроектировать схему базы данных, отражающую основную деятельность магазина бытовой техники
3. Разработать базу данных
4. Заполнить базу данных тестовыми данными

**Объект исследования**

Объектом является база данных

**Предмет исследования**

Предметом исследования является база данных для магазина бытовой техники

**Глава 1. Теория о разработки базы данных**

**1.1.** Анализ предметной области

При проектировании базы данных важно рассмотреть различные виды баз данных и систем управления базами данных (СУБД), чтобы выбрать наиболее подходящий тип для конкретной предметной области. Классифицируют базы данных по разным видам. Здесь основные наиболее распространённые, которые компании применяют для решения повседневных задач.

Виды баз данных:

1. **Реляционные базы данных**:

Описание: Основаны на реляционной модели данных, где данные представлены в виде таблиц, состоящих из строк и столбцов.

Примеры СУБД: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

Применение: Широко используются для приложений, где важна структурированность данных, ACID-транзакции, поддержка сложных запросов и высокая производительность при больших объёмах данных.

2. **Документоориентированные базы данных**:

Описание: Хранят данные в формате документов, например, JSON или XML, где каждый документ может иметь различную структуру.

Примеры СУБД: MongoDB, CouchDB, Firebase.

Применение: Используются для хранения и обработки полуструктурированных и изменяющихся данных, таких как веб-аналитика, контент-менеджмент, сеансовые данные пользователей.

3. **Ключ-значение базы данных**:

Описание: Хранят данные в виде простых пар ключ-значение, где ключ уникален и используется для быстрого доступа к значению.

Примеры СУБД: Redis, DynamoDB, Riak.

Применение: Широко применяются для кэширования данных, сессионного хранения, управления сессиями пользователей и временного хранения данных.

4. **Графовые базы данных**:

Описание: Основаны на теории графов и хранят данные в виде узлов (вершин) и связей (рёбер) между ними.

Примеры СУБД: Neo4j, Amazon Neptune, ArangoDB.

Применение: Используются для моделирования связей и сетей, таких как социальные сети, рекомендательные системы, анализ сетей.

5. **Временные базы данных**:

Описание: Оптимизированы для хранения и обработки временных данных, которые имеют определённый временной интервал жизни.

Примеры СУБД: TimescaleDB, InfluxDB, Kdb+.

Применение: Используются в финансовых аналитических системах, IoT (интернет вещей), мониторинге систем, где требуется высокая скорость записи и запросов по временным рядам.

Существует множество СУБД, каждая из которых предназначена для определённых задач. Вот несколько значимых СУБД, которые широко используются в различных областях:

1. **MySQL** - сочетает в себе хорошую производительность, высокую надежность и простоту, что делает рассматриваемую СУБД популярным выбором у создателей и администраторов веб-сайтов, требующих обработки данных. [9]
2. **MongoDB** - это NoSQL база данных с открытым исходным кодом, ориентированная на хранение данных в формате документов. Она использует JSON-подобные документы для представления информации, что позволяет ей гибко работать с разнообразными типами данных и иерархическими структурами. Эта база данных широко применяется в веб-разработке, аналитике данных, а также для построения высоконагруженных систем, где важна гибкость и масштабируемость хранения данных. [10]
3. **SQLite** - это встраиваемая система управления базами данных, которая отличается своей простотой и компактностью. Вся база данных SQLite хранится в одном файле, что делает её идеальной для встраивания в приложения, где требуется локальное хранение данных без необходимости запуска отдельного сервера баз данных. SQLite широко применяется в различных областях, включая веб-разработку, мобильные приложения, научные и образовательные проекты благодаря своей простоте использования и надёжности в работе с небольшими объёмами данных. [11]

1.2 **Анализ существующий решений**

В процессе разработки базы данных для магазина бытовой техники могут возникать проблемы и ошибки, которые нужно исправлять.

Чтобы их избежать нужно:

Изучить и спроектировать логику базы данных для магазина бытовой техники

Использовать MySQL Workbench как инструмент для создания базы данных

Провести тестирование при проектировании и реализации базы данных для выявления потенциальных проблем с безопасностью и целостностью данных.

1.3 **Требования для базы данных**

Исходя из исследованных типов баз данных, я выбрал реляционные базы данных, они подойдут для создания таблиц и связей между сущностями.

База данных будет хранить информацию о товарах, их характеристиках, а также данные о клиентах и их заказах.

**1.База данных должна находиться в одной из нормальных форм**:

**Третья нормальная форма (3NF) -** является частью процесса нормализации базы данных. Она гарантирует, что каждый столбец в таблице зависит только от ключа записи и непосредственно от него, и не зависит от других столбцов. Другими словами, если у нас есть какая-то зависимость от ключа к оба столбца, она должна быть разделена на разные таблицы. Такие таблицы соответствуют принципу "каждая таблица должна представлять только одну логическую сущность".

**Нормальная форма Бойса-Кодда (BCNF)** - это более строгий вариант третьей нормальной формы. Она применяется тогда, когда в таблице присутствуют множественные кандидатские ключи (то есть несколько столбцов в таблице могут однозначно идентифицировать строки). В BCNF каждая зависимость функциональных атрибутов в таблице должна быть прямой зависимостью от ключа.

**Четвертая нормальная форма (4NF)** - является расширением третьей нормальной формы (3NF) и тесно связана с множественными зависимостями (множественные зависимости, возникающие в табличных данных). Она гарантирует, что в таблице нет "наборов зависимостей" (dependency sets). Если в таблице имеются зависимости между мультимножествами, то эти зависимости должны быть вынесены в отдельные таблицы. Все эти нормальные формы (3NF, BCNF, 4NF) помогают устранить избыточность данных, минимизировать аномалии и обеспечивают стабильную структуру базы данных.

2. **Масштабируемость базы данных**

Одной из важных особенностей является масштабируемость базы данных. Так как внутри магазина может быть много клиентов и товаров, база данных должна быть подготовлена для обработки больших объемов информации и увеличения числа записей в таблицах.

3. **Основные характеристики и функциональные требования**

**1. Характеристики клиентов и их адресов**

Поддержка множественных адресов для одного клиента

Ведение истории изменения данных клиентов и адресов

2. **Характеристики товаров**

Отслеживание количества товара на складе

Обновление информации о товарах и их ценах

3. **Характеристики заказов**

Поддержка множественных заказов от одного клиента

Ведение истории заказов

4. **Управление доступом и безопасность**

Необходимо создать роли для управления базой данных. Разные пользователи будут иметь различные уровни доступа. Администраторы смогут управлять базой данных, включая добавление, изменение и удаление записей. Обычные пользователи смогу только просматривать данные.

**1.4. Выбор СУБД и проектирование схемы базы данных**

В качестве СУБД для данной работы был выбран MySQL. Это решение обусловлено несколькими факторами:

**Простота использования**: MySQL известен своей простотой и удобством, что делает его отличным выбором для учебных проектов и небольших коммерческих приложений.

**Широкое распространение**: MySQL является одной из самых популярных СУБД в мире, что обеспечивает доступ к большому количеству ресурсов, документации и поддержке сообщества.

**Производительность и масштабируемость**: MySQL предлагает высокую производительность и возможности масштабирования, что важно для приложений с большим количеством данных и пользователей.

В качестве инструмента для проектирования был выбран MySQL Workbench – это интегрированная среда для работы с базами данных MySQL. Она предоставляет набор инструментов для проектирования, разработки и администрирования баз данных. Основные функции MySQL Workbench включают:

**Проектирование баз данных:**

**ERD (Entity-Relationship Diagram)**: позволяет создавать и редактировать диаграммы сущностей-связей (ERD), что упрощает проектирование логической и физической модели базы данных.

**Визуальное моделирование данных**: поддержка различных типов связей и атрибутов для создания точных моделей данных.

**Реверс-инжиниринг**: возможность генерировать ERD из существующей базы данных, что позволяет анализировать и модифицировать уже созданные структуры данных.

**Разработка баз данных**:

**SQL редактор**: мощный инструмент для написания и выполнения SQL-запросов, скриптов и процедур.

**Автозавершение**: функция автозавершения кода, которая ускоряет процесс написания запросов и уменьшает количество ошибок.

**Отладка**: возможность отладки и тестирования SQL-кода непосредственно в MySQL Workbench.

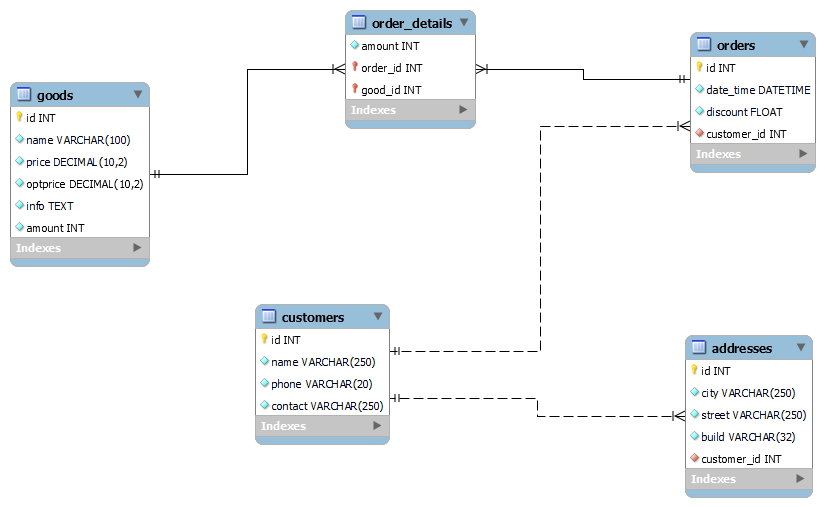
**Администрирование баз данных**:

**Управление пользователями**: инструменты для создания, изменения и удаления пользователей базы данных, а также управления их правами доступа.

**Мониторинг производительности**: встроенные инструменты для мониторинга состояния базы данных, анализа производительности

**Резервное копирование и восстановление**: функции для создания резервных копий базы данных и их восстановления в случае сбоя.

Согласно требованиям задачи мной была спроектирована ER-диаграмма:

****

**Рис. 1.** Схема базы данных

В схеме находится 4 основные таблицы и 1 вспомогательная таблица, используются для связи, каждая из которых реализует или помогает реализовать поставленную задачу:

[customers] клиенты – таблица, которая содержит в себе данные клиентов.

[addresses] адреса – таблица, которая содержит в себе информацию с адресами клиентов. [goods] товары – таблица, которая содержит в себе информацию о товарах.

[orders] заказы – таблица, которая содержит в себе данные о заказах клиентов. [order\_detalis] детали заказов – вспомогательная таблица, которая содержит в себе информацию о заказах.

**Глава 2. Разработка базы данных**

**2.1 Таблицы и типы значений**

В каждой таблице при создании в MySQL для каждого столбца нужно указывать тип данных. Он определяет, какие данные там могут храниться, как будут обрабатываться и сколько места будут занимать. Например, числовой тип данных INT представляет целые числа от -2147483648 до 2147483647 и занимает 4 байта. [1]

Приступим к созданию таблиц по схеме (рис.1), которую я разработал.

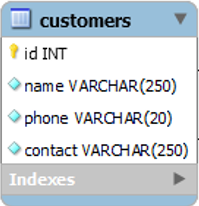
Таблица [customers] содержит столбцы: id, name, phone, contact.

Для колонки используется id тип данных INT, используются Primary Key, Not Null и Auto Increment. Тип данных INT используется для хранения идентификатора клиента. Primary Key используется для уникальной идентификации каждой записи в таблице. Auto Increment используется для создания уникального идентификатора для каждой записи в таблице. Not null – id не может быть нулевым

Для name используется тип данных VARCHAR(250)– покрывает строку, где максимальная длина не превышает 250 символов. Применяется Not Null

Для phone используется VARCHAR(20)- позволяет хранить номера в удобочитаемом формате, который можно легко адаптировать под стандартные и международные форматы телефонных номеров.

Для contact используется VARCHAR(250)- обеспечивает достаточно места для хранения большинства адресов электронной почты.



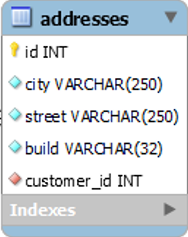
**Рис. 2.** Таблица customers

Таблица [addresses] содержит столбцы: id, city, street, build, customer\_id.

Для колонки id используется INT, Primary Key, Not Null, Auto Increment – используется для идентификации записи

Для сity и street используется VARCHAR(250)- покрывает строку, где максимальная длина не превышает 250 символов, применяется Not Null Для build используется VARCHAR(32)- Этот выбор позволяет эффективно хранить номера домов в базе, учитывая их возможную вариативность, применяется Not Null

customer\_id: тип данных INT. Используется для связи с customers. Связь 1:n один ко многим показывает, что у 1 клиента может быть несколько адресов, применяется Not Null



**Рис. 3**. Таблица addresses

Таблица goods имеет столбцы: id, name, price, optprice, info, amount.

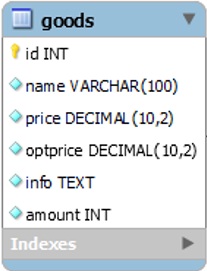
Для колонки id используется INT, Primary Key, Not Null, Auto Increment – используется для идентификации записи

Для колонки name используется VARCHAR(100) - Данный тип подходит для наименования товара, применяется Not Null

Для price, optprice используется тип DECIMAL(10,2)- выбранный тип используется для хранения чисел с фиксированной точностью

Для info используется TEXT – может хранить до 65535 символов, что позволяет вмещать описания, применяется Not Null [2]

Для amount используется INT, Not Null – для хранения количества конкретного товара, применяется Not Null



**Рис. 4.** Таблица goods

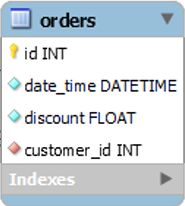
Таблица orders имеет столбцы: id, date\_time, discount, customer\_id

Для колонки id используется INT, Primary Key, Not Null, Auto Increment – используется для идентификации записи

Для колонки date\_time используется тип данных DATETIME – позволяет хранить полные значения даты и времени. Диапазон значений: 1000-01-01 00:00:00 — 9999-12-31 00:00:00, Not Null [3]

Для колонки discount используется FLOAT - используется для хранения скидок, которые могут быть выражены в виде дробных значений, например, 10.5% или 0.15, Not Null [4]

customer\_id: тип данных INT. Используется для связи с customers. Связь 1:n один ко многим показывает, что у 1 клиента может быть несколько заказов, Not Null



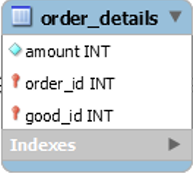
**Рис. 5.** Таблица orders

Таблица order\_detalis имеет столбцы: amount, order\_id, good\_id

Для колонки amount тип данных INT – количество единиц в заказе, Not Null

order\_id: тип данных INT, Primary Key. Внешний ключ для реализации связи n:n многие ко многим, Not Null

good\_id: тип данных INT, Primary Key. Внешний ключ для реализации связи n:n многие ко многим, Not Null



**Рис. 6.** Таблица order\_detalis

**2.2 Хранимые процедуры**

Хранимая процедура MySQL представляет собой подпрограмму, хранящуюся в базе данных. Она содержит имя, список параметров и операторы SQL. Все популярные системы управления базами данных поддерживают хранимые процедуры. Они были введены в MySQL 5.

Существует два вида подпрограмм: хранимые процедуры и функции, возвращающие значения, которые используются в других операторах SQL (например, pi()).

Основное отличие заключается в том, что функции могут использоваться, как любое другое выражение в операторах SQL, а хранимые процедуры должны вызываться с помощью оператора CALL. [5]

Обработчик исключений в MySQL 8 способны генерировать специальные ошибки, которые могут указать на каком этапе была допущена ошибка.

Транзакция в базе данных представляет собой последовательность операций, которые либо успешно завершаются все вместе, либо не выполняются совсем. [6]

В моем случае, с помощью первой процедуры add\_customer мы можем добавить нового клиента в таблицу. В ней содержится транзакция и обработчик исключений.

CREATE DEFINER=root`@`localhost PROCEDURE add\_customer(new\_name VARCHAR(250), new\_phone VARCHAR(20), new\_contact VARCHAR(250))

...

В этом блоке кода определяется название хранимой процедуры и указываются параметры, которые она будет получать, включая их типы данных.

...

BEGIN

DECLARE new\_customer\_id INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

END;

...

В данном блоке кода находится обработчик исключений, который сработает в случае ошибки SQL. При возникновении исключения происходит откат транзакции, чтобы отменить любые изменения, сделанные до момента ошибки.

...

START TRANSACTION;

INSERT INTO customers (name, phone, contact)

VALUES (new\_name, new\_phone, new\_contact);

SET new\_customer\_id = LAST\_INSERT\_ID();

COMMIT;

END ;;

...

В этом месте начинается транзакция и добавляется новый клиент в таблицу customers, используя значения, переданные в хранимую процедуру. В случае исключения она откатывается. Полный код хранимой процедуры см. Приложение 1.

Вторая хранимая процедура calculate\_last\_month\_revenue предназначена для вычисления общего дохода за последний месяц на основе данных о заказах и их деталях. Полный код хранимой процедуры см. Приложение 2.

**2.3 Функции**

Условия используются для выполнения логических проверок и принятия решений внутри запросов или хранимых процедур.

Функции могут использоваться для выполнения вычислений, преобразования данных, агрегации и других задач. В моей базе данных имеется функция с условием, которая вычисляет общую сумму заказа.

CREATE DEFINER=`root`@`localhost` FUNCTION `calculate\_order\_total`(order\_id INT) RETURNS INT

...

В данном блоке кода происходит объявлении функции, функция возвращает целое число INT, представляющее общую сумму заказа.

…

SELECT SUM(IF(od.amount >= 10, g.optprice, g.price) \* od.amount)

INTO total\_amount

FROM order\_details od

JOIN goods g ON od.good\_id = g.id

WHERE od.order\_id = order\_id;

RETURN total\_amount;

…

В этом месте функция вычисляет общую сумму всех значений. Условие проверяет, является ли количество товара больше или равно 10. В зависимости от результата проверки используется либо оптовая цена, либо обычная цена товара.

**2.4 Триггеры**

В данной базе данных триггер используется для автоматического подсчета количества новых записей, вставленных в таблицу customers. Каждый раз при выполнении операции вставки новой строки в таблицу customers, триггер увеличивает переменную @customers\_count на 1, обеспечивая актуальное количество клиентов в базе данных.

...

SET @customers\_count = 0;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`localhost`\*/ /\*!50003 TRIGGER `customers\_counter` BEFORE INSERT ON `customers` FOR EACH ROW SET @customers\_count := @customers\_count + 1 \*/;;

DELIMITER ;

...

**2.5 Представление таблиц**

Представления основаны на запросах SELECT к "реальным" таблицам, но разница в том, что представления не хранят информацию, в отличие от реальных таблиц. Представление только ссылается на таблицы и объединяет их так, как их называет SELECT. Это значительно упрощает часто используемые запросы. [7]

В моей базе данных представление выводит таблицу с информацией, которая включает в себя: id заказа, id клиента, дату и время заказа, общую сумму заказа. Это представление предоставляет удобный способ просмотра данных о заказах вместе с общей стоимостью каждого заказа, рассчитанной на основе количества и цен товаров в каждом заказе. Код представление см. Приложение 3.

**2.6 Создание ролей и прав**

В информационных системах безопасность данных является одним из важнейших аспектов. При работе с базой данных MySQL, рекомендуется следовать принципу наименьших привилегий и создавать пользователей с ограниченным доступом к базе данных. Администратор MySQL должен создавать такие учетные записи для "обычных" пользователей и определять их права доступа. Это позволяет предоставлять пользователям только те привилегии, которые им действительно необходимы для работы, и обеспечивать безопасность данных. Кроме того, проведение аудита полномочий и корректировка их при необходимости также являются важными шагами для обеспечения безопасности базы данных.

В моей базе данных имеется 2 роли: пользователь и администратор. Пользователю позволено выполнять только операции чтения на указанных таблицах в базе данных

SELECT на таблицы:

addresses в базе данных mydb

customers в базе данных mydb

goods в базе данных mydb

order\_details в базе данных mydb

orders в базе данных mydb

Пример создания роли “пользователь”

CREATE ROLE IF NOT EXISTS user\_role;

В этом блоке происходит создание роли

GRANT SELECT ON mydb.addresses TO user\_role;

GRANT SELECT ON mydb.customers TO user\_role;

GRANT SELECT ON mydb.goods TO user\_role;

GRANT SELECT ON mydb.order\_details TO user\_role;

GRANT SELECT ON mydb.orders TO user\_role;

В этой части происходит присвоение права SELECT

CREATE USER IF NOT EXISTS 'user'@'localhost' IDENTIFIED BY '123';

GRANT user\_role TO 'user'@'localhost';

FLUSH PRIVILEGES;

В этом блоке идет создание пользователя user с паролем 123 и активация роли для пользователя.

**2.7 Типовые запросы**

Типовые запросы - это стандартные или часто используемые SQL-запросы, которые выполняют определенные операции с данными в базе данных. Они обычно являются основными элементами взаимодействия с базой данных и включают операции извлечения, вставки, обновления и удаления данных. Основные типы запросов включают:

SELECT (Извлечение):

Запросы SELECT используются для извлечения данных из одной или нескольких таблиц.

INSERT (Добавление):

Запросы INSERT используются для добавления новых строк данных в таблицу базы данных.

UPDATE (Обновление):

Запросы UPDATE позволяют изменять существующие данные в таблице. Они используются для обновления значений определенных столбцов в строках.

DELETE (Удаление):

Запросы DELETE используются для удаления строк из таблицы базы данных. Они позволяют удалить выбранные данные без возможности восстановления.

Типовые запросы обычно оптимизированы для выполнения определенных задач и часто повторяются в различных приложениях и системах управления базами данных. [8]

Первый типовой запрос - вывод общей суммы продаж за последний месяц

SELECT SUM(goods.price \* order\_details.amount) AS total\_sales

FROM goods

JOIN order\_details ON goods.id = order\_details.good\_id

JOIN orders ON order\_details.order\_id = orders.id

WHERE orders.date\_time >= DATE\_SUB(NOW(), INTERVAL 1 MONTH);

Вычисляется общая сумма продаж за последний месяц. Запрос соединяет таблицы 'goods', 'order\_details' и 'orders', чтобы получить информацию о цене товаров, количестве и времени заказов. Выражение 'goods.price \* order\_details.amount' умножает цену товара на количество в заказе, а функция SUM суммирует эти значения. Условие 'WHERE orders.date\_time >= DATE\_SUB(NOW(), INTERVAL 1 MONTH)' фильтрует только те заказы, которые были сделаны в последний месяц, чтобы вычислить сумму продаж за этот период. После выполнения кода выводится общая сумма продаж за последний месяц.

Второй типовой запрос – вывод списка клиентов, у которых не было заказов

SELECT c.id as customer\_id, c.name as customer\_name

FROM customers c

LEFT JOIN orders o ON c.id = o.customer\_id

WHERE o.id IS NULL;

Выбираются id и имя клиентов из таблицы customers. Затем используется операция LEFT JOIN с таблицей orders для соединения клиентов с их заказами по идентификатору клиента. Условие 'WHERE o.id IS NULL' фильтрует только тех клиентов, для которых нет соответствующих записей в таблице заказов. После выполнения кода выводится список клиентов, у которых не было заказов.

Третий типовой запрос - выводит общую сумму и количество заказов каждого клиента

SELECT

customers.id,

customers.name,

COUNT(orders\_with\_total\_price.id) AS orders\_count,

IFNULL(SUM(orders\_with\_total\_price.total\_price), 0) as total\_price

FROM

customers

LEFT JOIN

orders\_with\_total\_price ON orders\_with\_total\_price.customer\_id = customers.id

GROUP BY

customers.id;

Этот запрос выбирает id и имя клиентов из таблицы 'customers', считает количество и общую стоимость их заказов, используя представление 'orders\_with\_total\_price'. Функция IFNULL заменяет NULL на 0 для общей стоимости, если у клиента нет заказов. Результаты группируются по id клиента. После выполнения кода выводится сводка с данными о каждом клиенте, включая его id, имя, количество сделанных заказов и общую сумму потраченных денег за заказы.

Четвертый типовой запрос - подсчет суммарной выручки от продаж в каждом городе

SELECT a.city, SUM(g.price \* od.amount) as total\_revenue

FROM addresses a

JOIN customers c ON a.customer\_id = c.id

JOIN orders o ON c.id = o.customer\_id

JOIN order\_details od ON o.id = od.order\_id

JOIN goods g ON od.good\_id = g.id

GROUP BY a.city;

Этот запрос объединяет данные из таблиц: 'addresses', 'customers', 'orders', 'order\_details', 'goods' для вычисления общей выручки по городам. Он использует JOIN по id клиентов 'customer\_id' и заказов 'order\_id', чтобы связать информацию о городах клиентов с деталями их заказов. Функция SUM вычисляет общую выручку 'goods.price \* order\_details.amount' для каждого заказа. Результаты группируются по городам 'addresses.city', чтобы представить общую выручку по каждому городу. После выполнения кода выводится сводка с данными в каком городе сколько была выручка.

Пятый типовой запрос – этот запрос считает среднюю сумму заказа

SELECT

AVG(orders\_with\_total\_price.total\_price)

FROM

orders\_with\_total\_price;

Этот запрос считает среднее значение (среднюю сумму) столбца total\_price из представления orders\_with\_total\_price.

**Заключение**

В ходе выполнения данной курсовой работы была достигнута цель, заключающаяся в разработке базы данных для магазина бытовой техники, предназначенной для управления товарами, заказами и клиентской информацией. Были исследованы основные потребности магазина бытовой техники в части управления данными, проведен анализ существующих проблем и потребностей, связанных с обработкой и хранением информации. Разработанная база данных предоставляет эффективный инструмент для управления информацией о клиентах, товарах и заказах, что обеспечивает необходимую функциональность для эффективного управления магазином бытовой техники.

Созданная база данных предоставляет возможность удобного и эффективного управления информацией о клиентах, их адресах, товарах и заказах, что позволяет улучшить качество обслуживания клиентов, повысить точность учета товаров и сократить время обработки заказов. Основные преимущества базы данных включают повышение эффективности управления данными, улучшение качества обслуживания клиентов благодаря быстрому доступу к информации о клиентах и их заказах, а также снижение времени обработки данных за счет автоматизации многих процессов.

Таким образом, разработка базы данных для магазина бытовой техники представляет собой важный шаг в направлении улучшения управления и повышения конкурентоспособности компании на рынке розничной торговли, способствуя созданию условий для более эффективного удовлетворения потребностей клиентов и оптимизации внутренних бизнес-процессов.

**Список литературы**

1. Типы данных MySQL [Электронный ресурс] / – Режим доступа: <https://metanit.com/sql/mysql/2.3.php?ysclid=lxll0ditz0825418637>

2. Типы данных в MySQL [Электронный ресурс] / – Режим доступа: <https://selectel.ru/blog/tutorials/data-types-in-mysql/?ysclid=lxlktp1x4n363540339>

3. Календарные типы данных в MySQL: особенности использования [Электронный ресурс] / – Режим доступа: <https://habr.com/ru/articles/69983/>

4. SQL FLOAT: 3 пункта, которые помогут избежать странных математических ошибок [Электронный ресурс] / – Режим доступа: <https://sql-ex.ru/blogs/?/SQL_FLOAT_3_punkta,_kotorye_pomogut_izbeZhat_strannyh_matematicheskih_oshibok.html>

5. Изучаем хранимые процедуры MySQL [Электронный ресурс] / – Режим доступа: <https://olexsyn.gitbook.io/enote/data/mysql/izuchaem-khranimye-procedury-mysql>

6. Транзакция, ACID, CAP теорема и уровни изоляций транзакций простыми словами [Электронный ресурс] / – Режим доступа: <https://habr.com/ru/companies/alfa/articles/812417/>

7. What are views in MySQL? [Электронный ресурс] / – Режим доступа: <https://translated.turbopages.org/proxy_u/en-ru.ru.a0d71907-6672fac7-112a6c85-74722d776562/https/stackoverflow.com/questions/3343136/what-are-views-in-mysql>?

8. Основные типы sql запросов по их видам [Электронный ресурс] / – Режим доступа: <https://studfile.net/preview/16496131/page:3/>

9. MYSQL ПРЕИМУЩЕСТВА И НЕДОСТАТКИ [Электронный ресурс] / – Режим доступа: <https://servergate.ru/articles/mysql-preimushchestva-i-nedostatki/?ysclid=lxmjhbxphq305032991>

10. MongoDB [Электронный ресурс] / – Режим доступа: <https://blog.skillfactory.ru/glossary/mongodb/?ysclid=lxmjio5hm6294419474>11. SQLite — самая простая база данных, которая работает везде [Электронный ресурс] / – Режим доступа: https://thecode.media/sqlite/?ysclid=lxmjjeygd6473400412

**Приложение 1. Хранимая процедура добавление нового клиента**

CREATE PROCEDURE add\_customer(new\_name VARCHAR(250), new\_phone VARCHAR(20), new\_contact VARCHAR(250))

BEGIN

DECLARE new\_customer\_id INT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

END;

START TRANSACTION;

INSERT INTO customers (name, phone, contact)

VALUES (new\_name, new\_phone, new\_contact);

SET new\_customer\_id = LAST\_INSERT\_ID();

COMMIT;

END ;;

**Приложение 2. Хранимая процедура вычисления общего дохода за последний месяц**

CREATE DEFINER=root`@`localhost PROCEDURE get\_last\_month\_revenue()

BEGIN

SELECT

SUM(order\_details.amount \* goods.price) AS last\_month\_revenue

FROM

orders

LEFT JOIN

order\_details ON order\_details.order\_id = orders.id

LEFT JOIN

goods ON goods.id = order\_details.good\_id

WHERE

orders.date\_time >= DATE\_SUB(NOW(), INTERVAL 1 MONTH);

END ;;

**Приложение 3. Представление получение всех заказов клиентов и их сумма**

CREATE OR REPLACE VIEW orders\_with\_total\_price AS

SELECT

orders.id,

orders.customer\_id,

orders.date\_time,

SUM(order\_details.amount \* IF(order\_details.amount >= 10, goods.optprice, goods.price)) AS total\_price

FROM

orders

LEFT JOIN

order\_details ON order\_details.order\_id = orders.id

LEFT JOIN

goods ON goods.id = order\_details.good\_id

GROUP BY

orders.id;

**Приложение 4.** Ссылка на репозиторий



Ссылка для ручного ввода:

|  |
| --- |
| https://github.com/qweasads/bd |

**Приложение 5.** Антиплагиат

