**Министерство науки и высшего образования**

**Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Российский университет дружбы народов ИМЕНИ ПАТРИСА ЛУМУМБЫ»**

Факультет физико-математических и естественных наук

Кафедра информационных технологий

«Допустить к защите»

Заведующий кафедрой  
информационных технологий  
д.ф.-м.н.  
\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Н. Орлов

«» 20г.

**Выпускная квалификационная работа**

**бакалавра**

Направление 02.03.02 «Фундаментальная информатика и информационные технологии»

Тема «Построение электронной системы управления работой складского терминала»

Выполнил студент Турсунов Баходурхон Азимджонович

(Фамилия, имя, отчество)

|  |  |
| --- | --- |
| Группа НФИбд-02–19 | Руководитель выпускной  квалификационной работы |
| Студ. билет № 1032195413 | к. ф-м. н., доц. Панкратов А. С.  (Ф.И.О., степень, звание, должность) |
|  | (Подпись) |
|  | Автор  (Подпись) |

г. Москва

2023 г.

Содержание

[Введение 3](#_Toc133627518)

[1. Теоретический раздел 4](#_Toc133627519)

[1.1. База Данных. 4](#_Toc133627520)

[1.2. История развития базы данных 5](#_Toc133627521)

[1.3. Обработка транзакций в системах баз данных 8](#_Toc133627522)

[Многоуровневые системы баз данных. Иерархическая база данных 12](#_Toc133627523)

[Хранилище данных 17](#_Toc133627524)

[2. Проектирование системы 21](#_Toc133627525)

[2.1. Описание предметной области. 21](#_Toc133627526)

[2.2. Реляционная модель базы данных. 24](#_Toc133627527)

[3. Построение прототипа системы 29](#_Toc133627528)

[Заключение 34](#_Toc133627529)

[Список литературы 35](#_Toc133627530)

[Приложение 1. 36](#_Toc133627531)

# Введение

В современном мире, где склады играют немаловажную роль в поставке товаров и услуг, особенно важно иметь самый эффективный инструментарий для управления складскими операциями.

Электронная система управления работой складского терминала – это незаменимый инструмент, который позволяет контролировать все аспекты работы склада, от приемки товаров до их отправки клиентам.

В этом проекте я предлагаю создать такую систему, которая обеспечит максимальную эффективность и минимальные затраты на управление складских операций. Мы будем использовать передовые технологии и лучшие практики, чтобы создать надежную и простую в использовании систему.

Электронные базы данных имеют множество преимуществ, таких как возможность хранения большого объема информации в одном месте, надежность и целостность хранения данных, удобство извлечения необходимой информации и возможность обмена базами данных между компьютерами.

Целью дипломной работы является Построение электронной системы управления работой складского терминала.

Задачи дипломной работы:

1. Описание предметной области;
2. Выявление объектов, связей и их характеристик;
3. Разработка реляционной и концептуальной схемы Базы Данных;
4. Выбор среды реализации;
5. Построение и отладка прототипа системы;

# Теоретический раздел

## База Данных.

База данных (БД) — это организованный набор данных, который хранится в компьютере или в другом устройстве. Благодаря базам данных, приложения и программы могут легко организовывать, сохранять, изменять и извлекать информацию.

В основе любой базы данных лежит структура данных, которая определяет, каким образом данные будут храниться и организовываться. Основными компонентами любой базы данных являются таблицы, строки и столбцы. Таблицы содержат набор строк, каждая из которых представляет собой запись или элемент данных, а столбцы определяют типы данных, которые будут храниться в таблице.

СУБД (система управления базами данных) — это программа, которая позволяет взаимодействовать с данными в базе данных. СУБД обеспечивает возможность создания новых таблиц, редактирования и удаления существующих таблиц, выполнения запросов к данным, а также управления безопасностью и целостностью данных.

Одним из ключевых аспектов баз данных является их способность работать с большими объемами данных. В этой связи базы данных часто используются в бизнесе для хранения клиентских данных, финансовых отчетов, инвентарных данных и т.д.

Существуют различные типы баз данных в соответствии с их функциональностью, структурой и способом управления данными. Некоторые из наиболее распространенных типов баз данных включают реляционные базы данных, объектно-ориентированные базы данных и графовые базы данных.

Разработка и поддержка баз данных требует определенных знаний и навыков, поскольку необходимо понимать основные принципы архитектуры, проектирования и управления базами данных, а также языки запросов, которые используются для обработки данных в СУБД (например, SQL).

## История развития базы данных

Первый коммерческий компьютер UNIVAC1 был поставлен Бюро переписи населения США в 1951 году. В 1954 году Metropolitan Life стала первой финансовой компанией, купившей 1 такую машину. Это был важный шаг вперед, дающий Metropolitan Life преимущество первопроходца перед другими страховыми компаниями. За UNIVAC последовали IBM701 в 1953 году иIBM650 в 1954 году. В то время IBM предполагала, что будет построено и установлено только 50 машин. К концу производства IBM650 в 1962 году было установлено около 2000 устройств. По сравнению с ручными системами данные теперь обрабатываются значительно быстрее, с меньшим количеством ошибок и более актуальными отчетами. 1. Одним из неприятных побочных эффектов является то, что "все должно быть правильно, это пришло с компьютера", напротив, "я ничего не могу с этим поделать, это на компьютере", и, к счастью, оба подхода больше не в моде.

Ранние компьютерные системы для обработки данных были основаны на существующих ручных системах. Это были последовательные системы, где отдельные файлы состояли из записей, организованных в заранее определенном порядке. Это не была база данных, поскольку не существовало интегрированного источника данных, но электронная обработка файлов была первым шагом на пути к созданию информационной системы электронной базы данных. Обработка должна была начинаться с первой записи и продолжаться до конца. Это было очень хорошо для расчета заработной платы и создания ежемесячных учетных записей, но не сильно помогало при попытке вызвать отдельные записи. В результате нередко появляются последние распечатки файлов, поиск по которым можно выполнять вручную по мере необходимости, пока не будет создана новая версия.Печать по сегодняшним меркам была относительно медленной, поэтому обычно изготавливался только один экземпляр, но использовалась многострочная печать с использованием чередующейся копировальной бумаги. Это также означало, что время отключения было необходимо, когда программа не могла принимать новые обновления до их выполнения.

Дисковое хранилище было доступно, но дорого, поэтому ранние системы, как правило, основывались на 80 рядах перфокарт, перфоленте или накопителе на магнитной ленте, а ввод, обновление и вывод данных должны были выполняться очень специфическим способом. IBM использовала карточный ввод, скопированный на магнитную ленту. Эти данные должны были быть отсортированы. Как только это будет сделано, его можно использовать для объединения и обновления существующих данных. Обычно это сохранялось на "старой" главной ленте, которая представляет собой ленту, содержащую основные записи, обработанные при предыдущем запуске системы (когда вы отключили "новую" главную ленту). Когда он обновляется, создается новая основная лента, и следующий запуск становится старой основной лентой. Мне пришлось запланировать обработку. В этом примере первым выполнением является сортировка, вторым - обработка. С помощью этих систем вы можете загружать и запускать только 1 программу одновременно, поэтому потребность в компьютерном времени быстро возросла. Также требовалось время на обработку данных для разработки, компиляции и тестирования новых компьютерных программ. В конце концов, многие компьютерные системы были запланированы на 1 день 24 часа, до 7 дней в неделю.

Кроме того, при хранении на магнитной ленте действовали очень строгие правила, поэтому, если основная лента была повреждена, ее можно было воссоздать, используя заархивированную старую основную ленту и ленту транзакций. Если возникает проблема с одной из транзакций, транзакция не обрабатывается и копируется в файл исключений на другой ленте. Это будет рассмотрено позже, чтобы выяснить, что пошло не так. Как правило, проблема устраняется путем добавления новой транзакции, которая будет обработана при следующем запуске программы.

IBM представила жесткие диски, которые обеспечивали прямой доступ к данным в 1956 году, но они были относительно маломощными и дорогими по сравнению с ленточными системами. К 1961 году система стала дешевле, и к вашей системе появилась возможность добавить дополнительный накопитель. Преимущество дисковода заключается в том, что он может выполнять обработку транзакций в режиме реального времени и может напрямую обращаться к записям в файле, который в настоящее время является основой для обработки практически во всех коммерческих компьютерных системах. В то же время операционная система была разработана таким образом, чтобы позволить нескольким пользователям и нескольким программам быть активными одновременно, устраняя некоторые ограничения, налагаемые жестким планированием, необходимым для работы на одной технологической машине.

Чтобы сделать хранение и извлечение данных более эффективными, было неизбежно каким-то образом организовать данные. Но пока они разрабатывали эти системы, производители также стремились привязать своих клиентов к своим собственным продуктам. Многим не удалось заставить первых пользователей "неправильной" системы перейти на другие широко распространенные системы, обычно с большими затратами. Например, Honeywell разработала язык под названием FACT (Fully Automated Compiler Technique), предназначенный для реализации корпоративных систем со связанными файловыми структурами. Последним крупным пользователем было Министерство обороны Австралии в 1960-х и 70-х годах, которому потребовались годы и солидный бюджет, чтобы перейти на систему DMS1100 от UNIVAC.

UNIVAC и IBM соревновались за разработку первой базы данных, в которой записи были связаны не последовательно. Поскольку UNIVAC решил принять язык программирования Common Business Oriented Language (COBOL), он также принял конвенцию Конференции по языкам систем данных (CODASYL) о разработке баз данных. CODASYL был консорциумом, созданным в 1959 году для разработки общего языка программирования, который стал основой COBOL. Интересно, что, несмотря на то, что Honeywell входит в группу CODASYL, эта стратегия не увенчалась успехом, поскольку их фактические заявления показали, что они опробованы и функционируют вместо COBOL, который не был опробован. В 1967 ГОДУ CODASYL СМЕНИЛА СВОЕ НАЗВАНИЕ на Database Task Group (DBTG) и РАЗРАБОТАЛА серию РАСШИРЕНИЙ ДЛЯ COBOL для ОБРАБОТКИ БАЗ ДАННЫХ. Honeywell, Univac и Digital Equipment Corporation (DEC) были среди тех, кто внедрил стандарт при внедрении своих баз данных.

IBM, с другой стороны, разработала свой собственный язык программирования PL/1 (Programming language 1) и реализацию базы данных, известную как IMS (Система управления информацией), для очень популярного семейства компьютеров IBM Series 360. Система UNIVAC представляла собой сетевую базу данных, в то время как IBM была строго иерархической. Обе были навигационными системами, которые в какой-то момент получали доступ к системе и перемещались по иерархии, прежде чем выполнять последовательный поиск для поиска требуемых записей. Обе системы опирались на указатели.

В следующих разделах мы рассмотрим базовую обработку транзакций во многих системах баз данных и кратко рассмотрим различные технологии баз данных в беглом порядке.

## Обработка транзакций в системах баз данных

Многие приложения для баз данных имеют дело с обработкой транзакций, поэтому нам нужно выяснить, что такое транзакция. По сути, транзакция — это 1 или более операций, составляющих 1 задачу. Операции делятся на одну из 4 категорий: создание, чтение, обновление или удаление (так называемый CRUDing). В качестве примера вы решили снять деньги с помощью банкомата. Здесь есть несколько транзакций, но сначала они должны быть аутентифицированы системой. Мы фокусируемся на фактическом результате. Система проверяет, достаточно ли у вас денег на счете; затем уменьшает сумму на счете на запрошенную сумму (обновление) и выдает деньги и квитанции. Он также регистрирует операции для записи ваших данных, времени, местоположения и суммы вывода (создания). Могут произойти некоторые вещи, которые приведут к прерыванию транзакции. Возможно, у вас недостаточно денег на счете, или в автомате нет банкнот нужного номинала для выполнения вашего запроса. В таких случаях учетная запись (т.е. база данных) будет возвращена в состояние, предшествующее запуску, поскольку операция будет отменена или откатана назад.

ACID означает атомарность, согласованность, изоляцию и долговечность и имеет важные последствия для обработки транзакций базы данных. В первой половине этой главы мы рассмотрели, как обрабатываются транзакции в последовательной файловой системе, но в многопользовательской системе, которая опирается на единую базу данных, обработка транзакций становится важной частью обработки. Элементы этого обсуждения подробно изложены в главе, посвященной доступности и безопасности.

Атомарность относится к транзакциям, которые применяются по принципу "все или ничего". То есть, если какая-либо часть транзакции завершается с ошибкой, вся транзакция считается завершившейся с ошибкой, и база данных возвращается в состояние, в котором она была до запуска транзакции. Когда это возвращается в исходное состояние, это называется откатом. С другой стороны, если транзакция завершится успешно, база данных будет постоянно обновляться - процесс, называемый фиксацией.

Согласованность означает, что данные, записанные в базу данных, должны быть действительными в соответствии с определенными правилами. В реляционной базе данных это включает ограничения, которые определяют достоверность введенных данных. Например, если будет предпринята попытка создать счет-фактуру и будет использован не назначенный идентификатор клиента, транзакция завершится неудачей (для счетов-фактур, с которыми не связаны клиенты, это также вызовет атомарную возможность отката транзакции).

Изоляция гарантирует, что транзакции, выполняемые параллельно, находятся в том же конечном состоянии, что и те, которые поступают, если транзакции выполняются последовательно. Это важно для онлайн-систем. Например, предположим, что 2 пользователя одновременно пытаются купить последнее место на рейсе авиакомпании. Оба изначально нашли свободное место, но только 1 человек может его купить. Когда вы впервые нажмете кнопку "Купить", чтобы одобрить покупку, транзакция для другого пользователя остановится, а введенные данные (такие как имя и адрес) будут отменены.

Долговечность — это свойство, которое остается зафиксированным при фиксации транзакции. В предыдущем примере покупатель последнего места в самолете завершил транзакцию и получил сообщение об успешном завершении, но по какой-то причине сведения о транзакции системы clat должны храниться в энергонезависимой области, чтобы их можно было обработать при восстановлении системы до применения последнего обновления.

**Двухфазная фиксация**

Двухфазная фиксация — это метод обработки транзакций, при котором база данных хранится на нескольких серверах. Это называется распределенной базой данных, и они более подробно обсуждаются ниже. Но дискуссия о значении обработки транзакций относится именно к этому разделу. При распределенной обработке баз данных система управления базами данных должна координировать фиксацию или откат транзакций как автономный блок для всех компонентов базы данных, независимо от того, где они физически расположены, чтобы не повлиять на транзакцию с точки зрения acid a (атомарность) и C (согласованность). Это повлияет на работу приложения. Существует 2 фазы: фаза запроса и фаза фиксации. На этапе запроса процесс согласования отправляет запрос к сообщению о фиксации всем другим процессам. Ожидаемый ответ - "Да". На этапе фиксации координатор отправляет сообщение другим процессам, пытающимся завершить фиксацию (делая изменения постоянными). Если чей-то процесс завершается сбоем, все процессы выполняют откат.

Oracle называет это двухэтапной фиксацией, но она определяет 3 фазы: подготовка, фиксация и забвение. На этапе подготовки все узлы, на которые ссылаются в распределенной транзакции, подготавливаются к фиксации стартовым узлом, который является глобальным координатором.

Затем каждый узел записывает информацию в журнал повторов, чтобы он мог зафиксировать или выполнить откат. Он также устанавливает распределенную блокировку на таблицу, подлежащую изменению, чтобы не производилось никаких операций чтения. Узел сообщает глобальному координатору, что он готов к фиксации, доступен только для чтения или прерван. Только для чтения означает, что данные на узле не могут быть изменены запросом, поэтому их не нужно подготавливать и они не участвуют в фазе фиксации. Если ответ - abort, это означает, что узел не может быть успешно подготовлен. Затем узел откатывает локальную часть транзакции. Когда 1 узел выдает прерывание, действие распространяется между остальными узлами, и транзакции, гарантирующие ACID A и C, также откатываются.

Если все узлы, участвующие в транзакции, отвечают "подготовлено", процесс переходит к фазе фиксации. Шаги на этом этапе предназначены для фиксации узла глобального координатора. Если это удается, глобальный координатор инструктирует все остальные узлы зафиксировать транзакцию. Затем каждый узел выполняет фиксацию и обновляет локальный журнал повторов перед отправкой сообщения о фиксации. Если фиксация завершается неудачей, отправляется сообщение, и все сайты откатываются.

Фаза забывания — это фаза очистки. Когда все участвующие узлы уведомляют зафиксированного глобального координатора, отправляется сообщение для очистки информации о статусе транзакции.

SQLServer использует аналогичную двухфазную стратегию фиксации, но она включает в себя третью фазу (забыть) в конце второй фазы.

## Многоуровневые системы баз данных. Иерархическая база данных

Хотя многоуровневые базы данных больше не распространены, IMS, многоуровневая система баз данных, по-прежнему является одним из самых высокооплачиваемых продуктов IBM и все еще активно разрабатывается, поэтому стоит потратить время на их обсуждение. Однако это программный продукт для мэйнфреймов, и своей долговечностью он отчасти обязан организациям, которые были привязаны к этому продукту с первых дней развития массовых вычислений. В то время как основным языком IMS обычно является IBM PL/ 1, COBOL также распространен.

Подобно сетевой базе данных, структура иерархической базы данных зависит от указателя. Основное отличие заключается в том, что нам нужно перемещаться с вершины дерева (рис. 1.1.).

Изображение выглядит как текст, Шрифт, линия, диаграмма

Автоматически созданное описание

Рисунок 1.1. Иерархическая база данных.

**Сетевая база данных**

В сетевой базе данных, такой как DMS1100 от UNIVAC, есть 1 запись, которая является родительской записью. В данном примере это пример на рис. 2. Это запись клиента. Он определяется несколькими атрибутами (например, идентификатором клиента, именем, адресом). С этим связано большое количество дочерних записей, в данном случае порядок с несколькими атрибутами. То, как они связаны, зависит от дизайна базы данных. По умолчанию часто использовался указатель "next", например, если родительский элемент указывал на первого дочернего элемента, а у первого дочернего элемента был указатель на 2-го дочернего элемента. Последний дочерний элемент возвращает указатель на родительский элемент. Если вам нужен более быстрый доступ, вы можете определить "передний" указатель, который позволяет осуществлять навигацию вперед и назад. Наконец, если вам нужно больше гибкости, вы можете определить "прямой" указатель, который указывает непосредственно из дочерней записи на родительскую. Компромисс был между скоростью доступа и скоростью обновления, особенно в случае вставки новых дочерних записей и удаления записей. В таких случаях указатель должен был быть обновлен.

Изображение выглядит как текст, диаграмма, Шрифт, План

Автоматически созданное описание

Рисунок 1.2. Сетевая база данных.

Все определение базы данных и ее работа были организованы на языке программирования COBOL, который был расширен для работы с приложениями баз данных DMS1100. Это основная причина, по которой язык COBOL все еще используется сегодня. Стоит отметить, что языковая ошибка привела к опасениям сбоя основного компьютера (что на самом деле означает базу данных) в полночь 1999-12-31. Это называлось ошибкой Y2K или millennium и было связано с методом хранения дат (в некоторых системах сохранялись только последние 2 цифры года, поэтому годом был 1900), и тот факт, что предполагаемый сбой так и не произошел, был учтен в усилиях по техническому обслуживанию, которые были предприняты в системе в предыдущие годы.

**Реляционная база данных**

Они появились в постановке Эдгара Кодда 1970 года.

Статья «Реляционная модель данных для больших общих банков данных» (Codd1970). То, что стало корпорацией Oracle, использовало ее в качестве основы для крупнейшей корпоративной системы управления реляционными базами данных. Он также был разработан, чтобы быть независимым от платформы, поэтому не имело значения, какое у вас оборудование.

Основой любой реляционной системы является набор таблиц записей с определенными атрибутами, связанных набором соединений. Эти соединения создаются с использованием внешнего ключа, атрибута, который содержит те же данные, что и первичный ключ другой таблицы. Первичный ключ — это уникальный идентификатор записи в таблице. Этот подход к хранению данных был очень эффективным с точки зрения используемого дискового пространства и скорости доступа к записям.

На рисунке 3 у меня есть простая база данных, состоящая из двух таблиц. Один содержит записи о сотрудниках, а другой содержит записи, относящиеся к отделам. Идентификатор отдела можно рассматривать как связь (объединение) между двумя таблицами, которая является первичным ключом таблицы отдела и внешним ключом таблицы сотрудников. Он также показывает отношение «один ко многим», означающее, что для каждой записи в таблице «Отдел» есть много записей в таблице «Сотрудник». Верно и обратное: для каждой записи в таблице «Сотрудники» есть только одна запись в таблице «Отдел».

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 1.3. Реляционная база данных, состоящая из двух таблиц или отношений

Данные в реляционных базах данных обрабатываются с помощью языка структурированных запросов (SQL). Он был формализован ANSI (Американским национальным институтом стандартов) в 1986 году. Существует семь редакций SQL86, последней из которых является SQL2011 или ISO/IEC9075:2011 (Международная организация по стандартизации/Международная электротехническая комиссия). SQL имеет прямое отношение к реляционной алгебре. Он описывает таблицы (реляционные алгебраические термины — отношения), записи (кортежи) и отношения между ними.

Oracle и Microsoft SQL Server являются примерами систем реляционных баз данных. Хотя Microsoft Access обладает многими функциями систем реляционных баз данных, включая возможность манипулирования данными с помощью SQL, это строго система управления реляционными базами данных.

**Объектно-ориентированная база данных**

Большая часть программирования сегодня выполняется на объектно-ориентированных языках, таких как Java и C++. Они создают богатую среду, в которой данные, а также процедуры и функции, необходимые для работы с ними, хранятся вместе. Реляционные базы данных часто рассматриваются объектно-ориентированными программистами как единый постоянный объект, над которым можно выполнять несколько операций.

Одной из первых проблем, с которыми сталкиваются базы данных, является появление буквенно-цифровых данных. Нам все больше и больше нужно хранить, обрабатывать и извлекать изображения, аудиофайлы, карты и видео. Даже традиционные данные рассматриваются иначе, чем традиционные объединения таблиц. Он вводит объектно-ориентированные конструкции, такие как иерархии, агрегаты и указатели. Это привело не только к множеству инноверсий, но и к стандартной фрагментации.

С середины 1980-х годов было разработано много объектно-ориентированных систем управления базами данных (OBMS), но они не получили широкого распространения в бизнес-среде. Они были популярны на нишевых рынках, таких как географические и инженерные области, где графические данные должны были храниться и обрабатываться. Как упоминалось выше, одной из больших проблем с объектно-ориентированными базами данных является создание стандарта. Это было предпринято группой управления объектными данными (ODMG), опубликовавшей пять редакций стандарта. Группа закрылась в 2001 году. Функциональность ODMG была заменена группой управления объектами (OMG). Ходят разговоры о разработке стандарта 4-го поколения для объектных баз данных, но этого пока не произошло

Проблема с разработкой объектно-ориентированных систем заключалась в том, что большой опыт и системы были разработаны вокруг реляционных баз данных, что сделало SQL, к лучшему или к худшему, универсальным языком запросов. Поэтому второй подход к объектной ориентации заключался в расширении реляционных баз данных для включения объектно-ориентированной функциональности. Они называются объектно-реляционными базами данных. Они управляются расширенными командами SQL для управления объектными структурами.

## Хранилище данных

Проблема с оперативной базой данных заключается в том, что отдельные записи в ней постоянно обновляются, что затрудняет анализ на основе его исторических данных, если только они не хранятся активно. Концепция хранилищ данных заключается в том, что они могут хранить данные, созданные различными системами в пределах организации, а не только транзакции, составляющие базу данных. Затем вы можете проанализировать эти данные и использовать результаты анализа для принятия решений и информирования о стратегическом направлении вашей организации. Хранилище данных уже упоминалось ранее. В одном месте они обсуждались в роли организационных активов. Вот краткий обзор некоторых технических деталей.

Хранилище данных — это центральный репозиторий данных внутри организации, в котором хранятся исторические данные, и, который постоянно заполняется текущими транзакциями. Данные могут быть проанализированы для создания отчетов о тенденциях. Его также можно анализировать с использованием методов интеллектуального анализа данных для получения стратегической информации. В отличие от данных реляционных баз данных, данные хранилища данных организованы в денормализованном формате. Существует 3 основных формата:

* Схема «звезда»: таблицы — это таблицы фактов, в которые записываются данные о конкретных событиях, таких как сделки купли-продажи, и таблицы измерений, в которых записывается связанная информация.
* Атрибуты в таблице фактов (рис. 1.4.)
* Схемы «снежинка» также основаны на таблицах фактов со связанными таблицами измерений. Однако к этим таблицам размеров могут быть привязаны дополнительные таблицы, создающие вид снежинок (рис. 1.5.)

Изображение выглядит как текст, снимок экрана, визитная карточка, Шрифт

Автоматически созданное описание

Рисунок 1.4. Схема звезда

Интеллектуальный анализ данных — это процесс извлечения информации из хранилища данных и преобразования ее в понятную структуру. Термин анализ данных начинает преобладать над интеллектуальным анализом данных. Этот процесс состоит из нескольких шагов:

* Очистка данных была некорректной, а повторяющиеся данные удалены. Например, две записи о клиентах могут быть идентичными, но одна из них содержит название пригорода как часть адреса, а другая — нет. Один должен быть удален.
* Первоначальный анализ, при котором определяются качество и распределение данных.
* Это делается, чтобы определить, есть ли какие-либо перекосы или другие проблемы в распределении данных.
* Для ответа на некоторые вопросы применялись статистические модели, основной анализ обычно задавали руководству.

Изображение выглядит как текст, снимок экрана, Прямоугольник, дизайн

Автоматически созданное описание

Рисунок 1.5. Схема снежинка

Интеллектуальный анализ данных — это лишь один из методов группы, известной под общим названием бизнес-аналитика, которая преобразует данные в полезную бизнес-информацию. Другие включали управление бизнес-процессами и бизнес-аналитику. Дальнейшее использование данных в хранилище данных заключается в создании систем принятия решений на основе закономерностей в данных. Этот процесс известен как машинное обучение. Нейронные сети — работы, использующие данные для обучения компьютерных программ, имитирующих нейронные связи в мозге для распознавания образов, и системы на основе правил, в которых генерируются деревья решений.

# Проектирование системы

## Описание предметной области.

Предметной областью является складской терминал, по приёму, хранению и продажам товаров.

Клиент может купить товар или совершить возврат товара. Клиенты оплачивают покупки с помощью карты или наличных, либо могут оплатить покупку при получении.

Складской терминал – это программа, которая помогает управлять операциями на складе. Она обеспечивает автоматизацию процессов от приемки товара на склад до отгрузки их покупателям.

Складом управляет администратор склада. Он должен проверять заявку на закупку товаров, после, убедиться что эти товары в наличии и утверждать ее, а также проводить и знать любые операции, которые происходят на складе.

В складе существует отдел закупок, который принимает решение по приобретению товаров у поставщиков

Поставщик же, напрямую поставляет товары на склад.

Сотрудники склада, а конкретно агенты (курьеры) – доставляют и обеспечивают заказ к клиентам.

Поставка включает в себя состав поставки, которая включает в себя товар.

Системы складского терминала объединяют множество функций и возможностей, включая:

* Управление запасами товаров.
  + Система складского терминала позволяет узнать количество товаров на складе, а также следить за их перемещением по складу. Каждому товару присваивается уникальный идентификатор, который используется для отслеживания перемещения продуктов внутри склада.
* Учет поставок
  + Система складского терминала позволяет вести учет поставок товаров на склад. Пи получении новых товаров все данные о них загружаются в систему, а затем используются для обновления информации о складских запасах.
* Учет отгрузок.
  + Также система складского терминала позволяет контролировать отгрузки, что помогает избежать ошибок при отправке заказов клиентам. Данные о заказах экспортируются напрямую в систему отгрузки.
  + Мониторинг сроков годности.
  + С помощью системы складского терминала можно автоматически мониторить срок годности товаров, что особенно полезно при хранении продуктов питания и лекарственных препаратов.
* Автоматизация процессов.
  + Основная цель складского терминала – автоматизация складских процессов. Он позволяет значительно повысить эффективность управления запасами, сократить время на обработку заказов и минимизировать ошибки в работе.
* Мобильность.
  + Система складского терминала может быть установлена на мобильных устройствах, что позволяет операторам склада быстро получать необходимую информацию о состоянии запасов и перемещении товаров по складу.
* Использование технологий RFID и штрихкодирования.
  + Использование технологии радиочастотной идентификации (RFID) и штрихкодирования товаров позволяет ускорить процессы складского учета, облегчить отслеживание перемещения товаров и уменьшить количество ошибок в работе.
* Анализ данных.
  + Система складского терминала может анализировать данные о записях и транзакциях на складе, что дает возможность для принятия интеллектуальных решений и более эффективного управления запасами товаров.
* Управление персоналом.
  + Система складского терминала может также использоваться для управления персоналом на складе. Она может вести учет рабочего времени, распределять задачи между операторами и отслеживать выполнение работ на складе.
* Системы уведомлений.
  + Система складского терминала может предоставлять форму оповещения для пользователей, если происходят нештатные ситуации, например, если возникают проблемы с поставкой товаров или если запасы истощаются. Это поможет операторам склада быстро реагировать на изменения в состоянии склада и своевременно принимать меры.

Для более подробного описания предметной области была нарисована концептуальная модель БД (Рис 2.1)

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 2.1. Концептуальная схема базы данных.

## Реляционная модель базы данных.

Для реализации базы данных построена реляционная модель, представляющая собой перечень реляционных таблиц, с перечислением их атрибутов, указанием первичных, внешних ключей и связей ключей разных таблиц (таблица 2.1–2.9).

|  |
| --- |
| Idсотрудника  Фио  Должность  email |

Таблица 2.1. Схема таблицы (Сотрудник)

Таблица 2.2. Схема таблицы (Клиент)

|  |
| --- |
| Idклиента  Id заказа  Email  Адрес  Телефон |

Таблица 2.3. Схема таблицы (Заказ)

|  |
| --- |
| Idзаказа  Id сотрудника  Id курьера  Сумма  Статус  Дата заказа |

Таблица 2.4. Схема таблицы (Товар)

|  |
| --- |
| Idтовара  Idкатегории  Название  Описание  Адрес хранения |

Таблица 2.5. Схема таблицы (Категория)

|  |
| --- |
| Idкатегории  Название  Описание |

Таблица 2.6. Схема таблицы (Состав поставки)

|  |
| --- |
| Idпоставки  Название  Единица измерения  Описание  Количество |

Таблица 2.7. Схема таблицы (Поставка)

|  |
| --- |
| Idпоставки  Idсотрудника  Дата |

Таблица 2.8. Схема таблицы (Поставщик)

|  |
| --- |
| Idпоставщика  Название компании  Адрес  Email |

Таблица 2.9. Схема таблицы (Корзина)

|  |
| --- |
| Idзаказа  Idкорзины  Idтовара  Сумма  Количество  Единица измерения |

Из выше представленных таблиц была составлена реляционная схема базы данных (рисунок 2.2).

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 2.2. Реляционная схема базы данных

# Построение прототипа системы

Для создания таблиц была написана программа на языке запросов SQL. Скрипт таблицы выглядит следующим образом:

CREATE TABLE Клиент

(

Id\_клиента int PRIMARY KEY,

Id\_заказа int,

Email varchar(255),

Адресvarchar(255),

);

Перейдем к заполнению таблиц, которые были заполнены экспериментальными данными. Имена, адреса и другие данные, внесенные в базу, являются случайными.

После написания программного кода всех таблиц поочередно запрашиваем SELECTзапросы в программе pgAdmin 4. Рассмотрим таблицы базы данных нашего склада на рисунках

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 3.1. Вид таблицы "Клиенты"

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3.2. Вид таблицы "Заказ"

Изображение выглядит как стол

Автоматически созданное описание

Рисунок3.3. Вид таблицы "Сотрудник"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.4. Вид таблицы "Товар"

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3.5. Вид таблицы "Категория"

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.6. Вид таблицы "Состав поставки"

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.7. Вид таблицы "Поставка"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.8. Вид таблицы "Поставщик"

Изображение выглядит как календарь

Автоматически созданное описание

Рисунок 3.9. Вид таблицы "Корзина"

# Заключение

Построение электронной системы управления работой складского терминала является актуальной и важной задачей в современном бизнесе. Это позволяет существенно упростить и ускорить работу с товаром, улучшить планирование и контроль всех процессов, а также повысить качество обслуживания клиентов.

Создание такой системы требует комплексного подхода, который включает в себя разработку базы данных, программного обеспечения, аппаратных средств, а также обучение и подготовку персонала. В рамках данной работы были созданы таблицы базы данных, заполненные данными, которые могут быть использованы для дальнейшей разработки программного обеспечения.

Результатом работы над проектом должна стать функциональная система, которая будет позволять управлять всеми процессами на складском терминале, от приема товара до его отгрузки, а также предоставлять оперативную информацию о состоянии складских запасов, выполненных заказах и т. д. В итоге создание электронной системы управления работой складского терминала может привести к улучшению бизнес-процессов, сокращению времени на обработку заказов, повышению эффективности работы и, как следствие, увеличению прибыли компании.

# Список литературы

* + 1. К. Дэйт. Введение в системы баз данных. - 8-е издание. - М.: Издательский дом "Вильямс", 2016.
    2. Э.Кодд. Реляционная модель данных для больших вычислительных систем. - М.: Мир, 1978.
    3. А.Сильверстоун. Проектирование баз данных. - М.: ДМК Пресс, 2017.
    4. М.Якобсон. Реляционные базы данных: проектирование и разработка. - М.: Издательство Триумф, 2018.
    5. Д.Маккей. Реляционный язык SQL. - М.: Издательство Триумф, 2015.
    6. Х.Кент. Базы данных для практиков. Проектирование и разработка. - М.: Издательство "Диалог-МИФИ", 2018.
    7. Д.Дж.Крелдль. Разработка баз данных на языке SQL. - М.: ДМК Пресс, 2015.
    8. Т.Коннолли, К.Бегг. Базы данных системы и проектирование. - М.: Издательский дом "Вильямс", 2016.
    9. М.Винсент. Практика проектирование баз данных. - М.: Козицкий, 2017.

# Приложение 1.

Скрипты к таблицам БД, изображенных на рисунках (3.1-3.9)

CREATE TABLE Заказ (

id\_заказа INT PRIMARY KEY,

id\_сотрудника INT,

id\_курьера INT,

сумма DECIMAL(10,2),

статус VARCHAR(20),

дата\_заказа DATE,

);

CREATE TABLE Сотрудник (

Id\_сотрудника INT PRIMARY KEY,

должностьVARCHAR(50),

почтаVARCHAR(50)

);

CREATE TABLE Клиент (

id\_клиента INT PRIMARY KEY,

id\_заказа INT,

почтаVARCHAR(50),

адресVARCHAR(100),

дата\_заказа DATE,

телефон VARCHAR(20),

);

CREATE TABLE Товар (

id\_товара INT PRIMARY KEY,

id\_категории INT,

название VARCHAR(50),

описание VARCHAR(200),

адрес\_храненияVARCHAR(100),

);

CREATE TABLE Категория (

id\_категории INT PRIMARY KEY,

название VARCHAR(50),

описание VARCHAR(200)

);

CREATE TABLE Состав\_поставки (

id\_поставки INT PRIMARY KEY,

название VARCHAR(50),

единица\_измеренияVARCHAR(20),

описание VARCHAR(200),

количество INT

);

CREATE TABLE Поставка (

id\_поставки INT PRIMARY KEY,

id\_сотрудника INT,

дата\_поставки DATE,

);

CREATE TABLE Поставщик (

id\_поставщика INT PRIMARY KEY,

название\_компанииVARCHAR(50),

адрес VARCHAR(100),

почтаVARCHAR(50)

);

CREATE TABLE Корзина (

id\_заказа INT,

id\_корзины INT,

id\_товара INT,

суммаDECIMAL(10,2),

количество INT,

единица\_измеренияVARCHAR(20),

);