**План**

Введение…………………………………………………….2

**Глава 1. Теоретические аспекты СУБД**

1. Основные понятия……………………………………….3
2. Функциональные возможности СУБД…………………7
3. Архитектура систем управления………………………..9
4. Типы СУБД………………………………………………13

**Глава 2. Разработка базы данных**………………16

Заключение………………………………………………….21

Список литературы…………………………………………22

 **Введение.**

Развитие средств вычислительной техники обеспечило для создания и широкого использования систем обработки данных разнообразного назначения. Разрабатываются информационные системы для обслуживания различных систем деятельности, систем управления хозяйственными и техническими объектами, модельные комплексы для научных исследований, системы автоматизации проектирования и производства, всевозможные тренажеры и обучающие системы. Одной из важных предпосылок создания таких систем стала возможность оснащения их «памятью» для накопления, хранения и систематизация больших объемов данных. Другой существенной предпосылкой нужно признать разработку подходов, а также создание программных и технических средств конструирования систем, предназначенных для коллективного пользования. В этой связи потребовалось разработать специальные методы и механизмы управления такого рода совместно используемыми ресурсами данных, которые стали называться базами данных. Исследования и разработки, связанные с проектированием, созданием и эксплуатации баз данных, а также необходимых для этих целейязыковых и программных инструментальных средств привели к появлению самостоятельной ветви информатики, получившей название системы управления данными.

 Такие программные комплексы выполняют довольно сложный набор функций, связанный с централизованными управлениями, данными в базе данных интерфейсах всей совокупности ее пользователей. По существу, система управления базами данных служит посредником между пользователями и базой данных.

 В настоящее время разработаны и используются на персональных компьютерах около двадцати систем управления базами данных. Они представляют пользователю удобные средства интерактивного взаимодействия с БД и имеют развитый язык программирования

 Данная работа представлена некоторые теоретические аспекты теории Б, основные понятия, функциональные возможности систем управления БД, а также описана БД по рынку бытовой химии города Улан-Удэ.

 **I. Теоретические аспекты СУБД.**

 **1. Основные понятия БД.**

Всякая прикладная программа является отображением какой – то части реального мира и поэтому содержит его формализованное описание в виде **данных.** Крупные массивы данных размещают, как правило, отдельно от исполняемого программы, и организуют в виде **Базы данных.** Начиная с 60-х годов для работы с данными, стали использовать особые программные комплексы, называемые **системами управления базами данных (СУБД).** Системы управления базами данных отвечают за:

* физическое размещение данных и их описаний;
* поиск данных;
* поддержание баз данных в актуальном состоянии;
* защиту данных от некорректных обновлений и несанкционированного доступа;

обслуживание одновременных запросов к данным от нескольких пользователей (прикладных программ).

 **Модели данных.**

Хранимые в базе данных имеют определенную логическую структуру, то есть, представлены некоторой моделью, поддерживаемой СУБД. К числу важнейших относятся следующие модели данных:

* иерархическая;
* сетевая;
* реляционная;
* обЪектно – ориентированная;

 В иерархической модели данные представляются в виде древовидной (иерархической) структуры. Она удобна для работы с иерархически упорядоченной информацией и громоздка для информации со сложными логическими связями.

 Сетевая модель означает представление данных в виде произвольного графа. Достоинством сетевой и иерархической моделей данных является возможность их эффективной реализации показателей затрат памяти и оперативности. Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе.

 Реляционная модель данных (РМД) название получила от английского термина Relation – отношение. Модель данных описывает некоторый набор родовых понятий и признаков, которыми должны обладать все конкретные СУБД и управляемые ими БД, если они основываются на этой модели.

 Объектно-ориентировочная модель данных – это когда в базе хранятся не только данные, но и методы их обработки в виде программного кода. Это перспективное направление, пока также не получившее активного распространения из-за сложности создания и применения подобных СУБД.

 **База данных** - это совокупность записей различного типа, содержащая перекрестные ссылки.

 Файл - это совокупность записей одного типа, в котором перекрестные ссылки отсутствуют.

 Более того, в определении нет упоминания о компьютерной архитектуре. Дело в том, что, хотя в большинстве случаев БД действительно представляет собой один или (чаще) несколько файлов, физическая их организация существенно отличается от логической. Таблицы могут храниться как в отдельных файлах, так и все вместе. И, наоборот, для хранения одной таблицы иногда используются несколько файлов. Для поддержки перекрестных ссылок и быстрого поиска обычно выделяются дополнительные специальные файлы.

 Поэтому при работе с базами данных обычно применяются понятия более высокого логического уровня: запись и таблица, без углубления в подробности их физической структуры.

 Таким образом, сама по себе база данных - это только набор таблиц с перекрестными ссылками. Чтобы универсальным способом извлекать из нее группы записей, обрабатывать их, изменять и удалять, требуются специальные программы, называются **СУБД.**

 По характеру использования СУБД делят на персональные (СУБДП) и многопользовательские (СУБДМ).

 К персональным СУБД относятся VISUAL FOXPRO, ACCESS и др. К многопользовательским СУБД относятся, например, СУБД ORACLE и INFORMIX. **Многопользовательские СУБД** включают в себя сервер БД и клиентскую часть, работают в неоднородной вычислительной среде допускаются разные типы ЭВМ и различные операционные системы. Поэтому на базе СУБДМ можно создать информационную систему, функционирующую по технологии клиент-сервер. Универсальность многопользовательских СУБД отражается соответственно на высокой цене и компьютерных ресурсах, требуемых для поддержки

 СУБДП представляет собой совокупность языковых и программных средств, предназначенных для создания, ведения и использования БД.

 **Персональные СУБД** обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними, и при необходимости создания приложений, работающих с сервером БД.

 Для обработки команд пользователя или операторов программ в СУБДП используются **интерпретаторы команд** (операторов) и **компиляторы**. С помощью компиляторов в ряде СУБДП можно получать исполняемые автономно приложения-ехе-программы.

 Обеспечение целостности БД-необходимое условие успешного функционирования БД. Целостность БД-свойство БД, означающее, что база данных содержит полную и непротиворечивую информацию, Для обеспечения целостности БД накладывают ограничения целостности в виде некоторых условий, которым должны удовлетворять хранимые в базе данные. Примером таких условий может служить ограничение диапазонов возможных значений атрибутов объектов, сведения о которых хранятся в БД, или отсутствие повторяющихся записей в таблицах реляционных БД.

 **Обеспечение безопасности** достигается СУБД шифрованием прикладных программ, данных, защиты паролем, поддержкой уровней доступа к базе данных, к отдельной таблице.

 Расширение возможностей пользователя СУБДП достигается за счет подключения систем распространения Си или Ассемблера.

 **Поддержка функционирования в сети обеспечивается:**

* средствами управления доступом пользователей к совместно используемым данным, т.е. средствами блокировки файлов (таблиц), записей, полей, которые в разной степени реализованы в разных СУБДП;
* средствами механизма транзакций, обеспечивающими целостность БД при функционировании в сети.

Теперь рассмотрим **функции СУБД** немного подробнее:

  **Определение данных.**

 СУБД должна допускать определения данных (внешние схемы, концептуальную схему, внутреннюю схему, а также все связанные отображения) в исходной форме и преобразовывать эти определения в форму соответствующих объектов. Иначе говоря, СУБД должна включать в себя компонент языкового процессора для различных языков определений данных. СУБД должно также «понимать» синтаксис языка определений данных.

 **Обработка данных.**

СУБД должна уметь обрабатывать запросы пользователя на выборку, изменение или удаление существующих данных в базе данных или на добавление новых данных в базу данных. Другими словами, СУБД должна включать в себя компонент процессора языка обработки данных.

 Запросы языка обработки данных бывают «планируемые» и «не планируемые».

1. **Планируемый запрос-это** запрос, необходимость которого предусмотрена заранее. Администратор базы данных, возможно, должен настроить физический проект БД таким образом, чтобы гарантировать достаточное быстродействие для таких запросов.
2. **Не планируемый запрос-это,** наоборот, специальный запрос, необходимость которого не была предусмотрена заранее. Физический проект БД может подходить, а может и не подходить для рассматриваемого специального запроса. В общем, получение возможной наибольшей производительности для не планируемых запросов представляет собой одну из проблем СУБД. (Подробнее эта проблема будет обсуждаться в следующих частях книги.)

**Безопасность и целостность данных.**

СУБД должна контролировать пользовательские запросы и пресекать попытки нарушения правил безопасности и целостности, определенные АБД.

**Восстановление данных и дублирование.**

СУБД или другой связанный с ней программный компонент, обычно называемый администратором транзакций, должны осуществлять необходимый контроль над восстановлением данных и дублированием. Подробности использования этих функций системы приводятся далее в этой книге.

 **Словарь данных.**

СУБД должна обеспечить функцию **словаря данных**. Сам словарь данных можно по праву считать БД (но не пользовательской, а системой). Словарь «содержит данные о данных» (иногда называемые **метаданными**), т.е. **определения** других объектов системы, а не просто «сырые данные». В частности, исходная и объектная формы различных схем (внешних, концептуальных и т.д.) и отображений будут сохранены в словаре. Расширенный словарь будет включать также перекрестные ссылки, показывающие, например, какие из программ какую часть БД используют, какие отчеты требуются тем или иным пользователям, какие терминалы подключены к системе и т.д. Словарь может быть (а на самом деле даже должен быть) интегрирован в определяемую им БД, а значит, должен содержать описание самого себя. Конечно, должно быть возможность обращения к словарю, как и к другой БД, например, для того узнать, какие программы и/или пользователи будут затронуты при предполагаемом внесении изменения в систему. (Дальнейшее обсуждение этого вопроса приводится в следующих главах книги.)

 **Производительность.**

 Очевидно, что СУБД должна выполнять все указанные функции с максимально возможной эффективностью.

 Подводя итог сказанному, можно сделать вывод, что в целом назначением СУБД является предоставление **пользовательского интерфейса** с БД. Пользовательский интерфейс может быть определен как граница в системе, ниже которой все невидимо для пользователя. Следовательно, по определению пользовательский интерфейс находится на внешнем уровне. Тем не менее, иногда встречаются случаи, когда внешнее представление вряд ли значительно отличается от относящейся, по мере в современных коммерческих продуктах.

 В заключении вкратце сопоставим описанную СУБД с системой файлами (или с управлением файлами). В своей основе система управления файлами является компонентом общей системы, которая управляет хранимыми файлами; проще говоря, она «ближе к диску», чем СУБД. Таким образом, пользователь системы управления файлами может создавать и уничтожать хранимые файлы, а также выполнять простые операции выборки и обновления хранимых записей в таких файлах. Однако, в отличие от СУБД, системы управления файлами имеют некоторые недостатки.

 **2. Функциональные возможности СУБД.**

Управляющим компонентом многих СУБД является ядро, выполняющее следующие функции:

* управление данными во внешней памяти;
* управление буферами оперативной памяти (рабочими областями, в которые осуществляется подкачка данных из базы для повышения скорости работы);
* управление транзакциями.
1. **Непосредственное управление данными во внешней памяти.**

 Эта функция включает обеспечение необходимых структур внешней памяти, как для хранения данных, непосредственно входящие в базу данных так и для служебных целей. Например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используется индекс).

 В некоторых реализациях СУБД активно используется возможность существующих файловых систем. В других работа производится вплоть до уровня устройств внешней памяти. Но подчеркнем, что в развитых СУБД пользователь в любом случае не обязан знать использование СУБД файловую систему и если использует, то, как организованные файлы. В частности СУБД поддерживает собственную систему и наименование объектов баз данных.

1. **Управление буторами оперативной памяти.**

 СУБД обычно работает с БД, по крайней мере, этот размер обычно существует, больше доступен объему оперативной памяти. Что если при обращении к любому элементу данных будет производиться объем с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практическим единственным способом реально увеличение этой скорости является буферизация данных в оперативной памяти. При этом даже если операционная система производит общесистемную буферизацию. Этого не достаточно для цели СУБД, которая располагает гораздо больше информации о полезности буферизации, т.е. той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти, собственной дисциплины замены буферов. Заметим, что существуют отдельные направления СУБД, которые ориентированно, но постоянно присутствуют в оперативной памяти БД. Это направление основывается на предположение, что на столько велик, что позволит, не беспокоится о буферизации. (Пака эта работа находится в стадии развития).

1. **Управление транзакциями.**

 Транзакция – это последовательность операций над БД, рассматриваемая СУБД как единое целое. При выполнении транзакция может быть либо успешно завершена, и СУБД зафиксирует произведенные изменения во внешней памяти, либо, например, при сбое в аппаратной части ПК, ни одного из изменений не отразится в БД. Понятие транзакция необходимо для поддержания логической целостности БД. Таким образом, поддержание механизма транзакции является обязательным условием даже однопользовательских СУБД. (Если такая система заслуживает СУБД). Но понятие транзакция гораздо более важно много пользователь СУБД, то свойство, то каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостное после своего завершения, делает очень удобным, использование понятие транзакция как единицы активности пользователя по отношению БД. При соответствующем управлении управляющимися транзакциями со стороны СУБД каждым использованием может в принципе ощущать себя единственным пользователем СУБД. Управление транзакции многопользовательской СУБД связаны важные понятия сериализация транзакции и сериального плана выполнения смеси транзакции. Под стерилизацией выполнении параллельно сериализация понимают такой порядок планирования их работ при которой суммарный эффект смеси транзакции эквивалентен эффекту их некоторого последовательного управления. Сериальный план выполнения смеси транзакции это такой план, который приводит к сериализация транзакции. Что если удается добиться действительного сериального выполнения смеси транзакции, то для каждого пользователя по инициативе, которой образованна транзакция присутствие других транзакций будет незаметно (если не считать некоторого замедления работы по сравнению с одно пользованием режимом). Существует несколько базовых алгоритмов сериализация транзакции. Централизованных СУБД наиболее распространены алгоритмы, основанные на синхронизации захвата объектов БД. При использовании любого алгоритма возможная ситуация конфликта между двумя или более транзакциями по доступу объекта БД. В этом случае для поддержания сериализация необходимы, выполнять откат одной ли более транзакции. Это один из случаев, когда пользователь многопользовательской СУБД может реально (и достаточно неприятно) ощутить присутствие в системе транзакции других пользователей.

1. **Архитектура СУБД.**

Три уровня архитектуры.

 Архитектура ANSI/SPARC включает три уровня: внутренний, концептуальный и внешний. В общих чертах они представляют собой следующее:

* Внутренний уровень-это уровень, наиболее близкий к физическому хранению, т.е. связанный со способами сохранения информации на физических устройствах хранения.
* Внешний уровень наиболее близок к пользователям, т.е. он связан со способами представления данных для отдельных пользователей.
* Концептуальный уровень-это «промежуточный» уровень между двумя первыми.

Внешний уровень (индивидуальные представления пользователей).

Концептуальный уровень (обобщенное представление пользователей).

Внутренний уровень (представление в

памяти).

 Если внешний уровень с индивидуальными представлениями пользователей, то концептуальный уровень связан с обобщенным представлением пользователей. Иначе говоря, может быть несколько внешних представлений, каждое из которых состоит из более или менее абстрактного представления определенной части БД, и может быть только одно концептуальное представление, состоящее из абстрактного представления БД в целом.

 **Внешний уровень-это** индивидуальный уровень пользователя. Пользователь может быть прикладным программистом или конечным пользователем с любым уровнем профессиональной подготовки. Особое место среди пользователей занимает администратор БД. (В отличие от остальных пользователей его интересует также концептуальный и внутренний уровень.)

 У каждого пользователя есть свой язык общения.

* Для прикладного программиста это либо один из распространенных языков программирования, такой как C, COBOL или PL/1, либо специальный язык рассматриваемой системы. Такие оригинальные языки называют (неформально!) языками четвертого поколения на том основании, что машинный код, язык ассемблера и такие языки, как COBOL, можно считать языками трех первых «поколений», а оригинальные языки модернизированы по сравнению с языками третьего поколения так же, как языки третьего поколения улучшены по сравнению с языком ассемблера.
* Для конечного пользователя это или специальный язык запросов, или язык специального назначения, возможно, основанный на формах и меню, созданный специально с учетом требований и поддерживаемый некоторым оперативным приложением.

#  Хотя с точки зрения архитектуры удобно различать подъязык данных и включающий его базовый язык, на практике они могут быть неразличимыми настолько, насколько это имеет отношение к пользователю. Безусловно, сточки зрения пользователя, предпочтительнее, чтобы они неразличимы или трудно различимым, их называют сильно связанными. Если они ясно и легко различаются, говорят, что они слабо связаны. Большинство систем на сегодняшний день поддерживает лишь слабую связь. Система с сильной связью могли бы предоставить пользователю более унифицированный набор возможностей, но, очевидно, требуют больше усилий со стороны системных проектировщиков и разработчиков (которые, вероятно, рассчитывают на статус-кво); однако есть основания предполагать, что на протяжении следующих нескольких лет будет происходить постепенное продвижение к более сильно связанным системам.

 Язык обработки данных состоит из таких выполняемых операторов PL/1, которые передают информацию **в** и **из** БД; опять же, возможно, включая, новые специальные операторы.

 В общем, внешнее представление состоит из множества экземпляров каждого типа внешней записи, которые, в свою очередь, отнюдь не обязательно должны совпадать с ранимыми записями. Находящийся в распоряжении пользователя подъязык данных определен в терминах внешних записей; например, операция выборки языка обработки данных будет проводить выборку из экземпляров внешних, а не хранимых записей.

 **Концептуальный уровень.**

 Концептуальное представление – это представление всей информации БД в несколько более абстрактной форме (как и случае внешнего представления) по сравнению с физическим способом хранения данных. Однако концептуальное представление существенно отличается от способа представления данных какому-либо отдельному пользователю. Вообще говоря, концептуальное представление – это представление данных такими, какие «они есть на самом деле», а не такими, какими вынужден их видеть пользователь в рамках, например, определенного языка или используемого аппаратного обеспечения.

 Концептуальное представление состоит из множества экземпляров каждого типа **концептуальной записи**. Например, оно может состоять из набора экземпляров записей, содержащих информацию об отдельных, плюс набор экземпляров, содержащих информацию о деталях и т.д. Концептуальная запись вовсе не обязательно должна совпадать с внешней записью, с одной стороны, и с хранимой записью- с другой.

 Концептуальное представление определяется с помощью **концептуальной схемы,** которая включает определения каждого типа концептуальных записей. Концептуальная схема использует другой язык определения данных - **концептуальный.**

 Концептуальное представление – это представление всего содержимого базы данных, а концептуальная схема – это определение такого представления. Однако было бы ошибкой полагать, что концептуальная схема – это не более чем набор определений, больше напоминающих простые отношения записей в программе на языке COBOL (или каком-либо другом).

 Теперь перейдем к более детальному исследованию трех уровней архитектуры.

 **Внутренний уровень.**

Третьим уровнем архитектуры является внутренний уровень. **Внутреннее представление – это** представление нижнего уровня всей БД; оно состоит из многих экземпляров каждого типа **внутренней записи.** Термин «внутренняя запись» принадлежит терминологии ANSI/SPARC и означает конструкцию, называемую хранимой записью. Внутреннее представление так же, как внешнее и концептуальное, не связано с физическим уровнем, так как в нем не рассматриваются физические области устройства хранения, такие как цилиндры и дорожки. Другими словами, внутреннее представление предполагает бесконечное линейное адресное пространство; подробности того, как адресное пространство отображено на физическое устройство хранения, очень зависят от системы и умышленно не включены в общую архитектуру.

 Внутреннее представление описывается с помощью **внутренней схемы,** которая определяет не только различные типы хранимых записей, но также существующие индексы, способы представления хранимых полей, физическую последовательность хранимых записей и т.д. Внутренняя схема пишется с использованием еще одного языка определения данных – внутреннего.

 В заключении отметим, что в некоторых исключительных ситуациях прикладные программы, в частности те, которые называют утилитами могут выполнять операции непосредственно на внутреннем, а не на внешнем уровне. Конечно, такой практикой пользоваться не рекомендуется; она определяет риск с точки зрения безопасности (правила безопасности игнорируются ) и целостности (правила целостности тоже игнорируется), к тому же программа будет зависеть от загруженных данных; но иногда это может быть единственным способом достичь выполнения требуемой функции или добиться необходимого быстродействия – так же, как пользователю языка высокого уровня иногда по тем же причинам необходимо прибегнуть к языку ассемблера.

 Приложения, использующие базы данных, обычно принято относить к одной из программных архитектур, имеющих свои плюсы и минусы.

 **Локальная архитектура**.

 И программа, и база данных расположены на одном компьютере. В такой архитектуре работает большинство настольных приложений.

 **Файл – серверная архитектура.**

 База данных расположена на мощном выделенном компьютере (сервере), а персональные компьютеры подключены к нему по локальной сети. На этих компьютерах установлены клиентские программы, обращающиеся к базе данных по сети. Преимущество такой архитектуры заключается в возможности одновременной работы нескольких пользователей с одной базой данных.

Недостаток такого подхода - большие объемы информации, передаваемой по сети. Вся обработка выполняется на клиентских местах, где фактически формируется копия базы данных. Это приводится к ограничению максимально возможного числа пользователей и большим задержкам при работе с базой. Эти задержки вызываются тем, что на уровне конкретной таблицы одновременный доступ невозможный. Пока программа на одном из клиентских мест не закончит работу с таблицей (например, не выполнит модификацию записей), другие программы не могут обращаться к этой таблице. Это называется блокировкой на уровне таблицы и исключает возникновение путаницы в ее содержимом.

 **Клиент – серверная архитектура**.

 В такой архитектуре на сервере не только хранится БД, но и работает программа СУБД, обрабатывающая запросы пользователей и возвращающая им наборы записей. При этом программы пользователей уже не работают, например, с БД как набором физических фалов, а обращаются к СУБД, которая выполняет операции. Нагрузка с клиентских мест при этом снимается, так как большая часть работы происходит на сервере. СУБД автоматически следит за целостностью и сохранностью БД, а также контролирует доступ к информации с помощью службы паролей. Клиент – серверные СУБД допускают блоки на уровне **записи** и даже отдельного **поля.** Это означает, что с таблицей может работать любое число пользователей, но доступ к функции изменения конкретной записи или одного из ее полей обеспечен только одному из них.

 Основной недостаток этой архитектуры не очень высокая надежность. Если сервер выходит из строй, вся работа останавливается.

 **Распределенная архитектура.**

 В сети работает несколько серверов, и таблицы баз данных распределены между ними для достижения повышенной эффективности. На каждом сервере функционирует своя копия СУБД. Кроме того, в подобной архитектуре обычно используются специальные программы, так называемые **серверы приложений**. Они позволяют оптимизировать обработку запросов большого числа пользователей и равномерно распределить нагрузку между компьютерами в сети.

Недостаток распределенной архитектуры заключается в довольно сложном и дорогостоящем процессе ее создания и сопровождения (администрирования), а также а высоких требованиях к сервером компьютерам.

 **Интернет – архитектура.**

Доступ к базе данных и СУБД (распространенных на одном компьютере или в сети) осуществляется из броузера по стандартному протоколу. Это предъявляет минимальные требования к клиентскому оборудованию. Такие программы называют «тонкими клиентами», потому что они способны работать даже на ПК с процессором 80386. Благодаря стандартизации всех протоколов и внедрять. Например, можно не организовывать локальную сеть, а обращаться к серверу через Интернет в локальной сети (в таком случае говорят о технологиях интранет). В этом случае не требуется разрабатывать специальные клиентские программы или придумывать собственные спецификации обмена данными между сервером и клиентскими местами. Достаточно использовать готовые броузера и программные решения.

 **4. Типы СУБД.**

 Системой управления базами данных называют программную систему, предназначенную для создания на ЭВМ общей базы данных для множества приложений, поддержания ее актуальном состоянии и обеспечения эффективности доступа пользователей к содержащимся в ней данным в рамках предоставленных им полномочий. СУБД предназначена, таким образом, для централизованного управления БД как социальным ресурсом в интересах всей совокупностей ее пользователей. Доступ к базе данных отдельных пользователей при этом возможен только через посредство СУБД.

 По степени их универсальности различаются два вида СУБД – системы **общего назначения** и **специализированные** системы. СУБД общего назначения не ориентированны на какую – либо конкретную предметную область или на информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной обстановки, и поставляется многим пользователям как коммерческое изделие. СУБД общего назначения обладают средствами настройки на работу с конкретной БД в условиях конкретного применения.

 Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, основанных на технологии БД, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Развитые функциональные возможности таких СУБД, присущая им, как правило, функциональная избыточность позволяют иметь значительный «запас мощности», необходимый для безболезненного эволюционного развития построенных на их основе информационных систем в рамках их жизненного цикла. Вместе с тем средства настройки дают возможность достигнуть приемлемого уровня производительности информационной системы в процессе ее эксплуатации.

 Однако в некоторых случаях доступные СУБД общего назначения не позволяют добиться требуемых характеристик производительности и/или удовлетворить заданные ограничения по объему памяти, предоставляемой для хранения БД. Тогда приходится разрабатывать специализированную СУБД для данного конкретного применения. Решение указанных проблем при этом может оказаться возможным благодаря знанию специфических особенностей данного применения, к которым оказываются нечувствительными средства настройки доступных СУБД общего назначения, либо за счет ущемления каких либо функций системы, не имеющих жизненно важного значения. Как правило, в этой роли оказываются, прежде всего функции, обеспечивающие комфортную работу пользователя.

 Создание специализированной СУБД – весьма трудоемкое дело даже в сравнительно простых случаях, и для того, чтобы избрать этот путь, нужно иметь действительно веские основания и твердую убеждаемость в невозможности или нецелесообразности использования какой – либо СУБД общего назначения.

 СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией БД информационной системы. Они позволяют определить структуру создаваемой БД, инициализировать ее и произвести начальную загрузку данных. Системные механизмы выполняют также функции управления ресурсами среды хранения, обеспечения логической и физической независимости данных, предоставления доступа пользователям к БД, защиты логической целостности БД, обеспечения ее физической целостности – защиты от разрушений. Другая важная группа функций – управления полномочиями пользователей на доступ к БД, настройка на конкретные условия применения, организация параллельного доступа пользователей к базе данных в социальной пользовательской среде, поддержка деятельности системного персонала, ответственного за эксплуатацию БД.

 Для создания БД разработчик описывает ее логическую структуру, организацию в среде хранения, а также способы видения базы данных пользователями. При этом используются предоставляемые СУБД языковые средства определения данных, и система настраивается на работу с конкретной БД. Такие описания БД называются соответственно **схемой** (или логической схемой, или концептуальной схемой) БД, **схемой хранения** (или внутренней схемой) и **внешними схемами**.

 Обрабатывая схемы БД, СУБД создает пустую БД требуемой структуры – хранилище, которое можно далее наполнить данными о предметной области начать эксплуатировать для удовлетворения информационных потребностей пользователей.

 Принципиально важное свойство СУБД заключается в том, что она позволяет различать и поддерживать два независимых взгляда на БД – взгляд пользователя, воплощаемой в «**логическом**» представлении данных, и «взгляд» системы – «**физическое**» представление, характеризующее организацию хранимых данных. Пользователя не интересует при его работе с БД байты и биты, представляющие данные в среде хранения, их размещения в памяти, указателя, поддерживающие связи между структурными различными компонентами хранимых данных, выбранные методы доступа. В то же время эти факторы важны для выполнения функций управления данными самой СУБД.

 Обеспечение **логической независимости данных** – одна из важнейших функций СУБД, предоставляющая определенную степень свободы вариации «логического» представления БД без необходимости соответствующей модификации «физического» представления. Благодаря этому достигается возможность адаптации взгляда пользователя на БД к его реальным потребностям, конструирования различных «логических» взглядов на одну и ту же «физическую» БД, что весьма важно в социальной пользовательской среде.

 Под **«физической» независимостью данных** понимается способность СУБД предоставлять некоторую свободу модификации способов организации БД в среде хранения, не вызывая необходимости внесения соответствующих изменений в «логическое» представление. Благодаря этому вносить изменения в организацию хранимых данных, производить настройку системы с целью повышения ее эффективности, не затрагивая созданных прикладных программ, использующих базу данных. «Физическая» независимость данных реализуется в СУБД за счет тех же самых трансформационных механизмов архитектуры системы, которые обеспечивают «логическую» независимость данных.

 Поддержка логической целостности (непротиворечивости) базы данных – другая важная функция СУБД. В развитых системах ограничения целостности базы данных объявляются в схеме базы данных, и их проверка осуществляется при каждом обновлении объектов данных или связей между ними, являющихся аргументами таких ограничений.

Глава 2. Разработка базы данных по рынку бытовой химии.

Процесс разработки (проектирования) базы данных включает два этапа: разработку логической организации базы данных и создание ее на носителе. Логическая организация базы данных – это предоставление пользователя о предметной области, информация о которой должна храниться в базе данных. Под физической организацией базы данных понимается совокупность средств и методов размещения данных во внешней памяти и на их основе внутренняя модель данных. Внутренняя модель является средством отображения логической модели данных, показывает, каким образом записи размещаются в базе данных, как они упорядочиваются, как организуются связи, каким путем можно осуществить выборку и так далее.

 В настоящее время разработаны и используются на персональных компьютерах около двадцати систем управления базами данных. Они представляют пользователю удобные средства интерактивного взаимодействия с БД и имеют развитый язык программирования. Одной из самых популярных настольных программных СУБД является Microsoft Access.

 Одной из основных причин такой популярности Access заключается в том, что, является по сути настольной СУБД, это приложение вобрало в себя многие возможности систем управления реляционными базами данных архитектуры клиент-сервер, называемой также SQL базой данных. Несмотря на то, что, Access включают в себя сложные функции и может послужить прекрасным инструментом для профессионального разработчика приложений БД, его использование не должно вызвать проблем и у непрофессиональной пользователей и даже тех, кто раньше не работал с СУБД. Кнопки на панелях инструментов дублируют основные команды меню, расширенный набор мастеров и настроек управляет практически всеми параметрами создания и изменения объектов БД (таблиц, форм, отчетов, запросов и т.д.). С помощью ACCESS можно создавать многопользовательских приложений, в которых файлы базы данных являются разделяемыми ресурсами в локальной сети. В ACCESS реализованного доступа к объектам базы данных. Microsoft Access для хранения объектов БД имеет собственную уникальную структуру для хранения всех связанных таблиц, форм, отчетов, запросов и макрософт в одном файле. Также имеет возможность импорта и экспорта данных во многие широкие распространенные форматы БД, электронных таблиц и текстовых файлов. ACCESS позволяет связывать БД с внешними таблицами в форматах dBase, FoxPro, Paradox и работать с ними в исходном формате. Также Access можно использовать в качестве клиентской части архитектуры клиент-сервер, что обеспечивает применение Microsoft Access не только в качестве профессиональной системы управления базы данных, но и как мощное инструментальное средство для создания приложений клиент-сервер.

 База данных по бытовой технике города Улан-Удэ была разработана в программе Microsoft Access. Вся необходимая информация представлена в двух таблицах. Таблица базы данных – это совокупность сведений. Так, например, в таблице «торговые салоны» отображена информация о торговом салоне, адресе, телефоне (рис. 1), а в таблице «товары» - информация о предоставляемых торговым салоном товарах (рис. 2). Каждое поле предоставляет собой столбец таблицы и содержит определенную категорию информации. Каждая запись предоставляет собой строку таблицы и содержит информацию об определенном товаре. Можно создать взаимосвязи между таблицами (Рис. 10), вместо того, чтобы хранить всю информацию в одной большой таблице, избегая тем самым ненужного дублирования данных, экономии памяти компьютера, а также увеличение скорости и точности обработки информации. Так, например, каждая запись в таблице «все товары» содержит информацию о фирме и предоставляемых ею товарах (рис. 3). Запросы используются примерно также, как и таблицы. Вы можете открыть запрос и просмотреть набор данных в табличном представлении. При создании запроса указываются таблицы, из которых будет производится выборка данных, указываются поля таблицы, которые должны быть внесены в результат запроса, указано условие отбора данных. В данном случае при выборе запроса «поиск по салону» (рис. 4), указав название салона, вы получите интересующую вас информацию о товарах в этом салоне.

 При выборе запроса «поиск по товарам» (рис. 3) вы получите информацию о представляемых фирмой товарах.

 С помощью форм можно отображать данные, содержащиеся в таблицах или запросах, в еще более удобное для восприятия виде. При помощи форм можно добавлять в таблицы новые данные или изменять и удалять существующие. Форма может содержать рисунки, графики и другие объекты. Формы создаются из набора отдельных элементов, называемых элементами управления (графический объект для предоставления данных). Форму для предоставления данных Microsoft Access предоставляет автоматически. После ее заполнения можно просмотреть информацию полностью или частично при помощи процедур поиска, запроса, сортировки. Для просмотра необходимой информации с помощью форм в моей базе данных представлены 6 различных форм: кнопочная, все товары, все торговые салоны, поиск, поиск по салону, поиск по товару (рис. 5-10). При выборе той или иной формы, указав, если есть необходимость, название товара, вы получите интересующую вас информацию.

 **Заключение.**

 Таким образом, БД является важнейшей составной частью информационных систем, которые предназначены для хранения и обработки информации. Изначально такие системы существовали в письменном виде. Для этого использовались различные картотеки, папки, журналы, библиотечные каталоги. Развитие средств вычислительной техники обеспечило возможность для создания и широкого использования автоматизированных информационных систем. Разрабатываются информационные системы для обслуживания различных систем деятельности, системы управления хозяйственными и техническими объектами, модельные комплексы для научных исследований, системы автоматизации проектирования и производства, всевозможные тренажеры и обучающие системы. Современные информационные системы основаны на концепции интеграции данных, характеризующих большими объектами хранимых данных, сложной организацией, необходимостью удовлетворять разнообразные требования многочисленных пользователей. Для управления этими данными и обеспечения эффективности доступа к ним были созданы системы управления данными.

 Таким образом, СУБД называют программную систему, предназначенную для создания ЭВМ общей базы данных для множества приложений, поддержания ее в актуальном состоянии и обеспечения эффективности доступа пользователей к содержащимся в ней данным в рамках предоставленных им полномочий.

 Приведенная для примера база данных по бытовой химии города Улан-Удэ разработана в программе Microsoft Access, которая является одной из самых популярных среди настольных программных систем управления базами данных. Среди причин такой популярности следует отметить:

* высокую степень универсальности и продуманности интерфейса, который рассчитан на работу с пользователями самой различной квалификации. В частности, реализована система управления объектами базы данных, позволяющая гибко и оперативно переходить из режима конструктора в режим их непосредственной эксплуатации;
* глубоко развитые возможности интеграции с другими программными продуктами, входящими в состав Microsoft Office, а также с любыми программами продуктами, поддерживающими технологию OLE;
* богатый набор визуальных средств разработки.

 **Список литературы:**

* 1. С. Бобровский С.- П. 2001г. «DELPHI 5».
	2. А.Д. Хоменко «Основы современных компьютерных технологий». М. 2000г.
	3. А.Я.Архангальский «Программирование в Delphi 5» М. 2000г.
	4. Ю. Бекаревич, Н.Пушкина «MS ACCESS 2000 ЗА 30 занятий».
	5. С.Н.Кандзюба, В.Н.Громова «DELPHI 5».
	6. Марко Кэнту «DELPHI 5».
	7. В.Гофман, А.Хаменко «Работа с БД в DELPHI».
	8. К.Дэйт «Введени
	9. Введение в системы баз данных» К.2000г.
	10. СУБД Microsoft Access 2.0 “Шаг за шагом» М. 1995г.

## Министерство Культуры РФ

Восточно-Сибирская Академия Культуры и Искусств

**Кафедра** МиМИД

## Курсовая работа

 **Тема:** Разработка базы данных по товарам потребительского рынка бытовой технике.

 Выполнила студент 2 курса

 Очного отделения 425 группы

####  Специальность 071900 «Информационные

 Системы в социально-культурной сфере»

 Квалификация «Менеджер

 Информационных систем»

  **Перевалов Станислав Николаевич**

 Научный руководитель

####  Иванова Надежда Сергеевна

Работа сдана на рецензию «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2002г. Допущена к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2002г.

Защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Улан-Удэ

2002г.