Федеральное агентство по образованию

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.В. БАТАЕВ

ИНФОРМАТИКА. ТЕХНОЛОГИИ БАЗ ДАННЫХ В ИНФОРМАЦИОННЫХ ЭКОНОМИЧЕСКИХ СИСТЕМАХ.

Учебное пособие.

САНКТ-ПЕТЕРБУРГ

Издательство Политехнического университета

2006

УДК 336.761: 336.763 (075.8)

А.В. Батаев. Технологии баз данных в информационных экономических системах. Учебное пособие. – СПб, изд. СПГПУ, 2006, - 77 с.

Ответственный редактор – зав.кафедрой «Финансы и
денежное обращение», д-р. э. наук., проф. Э.А.Козловская

Пособие предназначено студентам специальности «Финансы и кредит» и слушателям по второму высшему образованию. Оно может быть использовано также студентами других экономических специальностей при разработке баз данных с использованием Microsoft Access.
Пособие содержит описание систем управления баз данных Microsoft Access, SQL MS Server, Oracle, Progress, а также описаны модели баз данных используемых на сегодняшний день. Приведены виды архитектур существующих баз данных с позиции сетевых технологий.

Рис. 22, таблиц 14 , библиогр. – 10 назв.

Учебное пособие разработано по заявке кафедры «Финансы и денежное обращение» СПбГПУ, которая обладает эксклюзивным правом на его распространение.

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета.

СОДЕРЖАНИЕ

Введение 5
1. Системы управления базами данных (СУБД) 6
1.1 Понятие базы данных (БД) 6
1.2. Виды моделей БД 6
1.2.1. Сетевые модели БД 6
1.2.2 Иерархические модели БД 7
1.2.3 Реляционные модели БД 8
1.2.4 Объектные модели БД 10
1.3 Интеграция неоднородных информационных ресурсов 10
1.4 Распределенная обработка данных в СУБД 11
1.5 Архитектуры СУБД 13
1.6 Основные функции системы управления базами данных 15
2. СУБД Microsoft Access 16
2.1 Microsoft Access как реляционная СУБД 16
2.1.1 Возможности Microsoft Access 16
2.1.2 Области применения Microsoft Access 17
2.1.3 Архитектура Microsoft Access 20
2.1.4 Разработка проекта приложения 21
2.2. Работа с данными в Microsoft Access 25
2.2.1 Построение баз данных в Microsoft Access 25
2.2.1.1 Создание таблиц 25
2.2.1.2 Определение полей 25
2.2.2 Задание условий на значения для полей 28
2.2.3 Создание первичного ключа и установление связей 29
2.2.4 Создание индексов 30
2.2.5 Ограничения для баз данных 31
2.2.6 Изменение проекта базы данных 32
2.2.7 Режим таблицы 33
2.2.8 Работа с гиперссылками 33
2.2.9 Сортировка и поиск данных 34
2.2.10 Работа с данными при помощи запросов 35
2.2.10.1 Выбор данных из одной таблицы 35
2.2.10.2. Многотабличные запросы 39
2.2.11 Ограничения при использовании запросов на выборку для обновления данных 41
2.2.12 Модификация данных с помощью запросов на изменение 41
2.2.13 Импорт, экспорт и связывание данных 42
2.2.14. Язык SQL 44
2.3. Интерфейс пользователя в базах данных Microsoft Access 45
2.3.1. Использование форм 45
2.3.1.1. Основные сведения о формах 45
2.3.1.2 Элементы управления форм 46

2.3.2 Построение форм **48**
2.3.2.1 Формы и объектно–ориентированное программирование **48**
2.3.2.2. Инструменты построения форм **49**
2.3.3 Главная (основная) кнопочная форма **51**
2.3.4 Разработка отчетов **52**
2.4. Создание приложений **53**
2.4.1 Общие сведения о макросах **53**
2.4.2 Основные понятия процедур VBA для приложений **55**
2.4.3 Язык VBA **55**
2.4.4. Модули **56**
3. Виды корпоративных СУБД **59**
3.1 СУБД MS SQL Server **59**
3.1.1 Различие между MA и MS SQL Server **59**
3.1.2 Объекты MS SQL Server **60**
3.2 СУБД Oracle **62**
3.2.1 История развития СУБД Oracle **62**
3.2.2 Инструменты и технологии, реализованные в СУБД Oracle **62**
3.2.3 Структура базы данных Oracle **67**
3.2.4 Основные объекты и термины базы данных Oracle **68**
3.3 СУБД Progress **71**
3.3.1 Архитектура СУБД Progress **71**
3.3.2 Основные компоненты СУБД Progress **72**
3.3.3 Возможности СУБД Progress **73**
Список литературы **77**

Введение

Развитие технологий создания баз данных насчитывает несколько десятков лет. За свою историю базы данных прошли несколько этапов развития, и теперь их создание базируется на математической теории реляционной алгебры.

На сегодняшний день трудно представить хотя бы одну отрасль экономики, которая бы обходилась без построения баз данных. Они присутствуют как у отдельных пользователей, так и на крупных предприятиях и фирмах.

На данный момент на рынке программного обеспечения присутствуют различные программы, позволяющие писать базы данных любой степени сложности. Среди них можно выделить DBASE, PARADOX, FOXPRO, MICROSOFT ACCESS.

Наибольшей популярностью пользуется программа MICROSOFT ACCESS (созданная в 1992 году), если судить по количеству проданных копий. MICROSOFT ACCESS является удачной системой управления реляционными базами данных. MICROSOFT ACCESS позволяет легко объединять связанную информацию. Кроме этого, он является дополнением к другим работающим с базами данных программным продуктам, поскольку предоставляет широкие возможности для работы с данными из других источников, включая базы данных SQL. Полностью поддерживая технологию OLE, MICROSOFT ACCESS может выступать в качестве клиента или сервера по отношению к другим приложениям, таким как MICROSOFT WORD, MICROSOFT EXCEL, MICROSOFT POWER POINT и MICROSOFT OUTLOOK.

MICROSOFT ACCESS обладает развитой системой разработки приложений, позволяющей быстро создавать необходимые приложения для широкого спектра источников данных. Действительно, создание простого приложения в MICROSOFT ACCESS не представляет никаких трудностей. Достаточно определить формы и отчеты, исходя из имеющихся данных, и связать их в приложении с помощью нескольких простых инструкций языка VISUAL BACIC. При этом нет необходимости в написании программы в классическом значении этого термина.

Для написания сложных приложений, требующих нестандартных подходов, в распоряжении разработчиков предоставлен мощный аппарат языка запросов SQL и соответственно языка VISUAL BASIC. Такой подход, сочетающий возможность построения с помощью стандартных средств и языков программирования высокого уровня, позволяет использовать MICROSOFT ACCESS как на предприятиях малого бизнеса, где он в сочетании с MICROSOFT SQL SERVER представляет идеальную среду для быстрой и эффективной разработки новых приложений, так и для крупного бизнеса. На крупных предприятиях MICROSOFT ACCESS является средством, позволяющим связать в одном приложении данные, хранящиеся на главных серверах корпорации и на персональных компьютерах, в разных форматах.

Для создания баз данных корпораций используются корпоративные (профессиональные) СУБД, позволяющие создавать базы данных на основе технологии хранилищ данных. Среди них можно выделить такие СУБД, как MICROSOFT SQL SERVER, ORACLE, PROGRESS.

Если две первые СУБД являются лидерами среди приложений для баз данных, то СУБД PROGRESS менее заметен на рынке технологий баз данных, но тем не менее используется при построении интегрированных банковских систем, используемых в российских банках.

1. Системы управления базами данных.

1.1. Понятие базы данных.

При больших объемах информации возникает две проблемы, которые необходимо решить:
1) упорядочение информации
2) обеспечение наибольшей скорости поиска в ней.

В компьютере для хранения и использования информации создаются специальные хранилища – базы данных.

Технологии баз данных одна из наиболее востребованных технологий в практической разработке информационных экономических систем, сформирована широкая сфера самых разнообразных приложений систем баз данных.

База данных (БД) – это совокупность взаимосвязанных, хранящихся вместе сведениях о различных сущностях одной предметной области (реальных объектах, процессах, явлениях или событиях), обеспечивающая наличие такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений или пользователей.
Также можно определить базу данных – как набор записей и файлов, организованных определенным образом, предназначенный для поиска информации. Например, одним из типов базы данных могут быть документы, набранные с помощью текстовых редакторов и сгруппированные по темам; другим типом – файлы электронных таблиц, объединяемые по характеру использования.

Одним из основных свойств баз данных можно считать независимость данных от использующих их прикладных программ. Под независимостью данных подразумевается то, что изменения в данных не приводит к изменению программ. Разработка программ длительный, трудоемкий и дорогостоящий процесс, поэтому при возникновении потребности модифицировать структуру данных, нет необходимости сохранять уже созданные прикладные программы.

Системы управления базами данных (СУБД) - это программные средства, предназначенные для ввода, наполнения, удаления, фильтрации и поиска данных.

Фундаментом технологий баз данных является модель данных, на которой базируется конкретная СУБД. Модель описывает набор понятий и признаков, которыми должна обладать конкретная СУБД и управляемые ими базы данных, если они основываются на этой модели. Наличие такой модели позволяет сравнивать конкретные реализации СУБД и оценивать их соответствие модели.

1.2. Виды моделей БД

1.2.1. Сетевые модели БД

История создания и развития СУБД насчитывает около сорока лет.

Современные СУБД основываются на использовании моделей данных (МД), позволяющих описывать объекты предметных областей и взаимосвязи между ними. Существуют три основные МД и их комбинации, на которых основываются СУБД: реляционная модель данных (РМД), сетевая модель данных (СМД), иерархическая модель данных (ИМД).

Основное различие между этими моделями данных состоит в способах описания взаимодействий между объектами и атрибутами. Взаимосвязь выражает отношение между множествами данных. Используются взаимосвязи «один к одному», «один ко многим» и «многие ко многим». «Один к одному» - это взаимно однозначное соответствие, которое устанавливается между одним объектом и одним атрибутом. Например, в определенный момент времени в одной ЭВМ используется один определенный процессор. Номеру выбранной ЭВМ соответствует номер

выбранного процессора. «Один ко многим»- одно-многозначное соответствие, которое устанавливается между одним объектом и многими атрибутами. Например, один пользователь для решения различных задач использует различные языки программирования. «Многие ко многим» - это соответствие между многими объектами и многими атрибутами. Например, на множестве ЭВМ может одновременно работать множество пользователей. Взаимосвязи между объектами и атрибутами удобно представлять в виде графов и гиперграфов.

Сетевые модели данных базируются на табличных и графовых представлениях: вершинам графа обычно сопоставляются некоторые типы сущности, которые представляются таблицами, а дугам - типы связей. Наиболее развитой сетевой моделью данных являлась модель, предложенная в апреле 1971 г. рабочей группой по базам данных Ассоциации по языкам систем обработки данных (CODASYL), спецификации которой впоследствии неоднократно пересматривались. В СМД элементарные данные и отношения между ними представляются в виде ориентированной сети (вершины - данные, дуги - отношения).

Рассмотрим «классическую» сетевую модель данных CODASYL. Основные «строительные блоки» структуры сетевой базы данных - тип записи и тип набора. Тип записи представляет собой множество записей, обладающих структурой и другими свойствами, специфицированными в описании данного типа записей в схеме базы данных для всех записей этого типа.

Запись - совокупность логически связанных полей, характеризуется именем и полями, входящими в нее. Полем называется единая неделимая единица информации, которая характеризуется идентификатором, типом и размером.

Помещенная в базу данных запись может существовать в ней не только самостоятельно, но и являться одновременно детальной или главной записью каких-либо наборов в зависимости от того, описан ли ее тип в схеме базы данных как тип главной записи или детальной записи каких-либо типов наборов.

Тип набора сетевой модели представляет собой множество наборов, обладающих структурой и другими свойствами, специфицированными в схеме базы данных для этого типа набора. Наборы СМД служат для представления отношений вида 1: n между главными записями и детальными записями одного или нескольких типов.

Каждый экземпляр набора состоит из одного экземпляра записи, называемой главной записью набора, и в общем случае динамически изменяющегося при обновлениях базы данных множества записей, называемых детальными записями набора.

Главная и детальная записи данного набора связываются с помощью указателей в цепь и образуют упорядоченную последовательность. Могут быть предусмотрены дополнительные указатели, связывающие каждую детальную запись набора непосредственно с ее главной записью, а также указатели, обеспечивающие обход записей набора в обратном направлении. Типы главных и детальных записей наборов данного типа объявляются в описании этого типа набора в схеме. Каждый экземпляр главной записи набора, появляясь в базе данных, порождает экземпляр набора этого типа.

Главные и детальные записи одних наборов могут быть одновременно главными и/или детальными записями других наборов того же самого или иных типов. Таким образом, из записей базы данных и наборов может быть сконструирована база данных произвольно сложной структуры.

1.2.2 Иерархические модели БД.

Иерархическая модель данных (ИМД) основана на понятии деревьев, состоящих из вершин и ребер. Вершина дерева ставится в соответствие совокупности атрибутов данных, характеризующих некоторый объект. Вершины и ребра дерева как бы образуют иерархическую древовидную структуру, состоящую из n уровней.

Первую вершину называют корневой вершиной. Она удовлетворяет условиям:

1. Иерархия начинается с корневой вершины.

2. Каждая вершина соответствует одному или нескольким атрибутам.

3. На уровнях с большим номером находятся зависимые вершины. Вершина предшествующего уровня является начальной для новых зависимых вершин.

4. Каждая вершина, находящаяся на уровне i, соединена с одной и только одной вершиной уровня i-1, за исключением корневой вершины.

5. Корневая вершина может быть связана с одной или несколькими зависимыми вершинами.

6. Доступ к каждой вершине происходит через корневую вершину по единственному пути.

7. Существует произвольное количество вершин каждого уровня.

Иерархическая модель данных состоит из нескольких деревьев, т. е. является лесом. Каждая корневая вершина образует начало записи логической базы данных. В ИМД вершины, находящиеся на уровне i, называют порожденными вершинами на уровне i-1.

Операции в ИМД имеют аналогичный СМД «позаписный» характер. Аппарат перемещения по структуре в графовых моделях служит для установки тех объектов данных, к которым будет применяться очередная операция манипулирования данными. Такие объекты называются текущими. Механизмы доступа к данным и перемещения по структуре данных в таких моделях достаточно сложны и существенным образом опираются на концепцию текущего состояния механизма доступа.

Основные достоинства ИМД: простота построения и использования, обеспечение определенного уровня независимости данных, простота оценки операционных характеристик. Основные недостатки: отношение «многие ко многим» реализуется очень сложно, дает громоздкую структуру и требует хранения избыточных данных, что особенно нежелательно на физическом уровне, иерархическая упорядоченность усложняет операции удаления и включения, доступ к любой вершине возможен только через корневую, что увеличивает время доступа.

1.2.3 Реляционные модели БД.

Сетевые и иерархические модели в настоящее время считаются устаревшими, но существует множество баз данных созданных на их основе и требующих поддержания их работы.

Одним из крупнейших достижений в этой области является создание реляционной модели данных и базирующейся на ней теории реляционных баз данных, которая позволила получить важные результаты для развития теории баз данных. Как отмечают многие исследователи, своим успехом реляционная модель данных во многом обязана, в первую очередь тому, что опиралась на строгий математический аппарат теории множеств, отношений и логики первого порядка. Разработчики любой конкретной реляционной системы считали своим долгом показать соответствие своей конкретной модели данных общей реляционной модели, которая выступала в качестве меры "реляционности" системы. Существует широкий спектр реляционных СУБД для приложений различного масштаба. Разработан международный стандарт языка запросов SQL, ставший универсальным интерфейсом коммерческих реляционных СУБД. По оценкам специалистов, примерно 99% мирового рынка баз данных занимают в настоящий момент реляционные СУБД. Несмотря на то, что подавляющее большинство приложений базируется на реляционной технологии, их роль начинает ослабевать.

В основе реляционной модели лежит математическое понятие теоретико-множественного отношения, которое представляет собой подмножество декартова произведения списка доменов. Домен- это просто множество значений. Рассмотрим пример реляционной базы данных.

Для решения сложных задач одного файла с данными оказывается недостаточно. Например, если необходимо составить заявку на товары, в которой должны присутствовать

наименования товаров и имена поставщиков, и все это хранить в одной таблице, то в каждой записи наряду с наименованием товара следовало бы предусмотреть поле для адреса поставщика. Если один и тот же поставщик имеет несколько видов товаров, то многие записи будут хранить дублирующую информацию. Запись – это строка в таблице БД, а поле – столбец. Пример организации базы данных такого типа показан в таблице 1.

Таблица 1. Заявка на товары.

Товар	Цена, $	Адрес поставщика
Компьютер	900	Санкт - Петербург
Принтер	300	Санкт - Петербург
Факс	600	Москва

Данный способ организации базы данных имеет следующие недостатки:

1) При добавлении в базу данных информации о новом товаре уже известного поставщика придется опять вводить его адрес в создаваемую запись.
2) Если адрес поставщика изменится, необходимо будет проверить и обновить все записи, хранящие данный адрес.
3) При хранении в каждой записи полного адреса поставщика увеличивается объем используемой памяти и затраты времени на управление данными, а поиск записей, соответственно, замедляется.
4) Повышается вероятность появления ошибки при вводе и обработке данных.

Поэтому все современные системы основаны на реляционной модели управления базами данных. В такой базе данных каждая запись содержит информацию, относящуюся только к одному конкретному объекту, кроме того, с данными двух типов, например, со сведениями о товарах и поставщиках можно работать как с единым целым, основываясь на значениях, связанных между собой данных. Таким образом, преимущество ее заключается в том, что для каждого самостоятельного набора данных создается своя собственная таблица. В выше приведенном примере это была бы отдельная таблица наименований товаров и отдельная таблица поставщиков. В таблице наименований товаров адрес поставщика указывается в виде ссылки, номера записи таблицы поставщиков, содержащей реальный адрес. Эта взаимосвязь отражена в таблице 2.

Таблица 2. Взаимосвязь таблиц в реляционной базе данных.

Наименование товара	Цена	Кол - во	Адрес поставщика
1…			2
2…			
3…			2

Наименование поставщика	Адрес	ФИО руководителя
1…		
2…		
3…		

Преимущества реляционной модели:

1) Централизованное хранение информации о поставщиках.
2) Меньшие затраты на ввод данных.
3) Высокая скорость доступа к информации вследствие меньших затрат памяти.
4) Высокая устойчивость к ошибкам.

1.2.4 Объектные модели БД.

Вместе с тем в последние годы четко обозначилась тенденция развития СУБД в объектном направлении. Объектная (объектно-ориентированная) модель не противоречит реляционной модели данных, а дополняет и развивает последнюю (точнее сказать — реляционная модель является частным случаем объектной формы представления данных). Однако, трудности развитого математического аппарата, на который могла бы опираться общая объектная модель данных, не существует, как нет и признанной базовой объектной модели. С другой стороны, некоторые авторы утверждают, что общая объектная модель данных в классическом смысле и не может быть определена по причине непригодности классического понятия модели данных к парадигме объектной ориентированности.

Парадигма - это пространство идей и законы движения в этом пространстве. В рамках парадигмы определены аксиомы, на которых выстраивается своя логика. Решения, вырабатываемые в рамках парадигмы, непротиворечивы и логичны.

Преимуществами объектных СУБД можно считать:

- объектные СУБД – открытые системы. Несложно добавить новый тип данных;
- объектные СУБД быстрее, чем реляционные, если в программе многократно осуществляется переход от объекта к объекту по ссылке. Поскольку ссылка на объект есть идентификатор, однозначно определяющий его расположение в базе, то переход по такой ссылке происходит быстрее, чем ссылка в реляционной модели. Объектные СУБД (ОСУБД) устраняют необходимость в языке запросов

Традиционные области применения ОСУБД – системы автоматизированного проектирования, моделирование, мультимедиа. ОСУБД широко используются в телекоммуникациях, различных аспектах автоматизации предприятия, издательском деле.

1.3 Интеграция неоднородных информационных ресурсов.

Информационная неоднородность ресурсов заключается в разнообразии понятий, словарей; отображаемых реальных объектов; правил, определяющих адекватность моделируемых объектов реальности; видов данных, способов их сбора и обработки; интерфейсов пользователей и т.д.

Реализационная неоднородность источников проявляется в использовании разнообразных компьютерных платформ, средств управления базами данных, моделей данных и знаний, средств программирования, операционных систем, и т.п. Системы, обеспечивающие совместимость различных компонентов называются интероперабельными системами.

Традиционные системы баз данных, используемые в информационных системах для сопровождения бизнес - процессов поддерживают большие объемы информации с помощью технологий оперативной обработки транзакций – OLTP. В OLTP-технологии обрабатывается детализированные данные, главные свойства данных здесь, их полнота и актуальность.

Под транзакцией понимается неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации) такая, что либо результаты всех операторов, входящих в транзакцию, отображаются в БД, либо воздействие всех этих операторов полностью отсутствует. Лозунг транзакции – «Все или ничего». Поддержание

механизма транзакций - показатель уровня развитости СУБД. Корректный механизм поддержания транзакций одновременно является основой обеспечения целостности баз данных.

Для поддержки принятия решений нужны другие технологии. Необходимо объединять данные из различных источников (как из корпоративной информационной системы, так и из внешней среды), накапливать данные, делая их срезы во времени. Анализ таких данных позволяет оценивать состояние и динамику развития организации, делать обоснованные прогнозы и принимать обоснованные решения. Программные продукты, необходимые для обеспечения управленческих решений, должны обеспечивать хранение больших объемов данных, эффективный доступ к ним, а так же располагать развитыми средствами анализа данных и представления результатов в удобной для специалистов и руководства форме. Информационная технология, которая предоставляет руководителям различного уровня возможность получения необходимой информации для принятия управленческих, финансовых и кадровых решений называется OLAP (On-Line Analytical Processing- оперативной аналитической обработкой) - технологией.

OLAP – технологии базируются на технологиях хранилищ данных (Data warehouses). Хранилище данных обеспечивает накопление с течением времени данных для содействия в принятии решений. Хранилище - это репозитарий (склад) информации содержащий объединенные, проверенные данные, отражающие работу организации за длительный период. Объемы данных в хранилищах в несколько раз превосходят объемы данных в OLTP-системах.

Хранилища данных отличаются от баз данных или систем оперативной обработки транзакций (OLTP-систем) своим назначением и устройством:

- хранилище содержит данные, позволяющие проводить анализ деловых операций;
- хранилища обычно представляют собой системы, доступные только для чтения;
- в хранилищах накапливаются данные, не меняющиеся со временем и избавленные от ошибок.

Различия между OLAP и OLTP системами представлено в таблице 3.

Из-за большого объема данных в хранилищах, одной из основных проблем создания хранилищ является обеспечение высокой производительности обработки запросов. Запросы в хранилище отличаются высоким уровнем сложности.

Создание хранилищ данных – трудоемкий и длительный процесс. Наряду с хранилищами данных существуют и часто используются компаниями витрины данных (Data Mart), называемые также киосками данных. Такие системы создаются для отдельных подразделений компаний или для обеспечения отдельных видов деятельности. Объемы данных и требования к вычислительным ресурсам в витринах данных существенно меньше по сравнению с хранилищами. Витрины данных могут строиться как независимо, так и на основе хранилищ данных компании. Хранилища данных имеют двухуровневую или трехуровневую архитектуру. В двухуровневых хранилищах на верхнем уровне поддерживается объединенная информация. На нижнем уровне - различные источники баз данных. В трехуровневой архитектуре предусматривается поддержка витрин данных для отдельных подразделений компании над ее единым хранилищем.

1.4 Распределенная обработка данных в СУБД.

В современном бизнесе очень часто возникает необходимость предоставить доступ к одним и тем же данным группам пользователей, территориально удаленным друг от друга. В качестве примера можно привести банк, имеющий несколько отделений. Эти отделения могут находиться в разных городах, странах или даже на разных континентах, тем не менее необходимо организовать обработку финансовых транзакций (перемещение денег по счетам) между отделениями. Результаты финансовых операций должны быть видны одновременно во всех отделениях.

Таблица 3. Различие между OLAP и OLTP системами.

Характеристика базы данных	База данных OLTP (оперативная обработка транзакций)	База данных OLAP (хранилище данных, деловой анализ)
Содержимое	Текущие данные	Данные, накопленные за долгий период времени
Структура данных	Структура таблиц соответствует структуре транзакций	Структура таблиц понятна и удобна для написания запросов (кубы фактов - схема "звезда")
Типичный размер таблиц	Тысячи строк	Миллионы строк
Схема доступа	Предопределена для каждого типа обрабатываемых транзакций	Произвольная; зависит от того, какая именно задача стоит перед пользователем в данный момент и какие сведения нужны для ее решения
Количество строк, к которым обращается один запрос	Десятки	от тысяч до миллионов
С какими данными работает приложение	С отдельными строками	С группами строк (итоговые запросы)
Интенсивность обращений к базе данных	Большое количество бизнес - транзакций в минуту или в секунду	На выполнение запросов требуется время: минуты или даже часы
Тип доступа	Выборка, вставка и обновление	Выборка данных (почти 100 % операций)
Чем определяется производительность	Время выполнения транзакции	Время выполнения запроса

Существуют два подхода к организации обработки распределенных данных:

- технология распределенной базы данных. Такая база включает фрагменты данных, расположенные на различных узлах сети. С точки зрения пользователей она выглядит так, как

будто все данные хранятся в одном месте. Естественно, такая схема предъявляет жесткие требования к производительности и надежности каналов связи;

- технология тиражирования**.** В этом случае в каждом узле сети дублируются данные всех компьютеров. При этом передаются только операции изменения данных, а не сами данные. Передача может быть асинхронной (неодновременной для разных узлов), данные располагаются там, где обрабатываются.

Использование технологии тиражирования позволяет снизить требования к пропускной способности каналов связи. При выходе из строя линии связи какого-либо компьютера, пользователи других узлов могут продолжать работу. Однако при этом допускается неодинаковое состояние базы данных для различных пользователей в один и тот же момент времени. Следовательно, невозможно исключить конфликты между двумя копиями одной и той же записи.

В основе распределенной обработки лежит запрос к собственной локальной БД или удаленной (БД сервера). Запрос - формализованное задание на поиск и обработку информации. Удаленный запрос – единичный запрос к одному серверу. Несколько удаленных запросов к одному серверу объединяются в удаленную транзакцию. Если отдельные запросы транзакции обрабатываются различными серверами, то транзакция называется распределенной.

Распределенная база данных и распределенная обработка не синонимы. Распределенная БД размещается на нескольких серверах, работа с ней, для получения доступа к удаленным данным, требует использования сетевой СУБД. При распределенной обработке один запрос транзакции обрабатывается одним сервером. Распределенная СУБД позволяет обрабатывать один запрос несколькими БД. Такой запрос называется распределенным.

1.5 Архитектуры СУБД.

По способу организации взаимодействия с базой данных через сеть, СУБД делят на:

1) СУБД с централизованной архитектурой.
2) СУБД с архитектурой файл-сервер.
3) СУБД с архитектурой клиент-сервер.
4) СУБД с трехуровневой архитектурой: клиент - сервер приложений - сервер базы данных.

СУБД с централизованной архитектурой. СУБД и сама база данных размещается и функционирует на центральном миникомпьютере (мэйнфрейме), а пользователи получают доступ к базе данных при помощи обычных терминалов - компьютер рассматривается просто как устройство ввода и отображения информации: на мэйнфрейм передаются нажатия клавиш, в обратном направлении передаются данные, отображаемые непосредственно на мониторе пользователя. Примерами СУБД с централизованной архитектурой являются ранние версии СУБД OB2, ранние версии СУБД Oracle и IngresB.

В СУБД с архитектурой файл-сервер база данных хранится на сервере, а копии СУБД устанавливаются на компьютерах пользователей. Файл базы данных, находящийся на сервере, совместно используется всеми пользователями одновременно, при помощи сетевого программного обеспечения и самой операционной системы. Ярким примером такой архитектуры является СУБД MS Access: копии СУБД установлены на компьютере каждого пользователя, а сам файл базы данных находится на сервере в сетевой папке. Архитектура файл-сервер позволяет добиться приемлемой производительности, т.к. в распоряжении каждой копии СУБД находятся все ресурсы компьютера пользователя. С другой стороны, производительность такой схемы для каждого пользователя, напрямую зависит от характеристик компьютера пользователя. Кроме того, такая схема работы значительно загружает сеть. Допустим, что пользователю необходимо отобрать строки таблицы с товарами, по которым объем продаж не превышает 100 тыс. руб. Поскольку строки в таблице не упорядочены, то скорее всего по сети будут переданы все строки

таблицы, из которых СУБД уже "на месте" (на компьютере пользователя) отберет нужные. Очевидно, что такая схема нерациональна при больших объемах обрабатываемой информации или большом числе пользователей базы данных. Поэтому, для таких БД целесообразнее применять архитектуру клиент-сервер.

При архитектуре клиент-сервер база данных хранится на сервере, а СУБД подразделяется на две части: клиентскую и серверную. Клиентская часть СУБД выполняется на стороне клиента и обеспечивает интерактивное взаимодействие с пользователем и формирование запросов к базе данных (на языке SQL). Серверная часть работает на сервере и взаимодействует с базой данных, обеспечивая выполнение запросов клиентской части, т.е., если провести аналогию с рассмотренным выше примером, то клиентская часть сформирует и отправит серверной части запрос "Отбери для меня строки таблицы с товарами, по которым объем продаж не превышает 100 тыс. рублей", серверная часть выполнит данный запрос и отошлет клиентской части только те строки, которые необходимо, не передавая по сети все строки таблицы. Большинство современных СУБД реализованы по архитектуре клиент-сервер: Oracle, MS SQL Server, PostgreSQL, MySQL, Infonnix, OB2 и др.

Однако и архитектура клиент-сервер не лишена недостатков. Если деловая логика взаимодействия с базой данных (логика, определяющаяся порядком работы предприятия: какие таблицы и в каком порядке заполнять, что делать при добавлении нового сотрудника и т.д.) изменяется, то приходится заново переписывать клиентские программы (вводить новые формы, менять порядок их заполнения и т.д.). Если изменения происходят слишком часто, а количество рабочих мест велико, то постоянная переустановка программного обеспечения (которая, кстати, должна осуществляться достаточно быстро) становится серьезной проблемой. В таких случаях, следует переходить к трехуровневой архитектуре: клиент – сервер приложений - сервер базы данных. При трехуровневой архитектуре в функции клиентской части входит только интерактивное взаимодействие с пользователем, а вся деловая логика вынесена на сервер приложений, который собственно и обеспечивает формирование запросов к базе данных, передаваемых на выполнение серверу базы данных. "Тонкий клиент" находится на компьютере пользователя и чаще всего представляет из себя Web-броузер (например, Intemet Explorer) с применением в соответствующей HTML-странице средств Java или компонентов ActiveX. Сервер приложений находится на сервере и может являться специализированной программой (например, Oracle Fonns Server) или обычным Web-сервером, вызывающим для обработки HTTP-запроса внешнюю программу через интерфейс CGI (см. рис. 1).

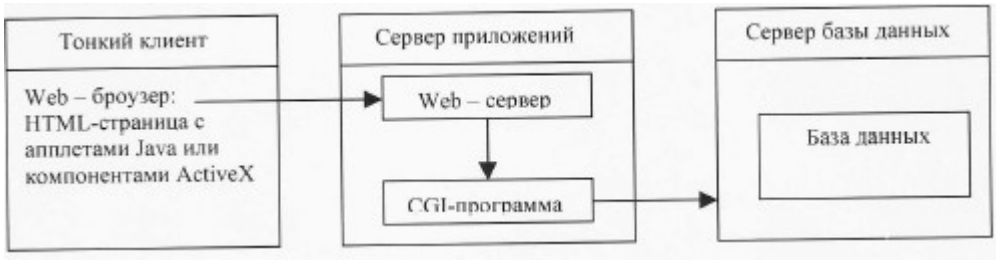


Рис. 1 Трехуровневая архитектура.

Преимущества трехуровневой архитектуры очевидны: при необходимости изменений в деловой логике, изменения вносятся только один раз - на сервере приложений. Изменять или переустанавливать клиентские программы нет никакой необходимости.

1.6 Основные функции системы управления базами данных.

При значительном увеличении объема информации и необходимости отслеживания связей между файлами, при возникновении необходимости подтверждения правильности ввода информации, возникает необходимость создания системы управления базами данных (СУБД).
СУБД – это совокупность набора данных (баз данных) и программ на обслуживание выполняющей управление БД (упорядочение, сортировка и т.д.).
Создание баз данных производится в несколько этапов.

1) Первый этап заключается в определении структуры базы данных, на нем устанавливается из каких полей, состоит отдельная запись базы данных и задается тип каждого поля. Например, бывают текстовые и числовые поля.
2) После разработки структуры базы данных осуществляется ввод данных. Это второй этап. Использование специальных форм упрощает как ввод, так и вывод данных - отдельных записей.
3) Третий этап – поиск информации в базе данных. Осуществляется на основе критериев поиска, который применяется к содержимому определенных полей базы данных. Совокупность критериев поиска предназначенных для отбора информации из набора данных называется запросом. В результате запроса отображаются только те данные, которые соответствуют заданным критериям.
4) Четвертый этап – оформление записей, полученных в результате запроса.

На основе этапов разработки СУБД можно сформулировать основные функции СУБД:

1) Определение данных: можно определить, какие сведения хранятся в базе данных, их типы и взаимосвязи друг с другом.
2) Обработка данных. Данные можно обрабатывать самыми различными способами: фильтровать, сортировать, выбирать итоговые значения, а также изменять копировать и удалять.
3) Управление данными: можно определить, какому пользователю или группе пользователей разрешен доступ к тем или иным данным, определить порядок изменения и удаления данных для каждого из пользователей.

2. СУБД Microsoft Access

2.1. Microsoft Access как реляционная СУБД

2.1.1. Возможности Microsoft Access

Определение данных и их хранение.

Microsoft Access (MA) предоставляет максимальную свободу при выборе типов данных, всего их насчитывается девять. С помощью Microsoft Access можно задавать форматы хранения данных, в частности длину строки, точность представления чисел и т.д. Чтобы быть уверенным в том, что в базе данных хранятся только правильные значения, можно задать условия на значения различной степени сложности. Кроме этого Microsoft Access проверяет правильность отношений между таблицами, т.к. Microsoft Access является приложением Windows, то в нем можно использовать возможности механизмов DDE и OLE. Механизм DDE позволяет выполнять обмен данными между Microsoft Access и любым другим приложением, поддерживающим механизм DDE. Механизм OLE позволяет устанавливать связи с объектами других приложений или внедрять объекты в базу данных Microsoft Access. Это могут быть рисунки, диаграммы, электронные таблицы и т.д. Кроме этого Microsoft Access воспринимает множество самых разнообразных форматов данных, включая файловые структуры других СУБД, в частности из баз данных « Paradox », « dBase », «Microsoft Fox Pro », а также осуществляет экспорт и импорт данных из текстовых файлов и электронных таблиц.

Обработка данных.

СУБД предоставляет разнообразные средства для работы с данными. Например, можно производить поиск любой степени сложности, как в отдельной таблице, так и в нескольких связанных таблицах или файлах. Кроме этого можно производить обновление данных в отдельном поле или отдельной записи. Для чтения или изменения данных можно создать процедуры, выполняющие функции СУБД. В Microsoft Access для обработки данных таблиц используется язык SQL (структурированный язык запросов). С помощью этого языка можно определить подмножество данных из одной или нескольких таблиц, соответствующих определенному критерию. При любой обработке данных из нескольких таблиц Microsoft Access использует заданные связи между таблицами, которые можно установить без знания языка SQL. Microsoft Access предоставляет средства графического построения запросов – так называемый запрос по образцу, используемый для задания данных, необходимых для решения конкретной задачи.

Управление данными.

При необходимости коллективного использования информации одной из функций СУБД становится защита информации от несанкционированного доступа. Задача защиты заключается в том, что права для работы в СУБД могут получать определенные пользователи или группы пользователей, которые имеют право того или иного вида доступа к информации. Предназначенные для коллективного пользования СУБД должны иметь средства, не позволяющие нескольким людям одновременно изменять одни и те же данные. Наиболее совершенные системы позволяют помимо этого группировать вносимые изменения

(последовательность таких изменений называют транзакциями) таким образом, что либо ни одно из них не будет сохранено, либо будут сохранены все. Microsoft Access спроектирован таким образом, что он может быть использован как в качестве самостоятельной СУБД на отдельной рабочей станции (компьютере), так и в сети в режиме « клиент- сервер » (рис. 2).

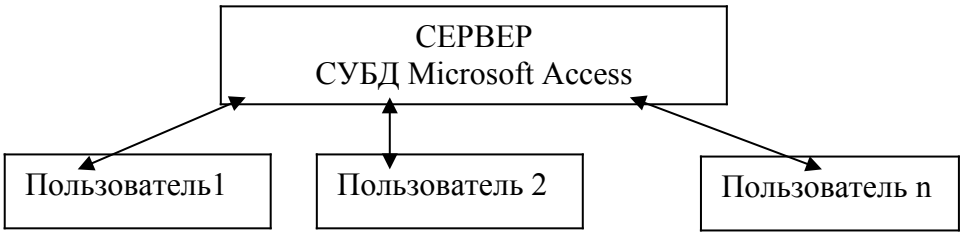


Рис. 2. Система связи «клиент – сервер».

В Microsoft Access предусмотрены средства защиты и обеспечения целостности данных. Можно заранее указать какие пользователи или группы пользователей обладают правом доступа к таблицам, формулам и запросам. В Microsoft Access применяется механизм автоматической блокировки для избежания одновременного изменения объекта несколькими пользователями. При работе с присоединенными таблицами из других баз данных « Paradox », « dBase », Microsoft Access распознает системы блокировки этих СУБД и подчиняется их требованиям.

Microsoft Access как средство разработки приложений.

При создании достаточно сложных приложений, работающих с базами данных, необходимы: реляционная СУБД и система разработки приложений, помогающая автоматизировать решение основных задач. Практически все существующие СУБД имеют средства разработки приложений, которые могут использоваться при создании процедур для управления и обработки данных. Но при этом следует отметить, что многие системы разработки приложений требуют знания языков программирования (например, « СИ »). В Microsoft Access существуют средства, позволяющие проектировать и создавать приложения без знания языка программирования. Работа в Microsoft Access начинается с определения таблиц и полей, предназначенных для хранения данных. После этого с помощью форм, отчетов, макросов и модулей можно определить действия над этими данными. Формы и отчеты можно использовать для задания форматов вывода данных на экран и дополнительных вычислений. Для автоматизации операций в простом приложении достаточно макросов (макрокоманд), позволяющих связать формы и отчеты между собой. При разработке очень сложных приложений можно использовать модули обработки событий для форм и отчетов или описывать функции с помощью языка программирования VBA (Visual Basic).

2.1.2 Области применения Microsoft Access.

В малом бизнесе.

На малом предприятии средства Microsoft Access могут использоваться для управления необходимыми для работы данными. В силу того, что средства разработки приложений в Microsoft Access довольно просты, то создание необходимых приложений или адаптация чужих для своих нужд является несложным процессом.

Области применения Microsoft Access в малом бизнесе:

1) Бухгалтерский учет.
2) Ввод заказов.
3) Ведение информации о клиентах.
4) Ведение информации о деловых контактах.

Для разработки прикладных программ на персональных компьютерах по заказу.

При нынешней конкуренции на рынке прикладного программного обеспечения побеждает тот разработчик, который быстро и за невысокую плату может предоставить пользователю нужное приложение. Для разработчиков прикладных программ Microsoft Access позволяет создавать законченные приложения для заказчика с помощью определенного набора запросов, форм и отчетов в очень краткие сроки. Если же клиент предъявляет какие – то особые требования, то в этом случае можно использовать возможности предоставляемые языком VBA.Для разработчиков прикладных программ в определенной области Microsoft Access позволяет строить ядро приложения и настраивать его в соответствии с потребностями конкретного заказчика. Кроме того, можно создавать дополнительные встраиваемые средства. При разработке конкретного приложения, если требования заказчика не слишком велики, то вполне хватает возможностей Microsoft Access. При больших же объемах данных со сложной структурой можно, не изменяя своего приложения, подключить его к Microsoft SQL Server, который является более сложным приложением по сравнению с Microsoft Access. Таким образом, основными средствами при разработке прикладных программ являются:

1) Разработка внутриотраслевых приложений.
2) Разработка межотраслевых приложений.

Как средство маркетинга.

Корпорация Microsoft использует возможности собственного программного продукта для создания приложений баз данных в области маркетинга и продажи своих продуктов. Существует база данных, называемая Microsoft Power Point, используемая для презентаций отделом маркетинга. С ее помощью торговый представитель может вводить условия отбора слайдов для представления презентаций, включая рекламируемые продукты, тип аудитории и максимальное время, отводимое на презентацию. В другой базе данных Microsoft Press Book содержится информация о книгах издательства Microsoft. Это приложение предоставляет широкие возможности для поиска информации, а также позволяет идентифицировать себя как заказчика и создать заказ на интересующие книги.

В области маркетинга можно выделить следующие области применения Microsoft Access:

1) Поддержка презентаций.
2) Информационное обеспечение.
3) Обработка заказов.

На крупном предприятии.

Современные фирмы обладают разветвленными информационными подразделениями, на которые возлагаются обязанности по обеспечению фирмы важной компьютерной информацией. Обычно все корпорации начинают с построения операционно–учетных систем обработки данных. Такие системы накапливают и обрабатывают сведения о ежедневно совершаемых сделках и операциях таких как:

1) Сведения о банковских операциях.

2) Данные о проданных изделиях и закупленных материально – технических запасах.

3) Сведения о заказанных и поступивших комплектующих и материалах.

4) Данные об энергозатратах, поступлении сырья и другие сведения административно–хозяйственного характера.

С точки зрения ввода, методов разработки и вывода данных подобные операционно-учетные системы достаточно просты в разработке и использовании, но с точки зрения анализа информации не всегда удобны. Потребность в анализе информации привела корпорации к необходимости создания обширных, структурированных по отделам производственных информационных сетей, которые в свою очередь связаны с рабочими местами сотрудников. С ростом объема данных, циркулирующих в корпорации, управление, поиск и доступ к ним становятся все сложнее. Основное преимущество Microsoft Access при использовании в крупных фирмах состоит в том, что он обладает способностью осуществлять связь с базами данных разных форматов, находящимися на рабочих станциях, сетевых серверах или больших ЭВМ. Microsoft Access позволяет легко получить прямой доступ к исходным данным, построить запрос для извлечения необходимой информации и создать отчет, по полученным данным. Именно способность получать данные из многих источников, в сочетании с легкостью использования позволяет Microsoft Access создавать эффективные системы обработки информации. Т.к. Microsoft Access может работать в режиме коллективного доступа к базам данных различных форматов, то он является хорошим средством разработки приложений для рабочих групп, которые хранят данные на серверах локальных сетей своих подразделений и в то же время периодически пользуются приложениями других подразделений и сбрасывают данные на серверы корпорации. Пример такой структуры приведен на рисунке 3.

В приложениях, ориентированных на небольшие рабочие группы, хранение и коллективный доступ к информации может осуществляться при помощи только одного Microsoft Access. В случае же больших приложений используется специальный сервер (Microsoft SQL Server), а Microsoft Access в этом случае выступает в роли клиента. Чаще всего в крупных корпорациях Microsoft Access используется в общей информационной системе как пользовательская среда для обработки данных. Соответствующим образом подготовленные пользователи могут применять Microsoft Access для создания собственных отчетов диаграмм и запросов.

Таким образом, области применения Microsoft Access в крупной фирме заключаются в следующем:

1) Приложение для рабочих групп.

2) Системы обработки информации.

В качестве персональной СУБД.

Microsoft Access является удобным средством для компьютерной обработки личной информации. Так как Microsoft Access упрощает процесс создания форм и отчетов и позволяет с помощью макросов и модулей легко связывать их, разработка больших личных приложений занимает мало времени.

Области применения в качестве личной СУБД:

1) Введение инвестиционного портфеля.
2) Справочник по адресам.
3) Различные виды каталогов.

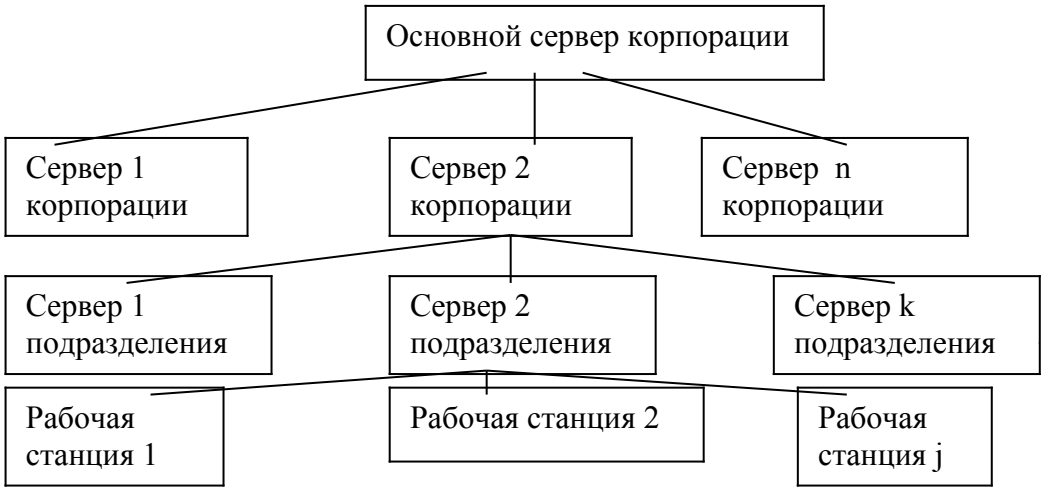


Рис. 3. Информационная структура корпорации.

2.1.3 Архитектура Microsoft Access.

Объекты базы данных Microsoft Access.

Microsoft Access (MA) называет объектами все, что может иметь собственное имя в смысле MA. Термин « база данных » обычно относится только к файлам, в которых хранятся данные. В Microsoft Access база данных включает в себя все объекты, связанные с хранимыми данными. В том числе и те, которые определяются для автоматизации работы. Всего в Microsoft Access определено 6 основных объектов.

1) Таблица.

Таблица – это объект, который определяется и используется для хранения данных. Каждая таблица содержит информацию о субъектах определенного типа (например, клиентах). Поля (столбцы таблицы) служат для хранения различных характеристик субъектов (например, ФИО и адреса клиентов), а каждая запись (строка) содержит сведения о конкретном субъекте (например, о клиенте по фамилии Иванов). Для каждой таблицы необходимо определить первичный ключ (одно или несколько полей, имеющих уникальное для каждой записи значение) и один или несколько индексов, ускоряющих доступ к данным.

2) Запросы.

Запрос – это объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц на основе заданных критериев. Для определения запроса можно использовать так называемый запрос по образцу, в котором определяется критерий поиска нужной информации, или написать инструкции на языке SQL. Можно создавать запросы на выборку, удаление, обновление или добавление данных. С помощью запросов можно так же создавать новые таблицы, используя данные из одной или нескольких существующих таблиц.

3) Формы.

Форма – это объект, предназначенный в основном для ввода данных, отображения их на экране и управления работой приложения.

Можно использовать формы для того, чтобы реализовать требования пользователя к представлению данных таблиц или наборов записей запросов. С помощью форм можно в ответ на какое – то событие запустить макрос или модуль. Причем форма может быть распечатана.

4) Отчет.

Отчет – это объект, предназначенный для форматирования, вычисления итогов и печати выбранных данных.

5) Макрос.

Макрос – это объект, представляющий собой структурированное описание одного или нескольких действий, которые должен выполнить Microsoft Access в ответ на определенное событие. Например, можно определить макрос, который при выборе некоторого элемента в основной форме будет открывать другую форму.

В макрос можно включить дополнительные условия для выполнения или пропуска тех или иных указанных в нем действий. Макросы можно использовать для открытия таблиц, выполнения запросов, просмотра или печати отчетов. Из одного макроса можно запустить другой макрос или процедуру VBA.

6) Модуль.

Модуль – это объект, содержащий программу на языке VBA для приложений, позволяющий разбить некоторый процесс, на несколько небольших процедур и обнаружить ошибки, которые невозможно найти, используя макросы.

Модули могут быть независимыми объектами, содержащими функции, вызываемые из любого места приложения или непосредственно привязанными к формам или отчетам для реакции на те или иные события.

Взаимосвязи основных объектов Microsoft Access.

Все объекты в МА взаимосвязаны друг с другом. Схематично связь между ними можно представить в следующем виде (рис.4).

В таблицах хранятся данные, которые можно извлечь с помощью запросов. Используя формы, можно выводить данные для пользователя и изменять их. Формы и отчеты получают данные как непосредственно из таблиц, так и через запросы. Для выполнения необходимых вычислений и форматирования данных запросы могут использовать встроенные функции или функции, созданные с помощью языка VBA. События, происходящие в формах или отчетах, могут запускать макросы или модули.

Событие – это любое изменение состояния объекта в Microsoft Access. Например, открытие формы, ввод новой строки в форму, изменение содержимого текущей записи и т.д., для обработки события можно создать макрос или модуль. С помощью макросов и модулей можно изменять ход выполнения приложения, открывать фильтровать и изменять данные в формах и отчетах, выполнять запросы и создавать новые таблицы. Используя язык VBA можно создавать, модифицировать и удалять любой объект Microsoft Access.

2.1.4 Разработка проекта приложения.

Основные этапы разработки приложений.

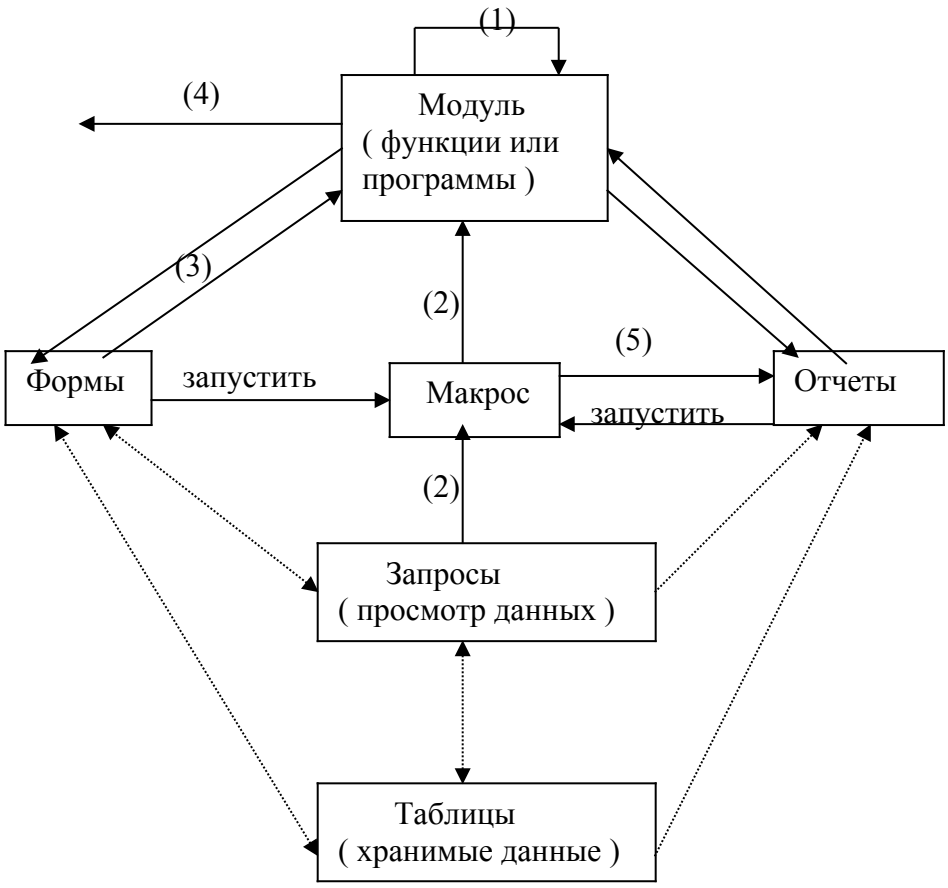
Основы методологии проектирования прикладных программ были заложены в 60-х годах прошлого века. В самом начале приходилось тратить не менее 60% всего необходимого времени на проектирование и лишь 40% на написание и отладку программы. Современные технологии разработки прикладных программ делают построение приложения дешевыми и быстрыми.

Однако, несмотря на мощность средств разработки, если не потратить определенных усилий на постановку задачи и принципов работы приложения, то потом придется потерять значительно больше времени на всевозможные переделки.

Этапы разработки приложений:

1) Уточнение задач.

Необходимо составить список всех основных задач, которые должны решаться этим приложением, включая и те, которые не нужны на данный момент, но могут потребоваться в будущем.



(1) – Выполнить, вызвать.
(2) – Выполнить функцию.
(3) – Запустить.
(4) – Открывать и форматировать в формах, отчетах и запросах.
(5) – Запустить печать, форматирование, просмотр.
——→ - Действие.
·····→ - Поток данных.

Рис. 4. Схема связей объектов в МА.

1) Последовательность выполнения задач.

Необходимо объединить основные задачи в тематические группы и затем упорядочить задачи каждой группы так, чтобы они располагались в порядке их выполнения.

2) Анализ данных.

Необходимо составление подробного перечня всех данных, необходимых для решения каждой задачи. Определить какие из них будут исходными и не будут изменяться в процессе выполнения задачи, которые будут добавлены или наоборот, удалены.

3) Определение структуры данных.

После предварительного анализа всех необходимых для приложения элементов данных нужно упорядочить их по объектам и соотнести объекты с таблицами и запросами базы данных.

4) Разработка макета приложения и пользовательского интерфейса.

Задав структуру таблиц в Microsoft Access легко создать макет приложения с помощью форм и связать их между собой, используя макросы или процедуры VBA.

5) Создание приложения.

В случае простых задач созданный макет является практически готовым приложением. Однако довольно часто приходится писать процедуры, позволяющие полностью автоматизировать решение намеченных в проекте задач, создать специальные формы, обеспечивающие переход от одной задачи к другой и решить ряд других вопросов, которые нужны для создания приложения.

6) Тестирование и усовершенствование.

После завершения работ по отдельным компонентам приложения необходимо проверить функционирование приложения в каждом из возможных режимов. По мере функционирования приложения могут возникнуть предложения по расширению или добавлению тех или иных функций, а также возможной отмене некоторых. Подобные усовершенствования могут происходить на данном этапе и по мере эксплуатирования приложения.

При разработке приложения СУБД могут использоваться два подхода к проектированию:

1) Проектирование «сверху вниз», при котором разработка приложения начинается с определения основных функций и задач.

2) Проектирование «снизу в вверх», при котором сначала проводится анализ данных и определяется их структура.

При построении некоторых СУБД могут использоваться сочетания обоих подходов. Например, начиная с определения задач и их группировки можно решить ограничиться одной базой данных или нет (подход «сверху вниз»). Базы данных привязываются к решению определенных, связанных между собой групп задач или функций. Для каждой задачи определяется набор необходимых данных, затем собираются все поля данных для связанных задач и начинается процесс формирования объектов (подход «снизу в вверх»).

Основные принципы проектирования базы данных.

В реляционной СУБД каждую базу данных нужно строить на основе некоторого набора задач или функций. В процессе проектирования возникает вопрос как в ориентированной на конкретные задачи базе организовать хранение данных, используя преимущества реляционной модели и избежать лишних затрат, связанных с неэффективным использованием памяти, заключающихся в следующем:

1) Создание дубликатов полей.
2) Резервирование памяти под максимально возможное число предметов.
3) Использование памяти для хранения сведений, которые могут быть вычислены.
4) Использование составных полей, включающих различные объекты.

Для решения этих проблем используется процесс, называемый нормализацией. Он позволяет организовать поля данных в группы таблиц. В основе процесса нормализации лежит математическая теория, которую можно сформулировать в виде определенных правил:

1) Уникальность полей.

Неэффективное использование памяти является основным недостатком ненормализованных таблиц, поэтому удаление избыточных полей из таблиц является одним из решений этой проблемы.

Определение: *каждое поле таблицы должно предоставлять уникальный тип информации*. Это правило означает, что необходимо избавиться от повторяющихся полей и разделить составные поля на отдельные элементы данных.

Если использовать первое правило, то возникают повторяющиеся данные, которые необходимы для поддержания связей между таблицами. Разрешить проблему может следующее правило нормализации.

2) Первичные ключи.

База данных спроектирована хорошо в том случае, если каждая запись любой таблицы однозначно идентифицируется. Это означает, что значение некоторого поля или нескольких полей не повторяется ни в одной другой таблице. Такой идентификатор называется первичным ключом.

Определение: *каждая таблица должна иметь уникальный идентификатор (первичный ключ), который может состоять из одного или нескольких полей.*

3) Функциональная зависимость.

После определения первичных ключей для каждой таблицы необходимо проверить, что бы все данные, включенные в таблицы, относились к соответствующим объектам, т.е. необходимо, чтобы каждое поле функционально зависело от первичного ключа.

Определение: *для каждого значения первичного ключа значения столбца данных должны относится к объекту таблицы и полностью его описывать.*

Это правило может использоваться двояко. Во–первых, в таблице не должно быть данных, не относящихся к объекту, определяемому первичным ключом. Например, в каждом заказе требуется информация о заказчике, но сам заказчик является самостоятельным объектом, и данные о нем располагаются в отдельной таблице.

Во – вторых, данные в таблице должны полностью описывать объект. Например, получателем заказанных книг не обязательно должен быть заказчик (может быть – это подарок). Поэтому следует ввести данные о получателе в таблицу.

4) Независимость полей.

Последнее правило позволяет проверить не возникнут ли проблемы при изменении данных в таблицах.

Определение: *необходимо иметь возможность изменять значение любого поля, не входящего в первичный ключ, без воздействия на данные других полей.*

Например, если в заказе неверно указан магазин, то нельзя заменить название магазина, без соответствующего изменения его адреса. Если же несколько заказов имеют неправильное название магазина, то придется вносить изменения в несколько строк таблицы. Следовательно, необходимо создать отдельную таблицу, описывающую магазин как отдельный объект.

Согласно, правил нормализации база данных представляет собой множество отдельных таблиц. При этом для связи между таблицами используются поля, называемые «чужими (вторичными) ключами». В хорошо спроектированной базе данных правильное использование чужих ключей обеспечивает максимальную эффективность работы приложения.

При построении БД ориентируются на перечисленные правила, но могут возникать ситуации, когда данным правилам не следуют. Существуют исключения из правил нормализации:

1) При необходимости повышения скорости выполнения важных задач, когда быстродействия не хватает, тогда разбиение таблиц делают на более высоком уровне, т.е. создают таблицы, включающие в себя данные, которые можно было бы занести в отдельные таблицы.

2) При использовании изменяющихся со временем данных создают таблицы, в которых происходит дублирование информации.

3) При накоплении данных для сложного запроса могут создаваться временные таблицы с нарушением правил нормализации.

2.2. Работа с данными в Microsoft Access

2.2.1 Построение баз данных в Microsoft Access.

2.2.1.1 Создание таблиц.

Создание базы данных начинается с построения таблиц. В Microsoft Access существует три режима построения таблиц:

1) Режим таблиц.

В этом режиме возникает обычная электронная таблица, в которую можно вводить данные: текст, числа, даты, денежные значения. Но в отличие от электронной таблицы в нее нельзя внести вычисляемые выражения. При сохранении таблицы Microsoft Access позволяет задать первичный ключ и присвоить ему тип данных «счетчик».

2) Мастер таблиц.

С помощью мастера таблиц можно на основе имеющегося набора таблиц создавать новые таблицы, используя те поля, которые уже существуют, внося в их названия необходимые изменения. С помощью мастера таблиц можно определить первичный ключ в таблице, а также создать новое имя, использовать уже имеющиеся связи между таблицами.

3) Режим конструктора.

Предоставляет возможность построения таблиц с нуля. Конструктор таблиц позволяет осуществить настройку таблиц для работы соответствующим образом запросов, форм и отчетов. В режиме конструктора можно задать имя поля, тип данных каждого поля и дать краткое описание каждого из полей. После выбора соответствующего поля можно указать свойства данного поля.

2.2.1.2 Определение полей.

Типы данных.

Microsoft Access позволяет поддерживать девять типов данных, краткое описание которых можно посмотреть в таблице 4.

Для символьных данных, как правило, выбирают тип данных текстовый. Размерность поля устанавливают с помощью определенного свойства.

Поле МЕМО используется в тех случаях, когда в тексте присутствуют символы табуляции и возврата каретки (Enter).

При выборе числового типа данных следует особо внимательно подходить к размеру данного поля, поскольку оно определяет точность значения данных и объем памяти, необходимый для их хранения

Тип данных дата/время используется для хранения календарных дат или значений времени и позволяет выполнять вычисления в единицах измерения времени.

Тип данных денежный обеспечивает точность до 15 знаков слева от запятой и 4 знаков справа.

Тип данных счетчик используется для определения первичного ключа таблицы. Числа могут генерироваться как случайным образом, так и последовательно. Таблицы не могут содержать более одного поля с таким типом данных.

Логический тип данных используется для хранения значений, имеющих два состояния: истина или ложь.

Поле объекта OLE позволяет хранить графики, рисунки, фотографии или звуковые фрагменты.

Гиперссылка содержит данные об адресе, который может находится в Internet, корпоративной сети, локальной сети или на самом компьютере.

Таблица 4. Типы данных Microsoft Access.

Типы данных	Использование	Размер
Текстовый	Алфавитно–цифровые данные.	До 255 байт (знаков).
Поле MEMO	Алфавитно–цифровые данные, предложения абзацы.	До 64 КБ.
Числовой	Числовые данные.	1,2,4,8 байт.
Дата/время	Временные параметры.	Максимально 8 байт.
Денежный	Данные о денежных суммах.	Максимально 8 байт.
Счетчик	Уникальное длинное целое число генерируемое Microsoft Access при создании каждой новой записи.	Максимально 4 байта.
Логический	Логические данные.	1 бит.
Поле объекта OLE	Графическая информация.	До 1 ГБ.
Гиперссылка	Адрес ссылки на необходимый документ.	До 2 КБ.

Свойства полей (общие).

Для каждого поля таблицы можно задать значения свойств, список которых зависит от выбранного типа данных.

Свойства:

1) Размер поля.

Это свойство задает максимальный размер данных, хранимых в поле. Поле с текстовым типом данных может иметь размер 1 – 255 символов, а по умолчанию устанавливается 50.

Для числового типа данных размеры поля могут быть следующими:

а) байт-целое число от 0 до 255 (занимает в памяти 1 байт).

б) целое - целые числа от –32768 до 32767 (в памяти занимает 2 байта).

в) длинное целое - целые числа от –2147483648 до 2147483647 (занимает в памяти 4 байта).

г) с плавающей точкой - числа с точностью до 6 знаков от $-3{,}4*10^{38}$ до $3{,}4*10^{38}$ (занимает в памяти 4 байта).

д) с плавающей точкой - числа с точностью до 10 знаков от $-1{,}797*10^{308}$ до $1{,}797*10^{308}$ (занимают в памяти 8 байт).

2) Формат поля.

Можно задать формат представления данных при выводе на экран или печать. Например, дата/время – 15-апр-00.

3) Число десятичных знаков.

Для числового или денежного типов данных можно задать число знаков, выводимых после запятой. По умолчанию устанавливается значение авто, для которого характерно 2 знака после запятой. В целом оно может изменяться от 0 до 15 знаков.

4) Маска ввода.

Для типов данных текстовый, числовой, денежный и дата/время можно задать маску, которую пользователь увидит при вводе значений в поле. Например, для телефона она может выглядеть так _ _ _ _ _ _ _.

5) Подпись.

Можно определить более содержательное название поля, которое Microsoft Access будет выводить в элементах управления форм и в заголовках отчетов.

6) Значение по умолчанию.

Можно определить значение по умолчанию для любого типа данных кроме счетчика и поля объекта OLE. Для числовых полей значение по умолчанию ноль. Для логических – ложь. Для текстовых – NULL, которое называется пустым значением, но для Microsoft Access оно является неизвестным.

7) Условие на значения.

Можно задать выражение, которое при вводе или редактировании значения всегда должно быть истинным.

8) Сообщение об ошибках.

Можно указать текст сообщения, которое Microsoft Access будет выводить в том случае, если вводимые данные не удовлетворяют условию на значения.

9) Обязательное поле.

Если пользователь не допускает, чтобы в поле хранилось значение NULL, можно установить для этого свойства значение « да ».

10) Пустые строки.

Для текстовых и MEMO полей можно разрешить ввод пустых строк, которые означают, что значение поля известно, но поле пустое.

11) Индексированное поле.

Данное поле позволяет ускорить доступ к хранящимся в нем данным. Это свойство доступно для полей с типом текстовый, числовой, денежный, дата/время и счетчик.

Свойства полей (подстановка).

Данные свойства предназначены для заполнения полей данными из других таблиц, а также для установления связей между таблицами. К ним относятся следующие свойства:

1) Тип элемента управления.

Задает тип элемента управления, используемого по умолчанию, для отображения данного поля в формах, отчетах и объектах в режиме таблицы. Для большинства полей следует использовать тип управления «поле». Если поле является чужим ключом, то для отображения значений из связанной таблицы лучше выбирать элемент управления список или поле со списком.

2) Тип источника строк.

Если для свойства тип элемента установлено значение список или поле со списком, то это свойство позволяет задать тип источника данных для элемента управления (таблица/запрос, список полей или список значений).

3) Источник строк.

Если для свойства тип источника строк установлено таблица/ запрос или список полей, необходимо указать таблицу или запрос, из которого поступает значение для списка. Если источником данных служит список значений, можно ввести эти значения, разделяя их точкой с запятой.

4) Присоединенный столбец.

Значения указанного столбца используются в качестве значения элемента управления. Если источник данных содержит один столбец, то для данного значения устанавливается 1.

5) Число столбцов.

Вводится число столбцов, поступающих из источника данных.

6) Заглавие столбцов.

При установке значения « да » значения свойства подпись, заданные в источнике данных выводятся в качестве заголовка столбцов списка.

7) Ширина столбцов.

Выводятся значения для ширины столбцов, которые разделяются точкой с запятой. Если нет необходимости выводить некоторый столбец, то для него устанавливают ширину равную 0, если элементом управления является поле со списком, Microsoft Access отображает в нем значения из первого столбца с ненулевой шириной.

8) Число строк списка.

Если элементом управления является поле со списком, то это свойство определяет число строк, выводимых в раскрывающемся списке.

9) Ширина списка.

Определяет ширину раскрывающегося списка. Для поля со списком значением по умолчанию является авто, при котором ширина раскрывающегося списка равна ширине элемента управления. Если ширина недостаточна, можно ввести необходимые значения.

10) Ограничится списком.

При установке значения «да» в поле можно ввести только значения, содержащиеся в списке. Чтобы иметь возможность вводить произвольные значения, нужно установить для данного свойства значение «нет».

2.2.2 Задание условий на значения для полей.

Условия на значения задаются выражением, которое в общем случае состоит из операторов сравнения и операндов. Если выражение не содержит оператора, то Microsoft Access по умолчанию использует оператор сравнения равно « = ». Можно использовать несколько сравнений, связанных логическими операторами OR или AND. Текстовые значения всегда должны заключатся в кавычки. Если в выражении используется текстовая строка, содержащая пробелы, или специальные символы, то в кавычки заключается вся строка.

В условии на значения в качестве операторов могут использоваться следующие символы сравнения: <, >, <=, >=, <>, =.

Есть несколько специальных операторов:

IN – проверяет на равенство любому значению из списка, где операндом является список, заключенный в круглые скобки. Например, для поля « Город » операнд может выглядеть следующим образом: (« Санкт – Петербург », « Москва », « Калуга »).

BETWEEN – проверят, что значение поля находится в заданном диапазоне. Верхняя и нижняя границы разделяются логическим оператором AND.

Например, запись BETWEEN 50 AND 100 означает, что диапазон значений находится в пределах от 50 до 100, по - другому это можно записать (>= 50 AND <=100).

LIKE – проверяет соответствие текстового или MEMO поля заданному шаблону символов. Оператор сравнения LIKE в качестве операнда использует текстовую строку, которая определяет допустимые символы в конкретной позиции шаблона.

В качестве символов могут выступать следующие обозначения:

? - соответствует одному произвольному символу.

✶ - соответствует любому, включая нулевое, количество произвольных символов. Используется для замены последовательности символов.

- знак числа. Соответствует одной произвольной цифре.

Примером строки с оператором LIKE может служить такая запись: LIKE ???ов?

Аналогично условия на значения можно устанавливать не только для одного поля, но и для всей таблицы в целом.

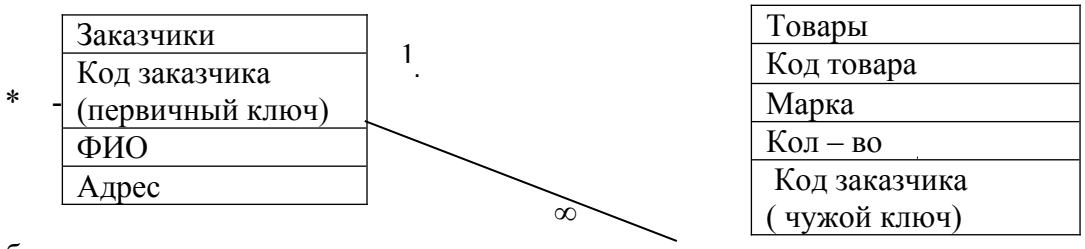
2.2.3 Создание первичного ключа и установление связей.

Каждая таблица в реляционной базе данных должна иметь первичный ключ. Если при разработке базы данных уже заранее определено, какие поля будут использованы в качестве первичного ключа, то остается только установить эти поля в соответствующем статусе. Если же такой определенности нет, то Microsoft Access позволит задать в качестве первичного ключа соответствующее поле с названием «Код». Тип данных, соответствующий этому полю, выставляется « Счетчик », который генерирует уникальные для данной таблицы номера записей.

Определив несколько таблиц необходимо определить взаимосвязь их друг с другом, тогда в дальнейшем Microsoft Access сможет связывать эти таблицы при их использовании в запросах, формах или отчетах. Перед установлением связей между двумя таблицами необходимо проверить следующие условия:

1) Первичному ключу одной таблицы соответствует вторичный ключ другой таблицы.
2) Определить где находится соотношение один, а где многие.

Иллюстрацию этих условий смотри на рис. 5.



будем считать, что один заказчик заказывает много товаров.

Рис. 5. Установление связи 1-∞.

Механизм установления связей между таблицами достаточно прост. Необходимо поле с соотношением один перенести в поле с соотношением многие. В результате этих действий Microsoft Access создаст связь и нарисует ее между таблицами (рис.4).

Для связи двух таблиц тип данных связываемых полей должен совпадать, исключение составляет связывание поля с типом данных « Счетчик » - ему должно соответствовать поле с типом данных «Числовой ».

При установлении связи между таблицами Microsoft Access поддерживает два вида связей: « один ко многим (1 - ∞) » и « один к одному (1 - 1) ».

1) « Один ко многим ».

Это означает, что любой записи в первой таблице может соответствовать несколько записей во второй. Однако любая запись второй таблицы соотносится только с одной записью в первой. Например, любой заказчик может иметь несколько заказов, но каждый заказ будет относиться только к одному заказчику.

2) « Один к одному ».

Любая запись в одной таблице может быть связана только с одной записью во второй и наоборот. Например, финансовое положение заказчика характеризуется такой однозначной связью.

Связи с соотношением многие ко многим Microsoft Access не поддерживает. Для того чтобы отойти от данного типа связей необходимо создавать таблицу пересечения, позволяющую свести связь ∞ - ∞ к двум связям 1 - ∞.

2.2.4 Создание индексов.

Чем больше данных хранится в таблицах, тем больше индексов нужно для эффективного поиска информации.

Индекс – это внутренняя таблица, состоящая из двух столбцов: значение выражения, содержащего все поля, включенные в индекс и местоположение каждой записи таблицы с данным значением индексного выражения.

Например, если очень часто осуществляется поиск заказчиков в соответствующей таблице по названию города и индекс отсутствует, а необходимо найти всех заказчиков из города Москва, Microsoft Access должен будет просмотреть все записи таблицы. Эта операция будет быстро выполняться при небольшом количестве заказчиков, если же записей несколько сотен тысяч, поиск осуществляется медленно. При наличии индекса по полю город программа сможет использовать его для ускорения поиска записей со сведениями о заказчиках, находящихся в указанном городе. Как правило, индексы содержат значения только одного поля. Microsoft Access использует этот тип индекса для ограничения количества записей, которые ему приходится просматривать для любого условия поиска по соответствующему полю. Например, реализацию условия Город = « Москва » смотри в таблице 5.

Таблица 5. Создание индекса.

	Код	Город	…
	1	Москва	
	40	Москва	
*	80	Москва	
	100	Москва	

Москва	1
Москва	40
Москва	80
Москва	100

При условии Город = « Москва » Microsoft Access создает индексную таблицу *

Если определены индексы для нескольких полей и задан критерий поиска более чем по одному полю, то Microsoft Access использует индексы совместно, что бы быстрее найти нужные записи. Например, если созданы отдельные индексы по полям « Город » и «Фамилия » и необходимо найти записи, удовлетворяющие условию Город = « Москва », Фамилия = « Иванов », то все строки индекса по полю Город, содержащие значение «Москва» Microsoft Accesscравнит со строками в индексе Фамилия = «Иванов». Результатом будет сокращение количества указателей к записям, удовлетворяющим обоим условиям.

Для установки индекса по одному полю существует две возможности:

1) Для полей, в которых создается индекс и существуют записи с повторяющимися значениями в свойстве полей « Индексированное поле » установить значение «Допускается повторение ».
2) Для этих же полей, не имеющих повторяющихся строк, Microsoft Access может создать индекс, содержащий только уникальное значение данного поля. Для этого устанавливают значение « Совпадения не допускаются ».

При частом поиске в больших таблицах по нескольким полям можно создать составные индексы, в этом случае Microsoft Access сокращает имя поиска, поскольку ему не приходится сравнивать значения из двух индексов. Например, можно объединить 3 поля: фамилия, имя, отчество в один индекс, тогда Microsoft Access осуществит поиск быстрее.

Microsoft Access использует составные индексы даже тогда, когда поиск ведется по значениям лишь некоторых полей. Microsoft Access может использовать составной индекс в тех случаях, когда в условия поиска включается последовательность полей, начиная с первого поля индекса, т.е. поиск может осуществляться по имени, по имени и отчеству, по ФИО. Существуют, правда, ограничения на использование составных индексов: только последнее условие поиска может содержать операторы неравенства <, <=, >, >=. Например: (Имя = « Анатолий » OR Фамилия = « Иванов »). Эта запись будет выполнена.
А эта запись неверна: (Имя < « Анатолий » OR Фамилия = « Иванов »).

2.2.5 Ограничения для баз данных.

При проектировании БД в МА существует ряд ограничений, накладываемых на БД:

1) Таблица может содержать не более 255 полей.
2) Таблица не может иметь более 32 индексов.
3) Составные индексы не могут включать более 10 полей, причем общая их длина не должна превышать255 байт.
4) Общая длина строки таблицы без учета полей MEMO и объектов OLE не может превышать 2 КБ.
5) Размер поля MEMO ограничивается 1 ГБ, но нельзя вывести поле MEMO размером более 64 КБ в форме или в режиме таблицы.
6) Размер объекта OLE не должен превышать 1 ГБ.
7) На число записей в таблице не существует ограничений, но размер файла базы данных Microsoft Access не может превышать 1 ГБ. Если же существует несколько больших таблиц, то в этом случае придется создавать для каждой из них отдельную базу данных, а затем связывать их с базой данных, содержащей формы, отчеты, макросы и модули соответствующего приложения.

2.2.6 Изменение проекта базы данных.

Ограничения на изменения.

Как бы тщательно не проектировалась база данных, в процессе работы неизбежно приходится столкнуться с необходимостью внесения изменений в исходный проект. Microsoft Access обеспечивает достаточную простоту при внесение изменений в проект базы данных, даже при наличии информации, находящейся в таблицах. Однако следует учитывать следующие ограничения:

1) Сделанные в таблицах изменения Microsoft Access не вносит автоматически в использующие их отчеты, формы, макросы и модули. Зависимые объекты необходимо редактировать самостоятельно.
2) Нельзя изменить тип данных поля, используемого в определении связи между таблицами. Сначала необходимо удалить соответствующую связь, затем изменить тип поля и снова определить связь.
3) Нельзя изменить определение таблицы, открытой в запросе, форме или отчете. Прежде чем открыть таблицу в режиме конструктора для внесения изменений необходимо закрыть все открытые объекты, ссылающиеся на нее. Если разрешен доступ к базе данных другим пользователям сети, то перед внесением изменений в таблицу необходимо убедиться, что ни один из них не работает с ней.

Преобразование данных.

Microsoft Access позволяет производить различные манипуляции с таблицами: удалять, переименовывать, трансформировать одни таблицы в другие. Особенно нужно отметить, что Microsoft Access позволяет изменять тип данных или размер поля даже в том случае, если поле уже содержит информацию. В этом случае возможны два варианта:

1) Преобразование без ограничений.

Таблица 6. Варианты преобразования типов данных без ограничений.

Типы преобразуемых данных	Типы преобразованных данных
Числовой	Текстовый
Дата/время	Текстовый Поле MEMO Числовой Денежный
Денежный	Текстовый Поле MEMO
Счетчик	Текстовый Поле MEMO Числовой Денежный
Гиперссылка	Поле MEMO

2) Преобразования, на которые накладываются ограничения.

В некоторых случаях при преобразовании одних типов данных в другие возможна трансформация информации, потеря отдельных данных, а иногда и полное исчезновение информации, поэтому изменение типов данных, с уже имеющейся информацией необходимо производить с большой осторожностью.

2.2.7 Режим таблицы.

Просмотр и изменение данных.

Самый простой способ просмотреть данные, хранящиеся в таблице, это открыть ее в режиме самой таблицы. С помощью данного режима можно выполнять такие основные операции, как поиск, обновление, вставка, удаление данных. В режиме таблицы можно изменять ее параметры:

1) Изменять высоту строк и ширину столбцов.
2) Изменять порядок расположения столбцов.
3) Скрывать столбцы, т.е. делать размер поля равным нулю.

Это используется в том случае, когда необходимо отследить только нужные поля.

4) Закреплять столбцы.

Это используется, когда необходимо удерживать на экране один или несколько столбцов, а остальные прокручивать относительно их влево или вправо.

5) Менять внешний вид ячеек таблицы.
6) Изменять шрифт в таблице.

В режиме таблицы кроме просмотра данных и их форматирования можно производить следующие операции над их изменениями:

1) Добавлять новую запись.
2) Выделять и изменять данные.
3) Заменять одни данные на другие.
4) Копировать и вставлять данные.

2.2.8 Работа с гиперссылками.

Начиная с Microsoft Access 97 был введен новый тип данных “ Гиперссылка ”. Данный тип данных позволяет хранить в поле простые или сложные ссылки на файлы или документы, находящиеся вне базы данных. Такие ссылки могут содержать адрес, указывающий на некоторое место в Internet, корпоративной сети, либо сетевой маршрут к файлу на сервере локальной сети или на самом компьютере. Ссылка может указываться в формате языка, поддерживающего гиперссылки. В самой таблице гиперссылка на обычном фоне выделяется синим цветом.

Гиперссылка по своей сути является текстовым полем и может содержать до 2048 символов и состоять из трех основных частей:

1) Необязательное описание.

Это текст, который отображается в поле или элементе управления, может начинаться с любого символа кроме знака числа # и должен заканчиваться знаком #.

2) Адрес гиперссылки.

Это может быть адрес, начинающийся с имен протоколов, работающих в Internet (например, WWW…) или сетевой путь к файлу (C:\…\…\имя файла…\)

3) Необязательный дополнительный адрес.

Этот адрес дает именованный объект внутри файла. Например, можно указать закладку в документе Word. Дополнительный адрес отделяется от основного символом #, т.е. строка имеет вид: описание # адрес # необязательный адрес.

При активизации гиперссылки Microsoft Access запускает соответствующее приложение, работающее с указанным файлом и передает ему основной и дополнительный адреса.

2.2.9 Сортировка и поиск данных.

Сортировка данных.

После открытия таблицы Microsoft Access выводит строки в последовательности, определяемой значениями первичного ключа. Если первичный ключ не был определен, то записи отображаются в том порядке, в котором они были введены в эту таблицу.

Для сортировки данных в таблице можно воспользоваться следующими способами:

1) Сортировка по возрастанию.

Сортирует записи по соответствующему полю в алфавитном порядке.

2) Сортировка по убыванию.

Сортирует строки соответствующего поля в порядке, обратном алфавитному.

3) Сортировка в таблице по нескольким полям осуществляется с помощью фильтра.

Фильтр – это средство Microsoft Access, позволяющее производить как сортировку, так и поиск данных в таблице.

Сортировка по нескольким полям возможна как по отдельному полю, так и по всем полям сразу. Пример данной сортировки показан в таблице 7.

Таблица 7. Пример сортировки таблицы по нескольким полям.

Область	**Город**	**Улица**
Ленинградская	Любань	Урицкого
Ленинградская	Тосно	Ленина
Ленинградская	Тосно	Чехова
Московская	Пушкино	Комсомола

Поиск и фильтрация данных.

Поиск данных в Microsoft Access может осуществляться как по одному полю, так и по всем полям таблицы сразу. Указывая необходимый образец поиска, Microsoft Access находит все записи, соответствующие этому образцу. В образце можно использовать подстановочные символы:

* - заменяет строку любой длины.

? – один произвольный символ.

По указанному шаблону Microsoft Access найдет все встречающиеся строки.

Для фильтрации необходимых данных могут использоваться следующие фильтры:

1) Фильтр по выделенному.

Данный фильтр используется следующим образом. Необходимо выделить нужную информацию, например город Санкт – Петербург. После применения фильтра будут найдены все записи, содержащие это слово. Точно также можно отобрать все записи с определенным значением поля. При этом следует отметить, что если выделить слово в начале поля, то Microsoft Access отберет записи, в которых это слово находится также в начале поля. Подобным образом

будет проводиться отбор записей и при выделении последнего слова в поле. Если же выделить слово в середине поля, то отбираются все записи, в которых оно находится в любом месте текущего поля. Можно также выделить значение в группе смежных столбцов одной строки. В этом случае Microsoft Access выделит все записи, в которых значение в группе смежных столбцов совпадает с выделенными. К отобранным значениям можно применить новый фильтр.

2) Обычный фильтр.

Фильтр по выделенному позволяет отбирать записи, удовлетворяющие всем частным условиям отбора, т.е. если задаем поиск записей с именами “ Петр ” и фамилией “Семенов”, то будут выведены записи, содержащие оба этих слова. Если же необходимо вывести записи, содержащие хотя бы одно из этих слов, заданных при поиске, то применяется обычный фильтр, т.е. он используется, когда необходимо найти записи, удовлетворяющие только некоторым из частных условий. Таким образом, применение этого фильтра выведет все записи, в которых встречается хотя бы одно из слов “ Петр ” или “Семенов ”, таким образом, разница между фильтрами такова: (усл 1 and усл 2), (усл 1 or усл 2).

3) Расширенный фильтр.

Объединяет в себе свойства фильтра по выделенному и обычного фильтра.

2.2.10 Работа с данными при помощи запросов.

2.2.10.1 Выбор данных из одной таблицы.

Задание запроса.

Запросы бывают двух видов: однотабличные и многотабличные. Однотабличные запросы в свою очередь делятся на два вида: запросы на выборку и запросы на изменение. Запрос на выборку отбирает информацию из таблиц и самих запросов; запрос на изменение предназначен для изменения, обновления и удаления информации. При создании запроса на выборку Microsoft Access создает набор записей, соответствующих выбранным критериям. С полученным наборам записей можно работать как с обычной таблицей, т.е. просматривать и выбирать информацию, печатать и обновлять данные, однако в отличии от реальной таблицы набор записей физически не существует в базе данных. Microsoft Access создает его из данных таблиц только во время выполнения запроса.

Классификация запросов представлена на рис.6.

Первым шагом при создании запроса является выбор полей из таблицы. Поля из таблицы выводятся в запросе, называемом запрос по образцу. Главным достоинством запроса по образцу является то, что пользователь не зная языка SQL может составить запрос на выборку.

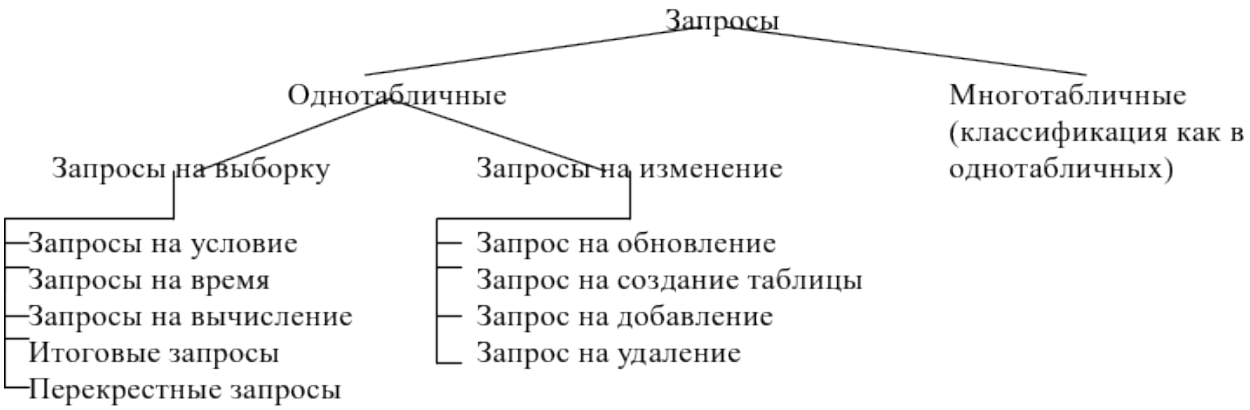


Рис.6 Классификация запросов в МА.

Выбранные поля в запросе наследуют свойства полей таблицы, но следует отметить, что в случае необходимости можно изменить некоторые свойства для полей запросов:

1) Описание.

Информация о соответствующем поле.

2) Формат поля.
3) Число десятичных знаков.
4) Маска ввода.
5) Подпись.

Следующим шагом определения запроса является отбор записей с нужными значениями полей. Поиск необходимых записей может осуществляться как по одному полю, так и по нескольким полям одновременно. В качестве условия отбора используются логические операторы AND и OR. Таблицы истинности этих операторов представлены в таблицах 8 и 9.

Таблица 8. Таблица истинности для оператора AND.

AND	Истина	Ложь
Истина	Истина (значение выбирается)	Ложь (значение отвергается)
Ложь	Ложь	Ложь

Таблица 9. Таблица истинности для оператора OR.

OR	Истина	Ложь
Истина	Истина	Истина
Ложь	Истина (значение выбирается)	Ложь (значение отвергается)

Кроме логических операторов могут использоваться операторы сравнения<,>, <=, >=, <>, =. А также операторы BEETWEN, LIKE, IN. Запросы, которые используют операторы сравнения и логические операторы, называют запросами на условие.

Условия отбора для дат и времени (запросы на время).

Microsoft Access хранит значения дат и времени, как числа с плавающей точкой двойной точности. Значения слева от десятичной точки соответствуют дате, а дробная часть числа соответствует времени суток (час, минута, секунда). Microsoft Access предоставляет несколько функций для задания условия отбора для дат и времени.

1) День (Day).

Эта функция выдает значение дня месяца в общем случае от 1 до 31.

Пример: Day [(название поля)] >10.

2) Месяц (Month).

Выдает значение месяца года от 1 до 12.

3) Год (Year).

Выдает значение года в диапазоне от 100 до 9999.

4) Неделя (Weekday).

Выдает значение, соответствующие дням недели в интервале от 1 до 7.

5) Интервал (Datepart).

Используется с ключами ww и q. Выдает номер квартала при наличии (значение от 1 до 4) и значение номера недели года при наличии ww (от 1 до 52).

Пример: Datepart [“q”; (название поля)].

6) час (Hour).

Выдает значение времени суток в диапазоне от 0 до 23.

Вычисляемые поля (запросы на вычисление).

Для выполнения вычислений с любыми полями базовой таблицы можно создать выражение и определить его в качестве нового поля запроса.

При создании вычисляемого поля можно использовать любые встроенные функции Microsoft Access.

Вычисляемое выражение может использовать следующие функции:

1) «+» - складываются два числовых выражения.
2) «-» - вычитается из первого числового выражения второе.
3) «*» - перемножение двух числовых выражений.
4) «/» - деление первого на второе.
5) «\» - округляет два числовых выражения до целых значений и делит первое на второе, результат также округляется до целого.
6) «^» - возводит первое числовое выражение в степень, задаваемую вторым числовым выражением.
7) «mod» - округляет оба числовых выражения до целых, делит первое на второе и выдает остаток.
8) «&» - создает новую текстовую строку, присоединяя вторую строку к концу первой. Если один из операндов является числом, Microsoft Access перед проведением объединения преобразует его в строку символов.

Для создания сложных выражений Microsoft Access предоставляет специальную программу, называемую « Построитель выражений », которая содержит целый ряд встроенных функций: математических, статистических, логических, временных.

Итоговые и перекрестные запросы.

Часто необходимо бывает узнать данные не по отдельным записям таблицы, а итоговые значения по группам данных. Пример показан на рис.7 .

Таблица:

ФИО студента	Номер группы	Стипендия
	1076/1	400
	1076/2	200
	1076/2	300
	1076/1	300

Запрос:

Номер группы	Стипендия группы
1076/1	700
1076/2	500

Рис.7 Пример итогового запроса.

Итоговый запрос создается с помощью команды, называемой « Групповые операции » - Σ.

Microsoft Access предоставляет следующие функции, обеспечивающие выполнение групповых операций:

1) Sum – выдает сумму всех значений в данной группе. Используется только для числовых или денежных полей.
2) Arg – выдает среднее арифметическое значение данного поля в каждой группе. Используется для числовых или денежных полей.
3) Min – выдает наименьшее значение, найденное в данном поле внутри каждой группы. Может использоваться для числовых, денежных и текстовых полей.
4) Max – выдает наибольшее значение в каждой группе.
5) Count – выдает число записей, в которых значение каждого поля отличны от “ null ”.
6) St Dev – выдает стандартное отклонение (корень квадратный из дисперсии) всех значений данного поля в каждой группе. Применяется только к числовым и денежным полям.
7) Var – выдает дисперсию среднего значения в каждой группе. Используется в числовых и денежных полях.
8) First – выдает первое значение данного поля в группе.
9) Last – последнее значение данного поля в группе.

Microsoft Access поддерживает особый тип итогового запроса, называемый перекрестным запросом. Он позволяет вывести вычисляемые значения в перекрестной таблице, похожей на электронную. Для перекрестного запроса необходимо определить одно поле в качестве

заголовков строк, одно для заголовков столбцов и одно поле значений для которого выбирается одна из групповых операций.

Пример такого определения показан на рис.8

Таблица данных:

Строки — Столбцы — Значение (Sum)

Фамилия	Года	Сумма заработка
Сидоров	1998	20000
Сидоров	1998	30000
Сидоров	1998	40000
Сидоров	1999	30000
Сидоров	1999	40000
Сидоров	1999	50000
Петров	1998	10000
Петров	1999	50000
Петров	1999	60000

Перекрестный запрос: Столбцы

Фамилия	1998	1999
Сидоров	90000	120000
Петров	10000	110000

Строки

Значение (Sum)

Рис.8 Пример перекрестного запроса.

2.2.10.2. Многотабличные запросы.

Все используемые приемы, применяемые при работе с однотабличными запросами, могут использоваться и для сложных многотабличных запросов. Только для формирования многотабличного запроса необходимо выбирать поля сразу из нескольких связанных между собой таблиц. В большинстве случаев набор записей для многотабличных запросов формируется на основе совпадения связанных полей базовых таблиц. Если создан запрос, в котором отражаются записи первой таблицы, имеющие соответствующие записи во второй, и при этом записи первой таблицы, не имеющие записей во второй не отображаются, называется симметричным отображением. Пример симметричного отображения приведен на рис.9.

Если необходимо вывести все записи первой таблицы, которые не имеют записей во второй используется внешнее объединение, для которого выставляется соответствующий параметр, означающий, что симметричное объединение переведено во внешнее.

Для внешнего объединения характерно два случая:

1) объединяются все записи из таблицы « Клубы » и только те записи из таблицы «Контракты », в которых связанные поля совпадают, т.е. в запросе выведется таблица 10.

Таблица. Клубы.

Клубы
Грибоедов
Money Honey
La Plage
Барометр
Havana Club
Plaza

Таблица. Клубы и контракты.

Контракты	Клубы
Контракт 1	Грибоедов
Контракт 2	Money Honey
Контракт 3	La Plage
Контракт 4	Барометр
Контракт 5	Havana Club
Контракт 6	
Контракт 7	

Запрос.

Контракт	Клуб
Контракт1	Грибоедов
Контракт2	Money Honey
Контракт3	La Plage
Контракт4	Барометр
Контракт5	Havana Club

Рис.9 Пример симметричного объединения.

Таблица 10. Первый случай внешнего объединения.

Контракт	Клуб
Контракт 1	Грибоедов
Контракт 2	Money Honey
Контракт 3	La Plage
Контракт 4	Барометр
Контракт 5	Havana Club
	Plaza

2) Объединяются все записи из таблицы « Контракты » и только те записи из таблицы «Клубы», в которых связанные поля совпадают, как в таблице 11.

Таблица 11. Второй способ внешнего объединения.

Контракт	Клуб
Контракт 1	Грибоедов
Контракт 2	Money Honey
Контракт 3	La Plage
Контракт 4	Барометр
Контракт 5	Havana Club
Контракт 6	
Контракт 7	

Одним из видов многотабличных запросов является построение запроса на основе другого запроса.

Сначала создается первый запрос, с помощью которого решается определенный круг задач и отбирается подмножество данных из нескольких таблиц, а затем на его основе строится другой запрос для получения окончательного набора записей.

2.2.11 Ограничения при использовании запросов на выборку для обновления данных

Набор записей, который появляется при создании запроса ведет себя как реальная таблица, содержащая данные. Во многих случаях можно вставлять строки, удалять их или обновлять данные. При этом Microsoft Access вносит необходимые изменения в соответствующие базовые таблицы запроса. Однако в некоторых случаях изменения производить нельзя:

1) В вычисляемых полях, т.к. Microsoft Access не может понять в каком из вычисляемых полей производить изменения.
2) Во всех полях итоговых или перекрестных запросов, т.к. Microsoft Access не позволяет выполнять изменения, воздействующие на несколько строк базовой таблицы.
3) В любом поле запроса, одним из источников данных которого является итоговый или перекрестный запрос.
4) В полях с типом данных “ Счетчик ”.
5) В полях, обозначенных как первичный ключ, если не установлен режим каскадного обновления связанных полей.
6) В любом поле запроса, которое выдает только уникальное значение.
7) В любом поле запроса, в котором используется операция языка SQL “UNION”, создающая результирующую таблицу по выбранным строкам.

2.2.12 Модификация данных с помощью запросов на изменение

Виды запросов на изменение.

Второй тип запросов – это запросы на изменение, с помощью которых можно трансформировать информацию в таблицах.

Существуют следующие виды запросов:

1) Запрос на обновление.

После запроса на выборку он преобразуется в запрос на обновление, с помощью которого можно вводить новые значения в отобранные записи, после чего соответствующие изменения будут произведены в базовых таблицах.

2) Запрос на создание таблицы.

После запроса на выборку он трансформируется в запрос на создание таблицы, который позволяет на основе отобранных записей создать новую таблицу.

3) Запрос на добавление

С помощью данного запроса можно копировать выбранные записи и вставлять их в другие таблицы.

4) Запрос на удаление.

Позволяет удалить выбранные записи из базовой таблицы.

Ошибки при выполнении запросов на изменение.

Во время выполнения запроса на изменение Microsoft Access различает следующие ошибки:

1) Дубликаты первичных ключей.

Такая ошибка возникает при попытке добавления или изменения некоторой записи в таблице, приводящей к появлению дублирующего значения первичного ключа.

2) Ошибки преобразования данных.

Ошибки этой категории возникают в тех случаях, когда вставляются данные в соответствующую таблицу и при этом оказывается, что типы данных полей – получателей и полей – источников не совпадают, т.е. не могут быть преобразованы друг к другу.

3) Заблокированные записи.

Данные ошибки возникают, когда при выполнении запроса на удаление или обновление, используется таблица, к которой имеют доступ другие пользователи сети.

4) Нарушение условий на значение.

Если вставляемые или обновляемые записи не удовлетворяют условию на значение для некоторого поля или таблицы в целом, Microsoft Access сообщает об этой ошибке и исключает записи, которые не удовлетворяют условию на значение при вставке или обновлении.

Помимо этого может возникнуть проблема (не являющаяся ошибкой), которая заключается в том, что в некоторых случаях Microsoft Access бывает вынужден произвести усечение данных, добавляемые в текстовые или MEMO поля. В этих случаях Microsoft Access не предупреждает об усечении данных, а просто его производит.

2.2.13 Импорт, экспорт и связывание данных

Microsoft Access можно использовать как замкнутую систему. Однако одним из основных преимуществ данного программного продукта является возможность работы с самыми разнообразными данными других баз данных, электронных таблиц или текстовых файлов.

Стандарт ODBC.

Microsoft Access для чтения, вставки, удаления и обновления данных использует язык SQL. Теоретически любой програмный продукт, который понимает SQL, должен понимать любую программу, работающую с этим языком. Хотя для SQL существуют определенные стандарты в реальной жизни большинство фирм, разрабатывающие программные продукты, используют для поддержки специальных функций различные диалекты и расширения этого языка.

Также некоторое программное обеспечение, выпущенное до принятия стандартов, получило широкое распространение на рынке и отказаться от этих программных продуктов было бы сложно.

В связи с этими проблемами в 1992 году был создан стандартный интерфейс языка CLI, позволяющий обеспечить взаимосвязь различных программных продуктов, написанных на диалектах языка SQL. Фирма Microsoft упростила интерфейс CLI, создав свой стандарт, который получил название ODBC – что означает открытый доступ к данным.

Архитектура данного стандарта может быть представлена в виде, показанном на рис.10.

Microsoft Accessявляется первым программным продуктом, использующим стандарт ODBC. Во время установки Microsoft Access также устанавливается и стандарт ODBC, с помощью которого после установки соответствующих драйверов Microsoft Access может работать с любыми базами данных, написанными на SQL.

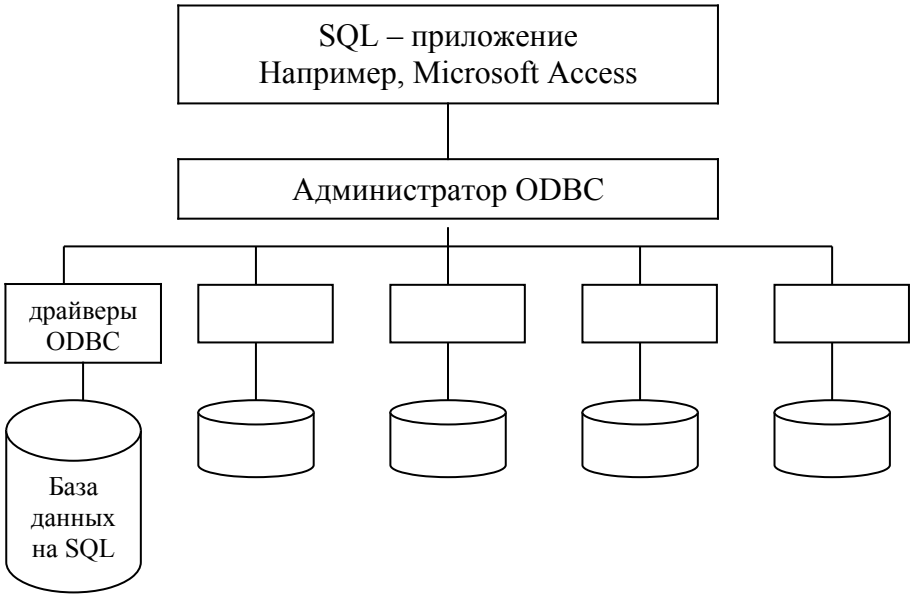


Рис. 10. Архитектура стандарта ODBC.

Сравнение импорта и связывания.

В связи с тем, что существует выбор между импортом и связыванием данных из других баз данных, необходимо принимать решение какой тип доступа лучше использовать в той или иной ситуации.

Импорт – это внесение данных непосредственно в базу данных Microsoft Access.

Связывание – это возможность работы с данными, находящимися в другой базе данных, без непосредственного внесения в базу данных Microsoft Access.

Следует отдать предпочтение импорту в следующих случаях:

1) необходимый файл сравнительно невелик и пользователи исходной базы данных редко меняют его содержимое.

2) Данные исходного файла не используются совместно с пользователями других приложений.

3) Когда заменяется старое приложение на новое и прежний формат данных больше использоваться не будет.

4) Необходимо обеспечить максимальную эффективность работы приложения.

Связывание используется в следующих случаях:

1) Файл превышает максимально возможный размер базы данных Microsoft Access.

2) Данные будут использоваться в режиме коллективного доступа.

3) Данные часто меняются пользователями исходной базы данных.

4) Созданное приложение будет распространяться среди различных пользователей и предусматривает возможные модификации его интерфейса (запросы, модули, формы, отчеты и макросы).

Импорт и экспорт данных.

Импортировать данные в Microsoft Access можно из следующих источников:

1) База данных dBase.
2) База данных Paradox.
3) База данных FoxPro.
4) База данных SQL.
5) Электронные таблицы, созданные в Excel от второй версии и выше, а также в Lotus.
6) Можно импортировать текстовые файлы, но для данного импорта нужно подготовить текст определенным образом.
7) Можно импортировать любой объект из любой базы данных Microsoft Access.

При импорте данных из электронных таблиц и текстовых файлов возможны ошибки, связанные с дублированием первичных ключей, нарушением типов данных, нарушением структуры полей, поэтому после этих случаев импорта необходимо проверить правильность занесенных данных.

Экспорт данных может быть осуществлен теми же семью объектами, что и импорт и существующей восьмой возможностью:

8) Любую таблицу, форму или отчет можно экспортировать в документ HTML (документ с разметкой гипертекста), который предназначается для просмотра в Internet.

Связывание данных.

Можно установить связь с таблицами из других баз данных и работать с ними так, как будто они определены в текущей базе данных.

Можно связывать данные с тем же списком объектов, с которым осуществляется импорт, т.е.:

1) Базы данных dBase.
2) Базы данных Paradox.
3) Базы данных FoxPro.
4) Базы данных SQL.
5) Электронные таблицы, созданные в Excel от второй версии и выше, а также в Lotus.
6) Текстовые файлы.
7) Таблицы базы данных Microsoft Access.

При связывании данных сами таблицы остаются в той базе данных, из которой они берутся, в том формате, в котором они были созданы. При связывании данных возможны проблемы с чтением, вставкой, изменением и удалением новых данных. Кроме того, следует отметить, что при связывании таблиц Microsoft Access уменьшает свою производительность, даже если исходная таблица находится на том же компьютере.

2.2.14. Язык SQL

Любой запрос в Microsoft Access реализуется с помощью языка SQL. Большинство запросов можно построить возможностями режима конструктора, но и в этом случае запрос будет храниться в форме инструкции языка SQL.

Первая версия языка SQL появилась в 70 – х годах XX века. В 1982 году был создан первый стандарт реляционных баз данных RDL. Первая версия стандарта SQL появилась в 1986 году и включила в себя большинство возможностей различных диалектов языка SQL, наработанных на данный момент.

В 1989 году появилась вторая версия языка SQL, которая явилась подмножеством практически всех реализаций SQL и обобщением всех его известных версий.

На сегодняшний день существует третья версия языка, называемая SQL III, которая разработана с учетом не только имеющихся на сегодняшний день возможностей, но и функций, которые могут быть использованы в будущем.

Язык SQL является языком высокого уровня и состоит из ряда инструкций. Ядром языка SQL является функция SELECT, которая выполняет реляционные операции выбора и объединения для создания логической таблицы (набора записей) на базе других таблиц или запросов. Используя данную инструкцию в совокупности с рядом функций можно создавать запросы любой степени сложности, которые невозможно построить в шаблоне для запросов.

Помимо данной инструкции существует еще целый ряд инструкций, позволяющих создавать конкретные виды запросов, например:

DELETE – инструкция для запроса на удаление
INSERT – для запроса на добавление
SELECT…INTO – для запроса на создание таблицы
UPDATE - для запроса на обновление

Язык SQL является мощным средством для создания запросов в БД, с помощью которого может осуществляться связь между различными видами БД.

2.3. Интерфейс пользователя в базах данных Microsoft Access

2.3.1. Использование форм

2.3.1.1. Основные сведения о формах

Применение форм.

Большинство форм создается на основе таблиц и запросов. Формы являются основным средством организации интерфейса пользователя в приложении Microsoft Access.

Формы создаются для следующих целей:

1) Вывод и редактирование данных.

Это наиболее распространенная сфера применения форм. Формы обеспечивают вывод на экран данных в выбранном представлении. Благодаря формам значительно упрощается внесение изменений, добавление и удаление данных.

2) Для управления ходом выполнения приложения.

Чтобы автоматизировать вывод определенных данных или выполнение некоторой последовательности действий можно создать формы для работы с макросами или процедурами VBA.

3) Для ввода данных.

Можно создавать формы, предназначенные только для ввода в базу данных новой информации или значений, помогающих автоматизировать работу приложения.

4) Для вывода сообщений.

Формы могут предоставить вспомогательную информацию о работе приложения или предстоящих действиях. Например, выдать информацию об ошибке.

5) Для печати информации.

Форму можно распечатать вместе со всей, находящейся в ней информацией.

Обзор форм.

Обычная информация, поступающая из базовой таблицы или запроса, размещается в области данных. Чтобы вывести на экран информацию или элемент управления, не зависящий от содержимого записи, можно добавить в форму заголовок и примечание. Таким образом, структура формы в общем виде представлена на рисунке 11.

Заголовок
Область данных
Примечание

Рис.11. Структура формы.

Различают несколько видов форм:

1) Многостраничные формы.

Если необходимо вывести на экран записи, каждая из которых содержит большой объем информации, можно создать многостраничную форму, которая включает в себя заданное количество страниц, где каждой странице соответствует только одна запись.

2) Ленточные формы.

Этот тип форм удобен для просмотра списка записей с небольшим числом полей данных. В отличие от многостраничной формы, отображающей только одну запись на странице, в ленточной форме отформатированные записи выводятся на экран одна за другой, как в таблице.

3) Подчиненные формы.

Данные формы используются для представления данных, находящихся на разных сторонах отношения один – ко – многим. Пример такой формы дан на рисунке 12.

4) Всплывающие формы.

Иногда информацию удобнее представлять в окне, постоянно находящемся на переднем плане. В Windows активное окно оказывается на переднем плане по умолчанию, а остальные окна перемещаются на задний план. Окно, которое встает на переднем плане даже в тот момент, когда активизировано другое окно, называется всплывающим. Microsoft Access позволяет создавать формы во всплывающих окнах.

5) Модальные формы.

При разработке приложения возникают ситуации, когда приложению требуется получить некоторые данные от пользователя или передать ему важную информацию прежде, чем продолжить свою работу. Таким образом, модальная форма – это такой тип форм, который требует от пользователя ответа, как непременного условия дальнейшей работы приложения.

2.3.1.2 Элементы управления форм.

Информация в форме содержится в элементах управления. Наиболее часто используемый элемент управления – «Поле». В нем могут отображаться данные из базовых таблиц или запросов, а также вычисляемые непосредственно в форме значения.

Кроме того, в форме могут использоваться следующие элементы управления:

1) Переключатели.

Позволяют сделать выбор, если имеется два или более допустимых значений для поля.

2) Флажки.

Используются для задания двух возможных вариантов для значения поля (рис. 13):

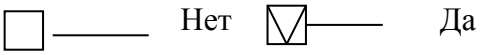


Рис. 13. Пример флажков.

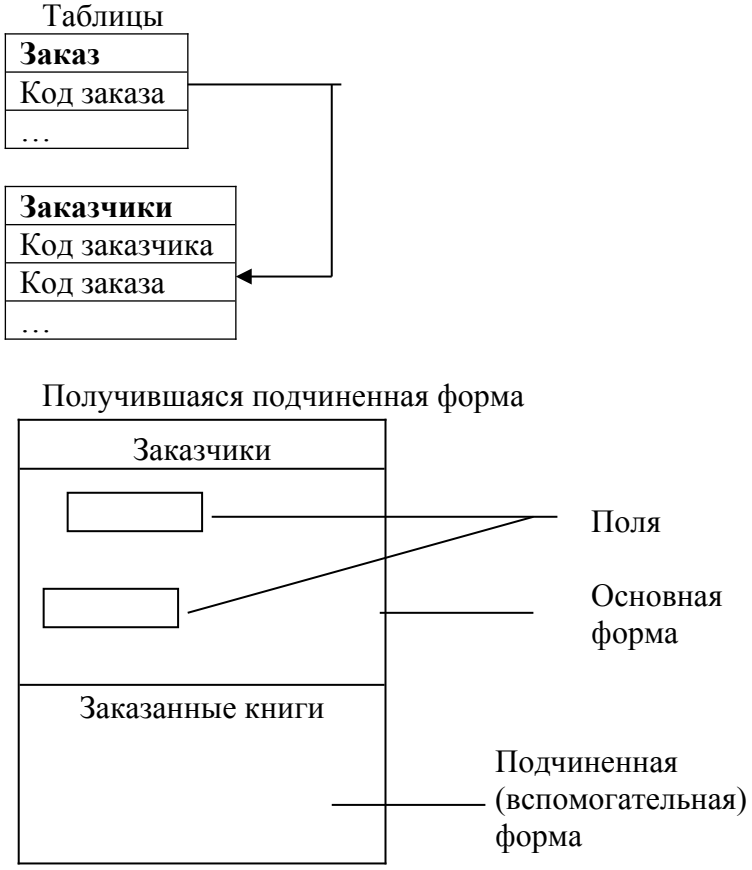


Рис. 12. Пример построения подчиненной формы.

3) Выключатели.

Представлены в виде кнопок, которые могут быть нажаты или нет (рис.14).

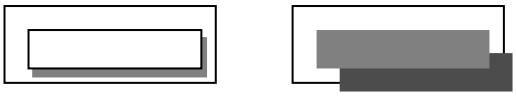


Рис. 14. Пример выключателей.

4) Группы.

Создаются из элементов переключателей, флажков и выключателей.

5) Список.

Данный элемент управления используется в том случае, если в форме требуется ввести перечень значений, который все время остается открытым.

Список может отображать набор значений, вводимых при его создании, значение поля таблицы или запроса, а также список имен полей таблицы или запроса.

6) Поле со списком.

Данный элемент похож на список где главное различие заключается в том, что помимо списка данный элемент содержит еще и поле. Одним из преимуществ данного элемента является то, что ему требуется место только для размещения одного значения, а не всего списка.

7) Набор вкладок.

Одним из способов отображения значительного объема информации является использование многостраничных форм. Альтернативой этому способу является включение в форму элемента управления Набор вкладок. Это позволяет отображать различную информацию в зависимости от выбранной вкладки на подобии того, как выводятся объекты в самом Microsoft Access.

8) Рамка объекта.

Элемент, используемый для отображения рисунков, диаграмм и других объектов OLE называется рамкой.

Данный элемент представлен в двух видах:

а) Присоединенная рамка объекта.

Используется для вывода в форме объектов OLE, хранящихся в полях таблиц базы данных.

б) Свободная рамка объекта.

Используется для выводов объектов OLE, находящихся вне базы данных.

9) Командные кнопки.

Используются для автоматизации приложения, для связи форм между собой, а также для выполнения различных действий, например, выхода из приложения. Преимущества командных кнопок заключаются в том, что они легко запускают макросы и модули.

2.3.2 Построение форм.

2.3.2.1 Формы и объектно–ориентированное программирование.

Microsoft Access является программным продуктом, обладающим свойствами объектно-ориентированного программирования.

В классических системах разработки приложений на основе процедурного программирования данные, необходимые для решения, прикладной задачи очевидным образом отличаются как от программ, создаваемых для их обработки, так и от результатов выполнения последних.

Каждая программа независимо работает с данным и обычно лишь в малой степени структурно вязана с остальными частями системы.

Объектно–ориентированное программирование основывается на понятиях объектов и действиях над ними. Объекты могут отражать другие, подчиненные им объекты.

Когда объект определяет новое действие над другим объектом, он наследует атрибуты и свойства другого объекта и тем самым расширяет свое определение.

Например, формы определяют действия над таблицами или запросами и включаемые в форму поля наследуют свойства таблиц или запросов. В форме можно задать новые свойства полей таблиц или запросов, но нельзя отменить наследуемые свойства.

Внутри базы данных Microsoft Access можно задавать взаимосвязь данных и объектов.

Таким образом, главное отличие объектно–ориентированного программирования от процедурного заключается в следующем:

1) Объектно–ориентированное программирование работает с объектами.

2) Объекты в системах объектно–ориентированного программирования взаимосвязаны друг с другом, могут наследовать свойства других объектов и используя свои возможости задания свойств придавать объектам новый статус.

3) Объектно–ориентированное программирование в отличие от процедурного может менять последовательность хода работы приложения.

Формы являются типичными представителями объектов объектно-ориентированного программирования. Общую структуру формы можно представить в следующем виде:

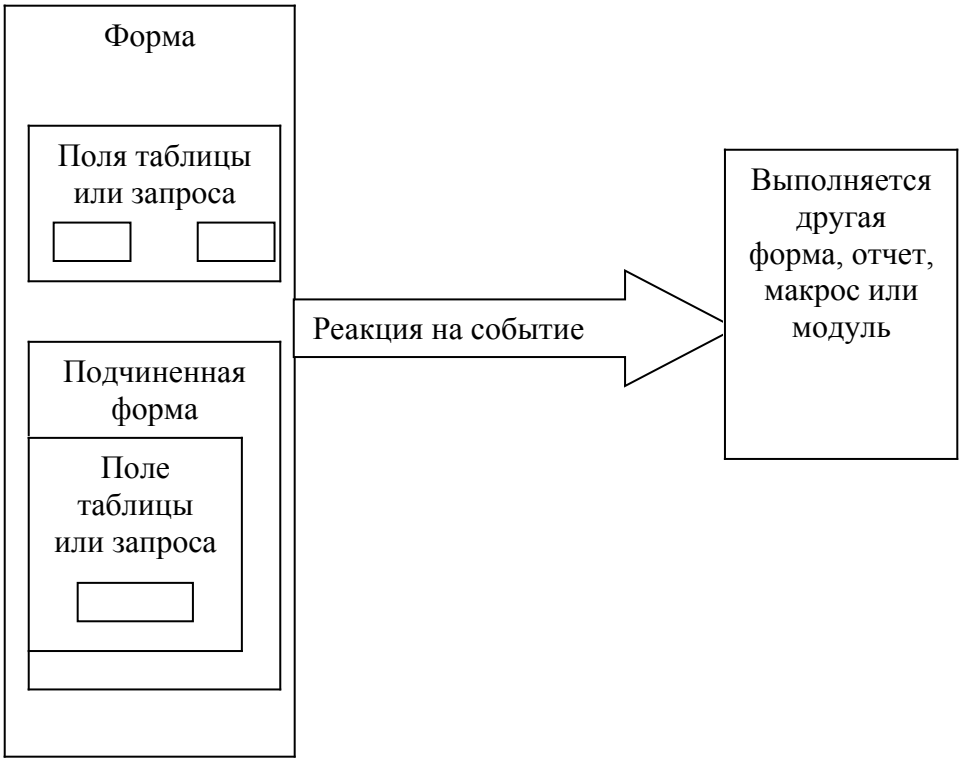


Рис.15. Общая структура формы Microsoft Access.

Формы используют базовые таблицы или запросы, также они могут включать в себя и другие формы. Подчиненные формы в свою очередь определяют действия над другими формами, отчетами или запросами. Формы могут запускать другие формы, модули, макросы или запросы: самый простой способ – с помощью командных кнопок, но могут использоваться и другие действия, например, запуск макроса при щелчке мышью на определенном месте.

2.3.2.2. Инструменты построения форм

Режимы построения форм.

Построение форм может проходить в двух режимах:

1) Режим конструктора.
2) Режим мастера форм

Режим конструктора позволяет создавать формы с нуля. Командным центром по созданию форм в режиме конструктора является панель элементов, которая включает в себя 20 элементов для создания форм, которые позволяют задавать поля, надписи к формам, задавать командные кнопки, создавать подчиненные формы и придавать форме различное цветовое и графическое построение.

Режим мастера форм позволяет создавать формы в первом приближении. Всего существует шесть различных мастеров:

1) Мастер форм.

Позволяет создавать как простые, так и подчиненные формы.

2) Автоформа в столбец.

Позволяет создавать форму в виде столбца.

3) Автоформа ленточная.

Позволяет создавать ленточную форму.

4) Автоформа табличная.

Создает форму в виде таблицы.

5) Мастер диаграмм.

Позволяет построить форму с включением в нее диаграммы, для которой определяет исходные данные и все настройки диаграммы.

6) Мастер сводной таблицы.

Позволяет создавать форму с внедренным объектом Microsoft Excel.

Свойства форм.

Сама форма, каждый ее раздел (область данных, заголовок, примечание), а также элементы управления самой формой обладают свойствами, которые выставляются разработчиком.

Для просмотра свойств, имеется окно свойств, которое включает в себя несколько видов вкладок:

1) Свойства макета.

Выводятся свойства, характеризующие сам объект, которому принадлежат данные свойства, такие как элементы управления, заголовок, область данных.

2) Свойства данных.

Характеризуют данные, используемые в объекте, их тип, источник и т.д.

3) Свойства событий.

Характеризуют отклики на события, которые должны произойти.

4) Все свойства.

Выводят все свойства различных категорий.

Любая форма может иметь более 70 настраиваемых свойств, а большинство элементов управления форм более 30. Удобным средством для задания значений свойств являются различного рода построители, в которых используются сложного вида выражения, определяющие значения свойств.

Построитель выражений работает в диалоговом режиме предоставляя пользователю различные варианты значений, из которых он выбирает необходимые.

Одним из таких свойств является свойство доступа и блокировки элементов управления. Если при построении формы разработчик не хочет, чтобы пользователи имели доступ к каким – то элементам управления форм или не могли их обновлять, то применяются свойства доступа и блокировки. В зависимости от выставляемых сочетаний различают режимы, которые приведены в таблице 12.

Таблица.12. Свойства доступа и блокировки.

Доступ	Блокировка	Описание
Да	Да	Элемент управления является доступным, данные выводятся обычным образом: их можно копировать, но нельзя изменять.
Нет	Нет	Элемент управления недоступен, элемент управления и данные выводятся серым цветом.
Да	Нет	Элемент управления доступен, данные могут быть скопированы и изменены.
Нет	Да	Элемент управления недоступен, данные не копируются и не изменяются, вывод производится в обычном виде.

2.3.3. Главная (основная) кнопочная форма.

Обычно последней создаваемой формой является главная кнопочная форма, предоставляющая пользователю прямой доступ к основным объектам приложения. Основная кнопочная форма, может быть простой формой с логотипом, заголовком и несколькими командными кнопками, открывающими основные формы приложения. Создание главной кнопочной формы может осуществляться в режиме конструктора, задавая данной форме определенные свойства, связанные с определением командных кнопок.

Построение всех кнопочных форм, необходимых для управления сложным приложением, является довольно трудоемким процессом, поэтому в Microsoft Access предусмотрена специальная надстройка “Диспетчер кнопочных форм”, которая занимается построением всех кнопочных форм. Эта надстройка применяет сложную, с точки зрения программирования, технику для управления всеми кнопочными формами с помощью одной формы и использует специальную таблицу – драйвер « Элементы кнопочной формы », что позволяет определить любое число кнопочных форм и создать до восьми командных кнопок в каждой из них.

Кнопочная форма, управляющая остальными кнопочными формами, является главной кнопочной формой.

В главной кнопочной форме следует создать командные кнопки, открывающие другие формы, и включить кнопку для выхода из приложения.

В каждой дополнительной кнопочной форме нужно предусмотреть одну кнопку для возврата на предыдущий уровень в иерархии кнопочных форм или для перехода в основную кнопочную форму. Пример этого показан на рис.16.

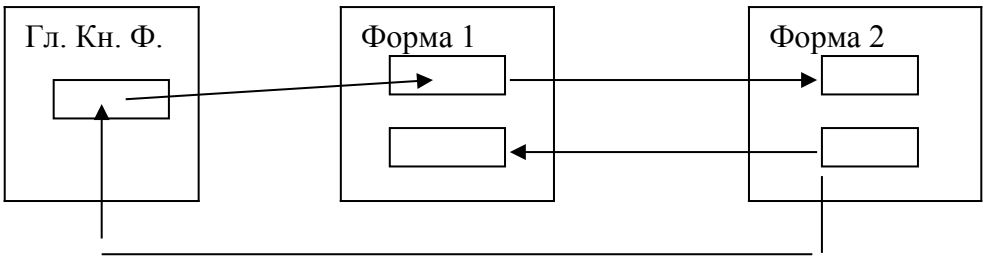


Рис.16. Переход по кнопочным формам.

2.3.4 Разработка отчетов.

Использование и виды отчетов.

Отчеты представляют собой средства представления информации из базы данных в виде печатного документа. По сравнению с другими методами вывода информации отчеты обладают двумя преимуществами:

1) Они предоставляют широкие возможности для группировки и вычисления промежуточных и общих итогов для больших наборов данных.
2) Отчеты могут быть использованы для оформления различных счетов, бланков заказов, почтовых наклеек, материалов для презентаций.

В отчете можно использовать следующие возможности:

1) Для обеспечения иерархического представления данных можно задать до 10 уровней группировки. Например:

4. Название ВУЗа.
3. Факультет.
2. Кафедра.
1. Номер группы.

ФИО (сумма стипендии).

2) Для каждой из групп можно задавать отдельные заголовки и примечания.
3) Можно производить сложные вычисления не только внутри одной группы, но и по нескольким группам одновременно.
4) Помимо верхнего и нижнего колонтитула отчет может включать в себя заголовок и примечание.
5) В любой раздел отчета можно включать различные рисунки и диаграммы.

Среди разновидностей отчетов можно выделить следующие:

1) Простые отчеты.

Информация, в которых выдается последовательно.

2) Многостраничные отчеты.

Используются для вывода большого количества информации, принадлежащей одной записи.

3) Подчиненные отчеты.

Используются для вспомогательных операций и передают информацию основным отчетам. Например, расчет каких – то значений может проводиться в подчиненном отчете, а сами значения выводятся в основном.

Режимы создания отчетов.

Для создания отчетов используются два режима:

1) Режим конструктора.
2) Режим мастера.

Режим конструктора построения отчетов практически совпадает с режимом конструктора форм. Точно также имеется панель элементов управления, позволяющая задавать, такие же элементы как и в формах. Основным отличием при построении отчетов является возможность задания группировок внутри самого отчета, поэтому в общем виде структура отчета может быть представлена в виде рис.17.

Заголовок отчета
Верхний колонтитул
Заголовок группировки 1
Заголовок группировки … 10
Область данных
Примечание группировки 1
Примечание группировки … 10
Нижний колонтитул
Примечание

Рис.17. Структура отчета в общем виде.

С помощью режима мастера можно создавать отчеты на основе возможностей, имеющихся в Microsoft Access. Существуют следующие режимы мастеров:

1) Мастер отчетов.

Позволяет разрабатывать отчеты, задавать форматы представления данных в отчетах, задавать уровни группировки в отчетах и вводить итоговые функции.

2) Автоотчет в столбец.

Создает отчет, в котором записи базового запроса или таблицы представлены в виде простого списка.

3) Автоотчет ленточный.

В отчете этого типа данные записи базового запроса или таблицы выводятся в одной строке.

4) Мастер диаграмм.

Помогает построить диаграмму и создает в отчете свободную рамку объекта OLE с внедренной диаграммой Microsoft Graph.

5) Почтовая наклейка.

Позволяет выбрать поля с именами и адресами, отформатировать их и создать отчет для печати почтовых наклеек.

2.4. Создание приложений.

2.4.1 Общие сведения о макросах.

Функции макросов.

Основным преимуществом макросов в Microsoft Access является то, что они могут выполняться в ответ на многие события. Макросы используются для выполнения следующих функций:

1) Открытие и закрытие таблиц запросов, форм или отчетов в любом доступном режиме.
2) Для выполнения запроса на выборку или запроса на изменение.
3) Для выполнения действий в зависимости от значений в форме или отчете или самой базе данных.
4) Для установки значения любого элемента управления формы или отчета.
5) Для использования фильтра и поиска данных в таблице или запросе.
6) Для определения специальной строки меню, заменяющей стандартную.
7) Для выполнения команд как стандартного, так и специального меню.

8) Для перемещения, изменения размеров, сворачивания или восстановления любого окна внутри рабочей области Microsoft Access.

9) Для вывода на экран информационных сообщений.

10) Для переименования любого объекта базы данных, а также для копирования объектов.

11) Для запуска приложения, а также для обмена данными с использованием буфера или механизма DDE.

Создание макроса.

Microsoft Access позволяет создавать макросы, состоящие как из одной макрокоманды, так и из нескольких.

Задав такую макрокоманду, присваивается имя макроса, в котором будет выполняться такая макрокоманда. Кроме этого можно задать условие, по которому будет выполняться указанная макрокоманда.

Макрос, состоящий только из одной макрокоманды, называется простым макросом.

Макрос, в котором указаны несколько команд и задан порядок их выполнения, называется сложным макросом.

При выполнении макроса Microsoft Access не всегда ждет завершения работы макрокоманды. Например, макрокоманда “ открыть форму ” может начаться открытием формы и если форма содержит большое количество информации, то не дожидаясь вывода всей информации, Microsoft Access может запустить следующую макрокоманду, т.е. макросы могут выполняться параллельно. Это позволяет сделать операционная система Windows, которая является многозадачной.

В сложных приложениях используются, как правило, сотни макросов.

Для того чтобы их сгруппировать существует две возможности:

1) Группировка макросов, относящихся только к одному объекту.

2) Группировка макросов по функциональному признаку, т.е. группируются макросы, выполняющие одни и те же действия.

Обзор макрокоманд.

Макросы подразделяются на несколько классов, сформированных по функциональному признаку:

1) Открытие и закрытие таблиц, форм, отчетов и запросов.
2) Печать данных.
3) Выполнение запросов.
4) Проверка истинности условий и управление выполнением макрокоманд.
5) Установка значений в элементах управления форм и отчетов.
6) Поиск данных.
7) Команда для построения специального меню.
8) Управление выводом на экран и активизация окон.
9) Выдача сообщений пользователю.
10) Импорт и экспорт объектов, а также их копирование, переименование и сохранение.
11) Запуск других приложений операционных систем MS DOS и Windows.

Обзор событий форм и отчетов.

Microsoft Access предоставляет более 40 свойств событий форм и отчетов, позволяющих запускать макросы или модули:

1) Открытие и закрытие форм и отчетов.
2) Изменение данных в элементах управления форм и отчетов.
3) Смена фокуса элемента управления формы или отчета (переход с одного элемента управления на другой).
4) Использование фильтров.
5) При работе с мышью и клавиатурой.
6) Печать данных.
7) Активизация специального меню для формы, отчета или приложения.
8) При отслеживании ошибок.
9) По истечении промежутка времени.

2.4.2 Основные понятия процедур VBA для приложений.

Применение макросов.

Несмотря на широкие возможности макрокоманд, у них есть свои ограничения, связанные в частности с невозможностью анализировать ошибки внутри макросов или восстанавливать работу приложения после обнаружения ошибки. В связи с этим макросы могут использоваться в следующих случаях:

1) Когда не требуется отслеживать и обрабатывать ошибки.
2) Приложение содержит небольшое количество форм и отчетов.
3) Приложение будет использоваться непрофессионалами, которые сами могли бы расширить его функции за счет использования макросов.
4) При разработке прототипа приложения и для демонстрации проекта, когда необходимо быстро автоматизировать решение некоторых задач.

Существует две проблемы, которые чаще всего решаются с помощью макросов:

1) Определение альтернативных действий для определенных нажатий клавиш.
2) Управление запуском приложения при открытии базы данных с помощью макроса Autoexec.

Применение модулей.

Модули используются в следующих случаях:

1) Нужно отслеживать и обрабатывать ошибки в приложении.
2) Необходимо создать нестандартную функцию.
3) Во время работы приложения необходимо создавать новые объекты.
4) Созданное приложение должно взаимодействовать с другими приложениями Windows.
5) Необходимо обрабатывать отдельные записи, а не наборы записей.
6) Необходимо создать приложение с высокой производительностью.

Это связано с тем, что модули в отличии от макросов компилируются вместе со всей программой и выполняются быстрее.

2.4.3 Язык VBA.

VBA для приложений заменил язык программирования Access Basic, использовавшийся в первых версиях Microsoft Access.

На данный момент язык VBA является общим языком программирования для приложений Microsoft Office, включая Word, Microsoft Access и Excel.

Наличие общего языка программирования для нескольких приложений обеспечивает ряд преимуществ: зная один язык программирования можно интегрировать объекты в другие приложения с помощью Visual Basic и механизма OLE.

Достоинством языка Visual Basic является более развитая среда программирования по сравнению с предыдущим языком. Также он имеет улучшенные средства отладки, которые помогают убедиться, что разработанная программа функционирует должным образом.

Язык VBA относится к языкам объектно–ориентированного программирования и обладает всеми возможностями языков высокого уровня.

Язык VBA позволяет поддерживать 12 типов данных, как простых – целое число, так и сложных – объект. Типы данных представлены в таблице 13.

Язык VBA содержит инструкции описания переменных:

CONST – для описания констант

DIM – для описания локальных переменных

PUBLIC – для описания глобальных переменных.

инструкции переходов в программах:

GOTO – безусловный переход

IF…THEN…ELSE – условный переход

инструкции, задающие функции:

FUN – задание функции для своей процедуры

SUB – задание функции для всех процедур

инструкции вызова подпрограмм:

CALL - вызов подпрограммы.

Кроме приведенных выше инструкций существует еще целый ряд инструкций, позволяющий писать программы любой степени сложности.

2.4.4. Модули

Все написанные на языке VBA программы хранятся в модулях.

База данных может содержать два вида модулей:

1) Стандартные, являющиеся объектами самой базы данных.
2) Модули форм или отчетов, являющиеся частью этих объектов, иначе их называют модули класса.

Стандартные модули используются для создания и хранения процедур, выполняемых из запросов или из нескольких форм или отчетов приложения.

Процедуры, определенные в стандартном модуле, могут вызываться из любого места приложения.

Чтобы облегчить создание процедур обработки событий, возникших в форме или отчете, Microsoft Access позволяет создать модули, относящиеся к этим формам или отчетам. В модуле класса или отчета содержаться специальным образом именованные процедуры обработки событий.

Таблица. 13 Типы данных языка VBA.

Тип данных	Размер	Информация
BYTE	1 байт	Положительные целые числа в диапазоне от 0 до 255
INTEGER	2 байта	Целые числа в диапазоне от -32768 до 32767
LONG	4 байта	Целые числа в диапазоне от -2147483648 до 2147483647
	4 байта	Числа с плавающей точкой, принимающие значения от $-3,4*10^{38}$ до $3,4*10^{38}$
DOUBLE	8 байт	Числа с плавающей точкой, принимающие значения $-1,79*10^{308}$ до $1,79*10^{308}$
CURRENCY	8 байт	Числа с фиксированной десятичной запятой и четырьмя цифрами в дробной части в диапазоне -922337203685477,5808 до 922337203685477,5807
STRING	10 байт плюс 1 байт на символ	Любая текстовая или двоичная строка длиной примерно до 2 миллиардов байт, включая данные полей MEMO и фрагменты OLE. Строка фиксированной длины может содержать до 65400 символов.
BOOLEAN	2 байта	0 или 1
DATE	8 байт	Значения даты/времени в диапазоне от 1 января 100г. до 31 декабря 9999.
OBJECT	4 байта	Любая ссылка на объект
VARIANT	От 16 байт до примерно 2 миллиардов байт	Любые данные, включая NULL , EMPTY и даты/времени
Тип данных определяемый пользователем	Зависит от определяемых элементов	Любое число переменных с любыми из перечисленных выше типами данных

Данные процедуры могут быть двух видов:

1) Личные – работают только для данной формы или отчета.

2) Общие процедуры (методы класса) – могут использоваться для других объектов помимо данной формы или отчета.

Модули класса имеют 3 преимущества перед стандартными модулями:

1) Процедуры, необходимые для автоматизации работы формы или отчета хранятся вместе с формой или отчетом и поэтому нет необходимости запоминать имя модуля, в котором находятся эти процедуры.

2) Microsoft Access загружает стандартный модуль базы данных при первом обращении к нему и оставляет его в памяти до закрытия базы данных. Модуль класса загружается только при открытии соответствующего объекта и убирается из памяти вместе с его закрытием. Соответственно уменьшается размер занимаемой памяти.

3) При экспорте вместе с формой или отчетом экспортируются все программы, содержащиеся в модуле класса.

Наряду с преимуществами модуль класса имеет один существенный недостаток: при открытии формы или отчета, обладающего соответствующим модулем, процесс открытия будет заметно дольше, чем открытие формы или отчета, не имеющего данного модуля.

3. Виды корпоративных СУБД.

3.1 СУБД MS SQL Server.

3.1.1 Различие между MA и MS SQL Server.

MS SQL Server - это реляционная СУБД построенная по архитектуре клиент-сервер. MS SQL Server ориентирован на использование в операционных системах (ОС) Windows NT/2000 и использует в своей работе системные функции этих ОС, что значительно упрощает архитектуру MS SQL Server, в отличие от других СУБД вынужденных дублировать некоторые функции ядра операционной системы, для обеспечения межплатформенной переносимости. За счет такой тесной интеграции с Windows NT/2000, СУБД MS SQL Server работает на всех платформах, для которых реализована Windows NT/2000 (Intel, Alpha, PowerPC и др.), поддерживает большое количество сетевых протоколов (TCP/IP, IPX/SPX, Named Pipes (NetBEUI), Вапуап Vines, AppleTalk (ADSP), DECNet), поддерживает многопроцессорность и др.

MS SQL Server является разработкой корпорации Microsoft (последней версией, на середину 2002 г. является MS SQL Server 2000, широко используется также версия MS SQL Server 7.0). Поскольку СУБД MS Access также является разработкой этой корпорации, то очевидно, что между этими двумя продуктами существует некоторая общность. Так интерфейс конструктора таблиц, ввода данных в них, описания схемы данных и т.д. в MS SQL Server, в значительной мере напоминает MS Access. MS Access может также использоваться и в качестве клиентской части MS SQL Server (Меню Файл/Внешние данные/Связь с таблицами/ Тип файлов: Базы данных ODBC/Кнопка New/Выбрать драйвер MS SQL Server). Впрочем, MS Access может использоваться и в качестве клиента для доступа к базе данных Oracle и др. базам данных.

Однако SQL Server - это не просто улучшенный Access. Между этими СУБД есть принципиальная разница. Если Access базируется на архитектуре файл-сервер и приемлем для небольших баз данных и небольшого числа пользователей (офисная СУБД), то SQL Server базируется на архитектуре клиент-сервер и применяется в базах данных среднего размера и большим числом пользователей (профессиональная СУБД). Даже по тем позициям, где возможности Access и SQL Server схожи, SQL Server значительно превосходит Access. Так, например, и в Access и в SQL Server реализована система разграничения доступа к объектам базы данных (разные пользователи имеют разные права по работе с различными таблицами, запросами и т.д.). Однако в SQL Server ограничения доступа можно выставлять не только на таблицу в целом, но даже и на отдельные ее столбцы. Также в SQL Server поддерживается механизм ролей. Роль - это набор прав доступа к объектам базы данных. Роли для каждой базы данных можно определять самостоятельно или пользоваться заранее определенными ролями. Например, роль администраторы безопасности (security admin) - это пользователи которые могут допускать других пользователей к работе с базой данных, роль создатели базы данных (db creators, database creators) - пользователи которые могут создавать и изменять структуру базы данных и т.д. Используя роли можно быстро и удобно разграничить доступ между пользователями, предоставив им только те права, которые действительно необходимы. Причем нарушения прав доступа, также как и сама работа SQL Server будут протоколироваться в специальных log-файлах. SQL Server также позволяет пользователям, правильно указавшим свой пароль при входе в сеть (домен Windows NT/2000), повторно не вводить пароль при доступе к базе данных (Windows authenticatio(1 mode).

К другим существенным отличиям относится то, что если в Access хранимые процедуры и функции пишутся на VBA (Visual Basic for Applications), то в SQL Server основным языком написания хранимых процедур является Transact SQL (хотя могут использоваться и внешние

хранимые процедуры, написанные на других языках). Transact SQL - это диалект SQL, в котором, помимо стандартных инструкций SQL, предусмотрены возможности организации циклов, условного выполнения операторов и т.д.

Также необходимо упомянуть, что существующие базы данных Access (также как и файлы Excel, базы данных Paradox, Visual FoxPro и др.) легко импортируются в SQL Server, так что переход на SQL Server относительно прост.

3.1.2 Объекты MS SQL Server.

MS SQL Server может содержать несколько баз данных. На одном компьютере может быть установлено несколько экземпляров MS SQL Server. Каждая база данных содержит следующие объекты:

1) таблицы
2) индексы
3) представления (виды, запросы)
4) диаграммы (схема данных)
5) курсоры (окно накладываемое на набор данных, аналог окна в текстовом редакторе, в котором в каждый момент времени отображается только часть документа)
6) триггеры, хранимые и внешние процедуры, функции пользователя, пользовательские типы данных
7) правила (ограничения на значения столбцов) и умолчания (значения столбцов по умолчанию) необходимо для совместимости со старыми версиями, в новых версиях все эти ограничения указываются в инструкции Create языка SQL..
8)перечень пользователей, допущенных к базе данных и их разрешения, роли пользователей
9) каталоги полнотекстного поиска (позволяет осуществлять поиск текста по столбцам таблицы, включая столбцы, в которых в качестве значений содержатся файлы .doc, .xls, .txt, .htm)

Управление SQL Server 2000 осуществляется через оснастку Enterprise Manager MMC (Microsoft Management Console), которая по внешнему виду напоминает проводник (рис. 18). Эта оснастка, помимо непосредственной работы с конкретной базой данных (создание таблиц, импорт или ввод данных в таблицы, создание пользователей и определение их прав доступа и т.д.), позволяет выполнять много дополнительных операций: создавать расписание обслуживания базы данных (maintenance plan: проверка целостности БД, реорганизации: свободного места в БД, резервное копирование БД и журналов транзакций и др.), создавать перечень операторов, ответственных за обслуживание БД (имя, адрес электронной почты или номер пейджера, время работы) и определять ошибки в базе данных (alerts), при которых данному оператору будет направлено сообщение.

Репликация баз данных также настраивается через Enterprise Manager MMS. Репликация - это распространение копий базы данных на другие компьютеры в сети (например, пользователям других сетей, или пользователям мобильных компьютеров). Репликация в MS SQL Server 2000 может быть следующих типов:

.1) Snapshot replication (репликация снимков): копии соответствующих таблиц передаются другим базам данных как снимок (snapshot) их текущего состояния, и не обновляются автоматически. Пользователи других баз данных не могут изменять содержимое снимка.

. 2) Transactional replication (репликация транзакций): копии соответствующих таблиц передаются другим базам данных. При изменениях в исходной базе данных, эти копии обновляются автоматически. Пользователи других баз данных не могут изменять содержимое копии.

.3) Merge replication (объединяющая репликация): копии соответствующих таблиц передаются другим базам данных. Причем как пользователи основной базы данных, так и

пользователи баз данных, на которых находятся копии, могут изменять данные. Все изменения в исходный вариант и копии будут внесены автоматически.

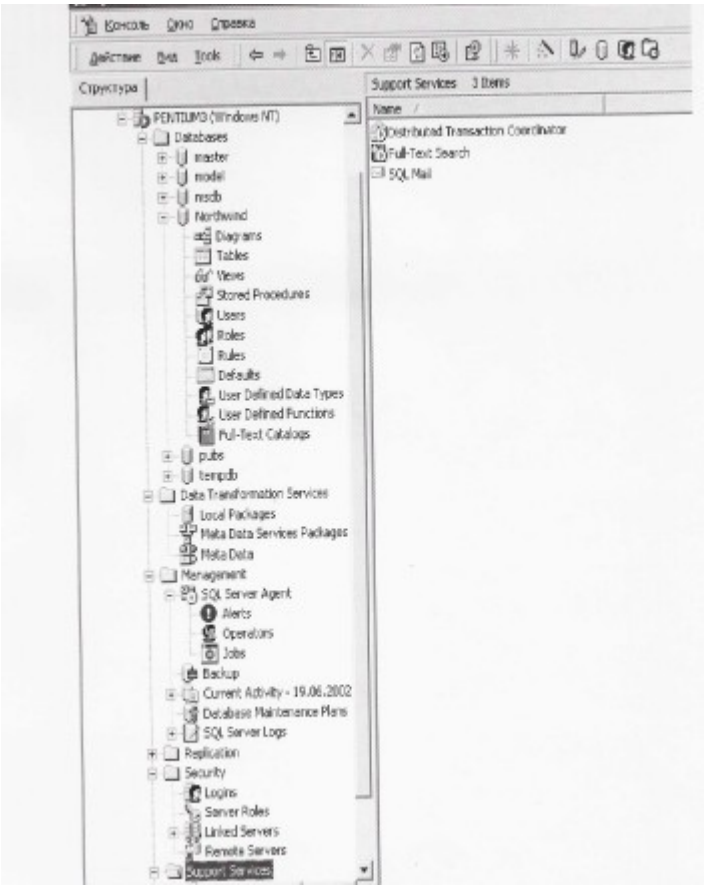


Рис. 18 Управление СУБД SQL Server 2000 при помощи Enterprise Manager MMC.

Помимо Enterprise Manager MMC, существует целый ряд других утилит, позволяющих управлять работой SQL Server. Используя утилиту Profiler можно отслеживать работу SQL Server, анализировать и настраивать его производительность. Утилиты Server Network Utility и Client Network Utility позволяют настроить список протоколов, поддерживаемых на сервере и клиентах (например, включить шифрование передаваемых данных). Утилита Analysis Manager MMC (пакет MS SQL Analysis Services устанавливается отдельно) позволяет использовать MS SQL Server 2000 не только как систему оперативной обработки транзакций (OLTP-систему), но и как систему делового анализа (хранилище данных, OLAP-систему). Помимо утилит с графическим интерфейсом, MS SQL Server может управляться и при помощи утилит командной строки, что позволяет настраивать SQL Server из файлов-сценариев.

Важным моментом является и то, что MS SQL Server 2000 представляет средства интеграции с сервисами сети Internet. В частности, запросы к SQL Server можно направлять непосредственно по протоколу http (используя внешний Web-сервер, например IIS). Кроме того, в MS SQL Server 2000 поддерживается язык XML, наилучшим образом подходящий для представления структурированных данных в Web-броузерах.

3.2 СУБД Oracle.

3.2.1 История развития СУБД Oracle.

СУБД Oracle является на сегодняшний день самой мощной, многофункциональной и легко масштабируемой СУБД, построенной по архитектуре "клиент-сервер", поддерживающей практически все существующие платформы. Это прекрасный выбор для крупной организации: первоначальные затраты на установку (лицензия, приобретение высокопроизводительных серверов) в будущем обернутся значительной экономией средств при необходимости расширения базы данных. Для небольшой организации мощь Oracle может оказаться чрезмерной, в таком случае можно рекомендовать использование Microsoft SQL Server (Windows NT/2000) или PostgreSQL (Linux/Unix). Для фирм малого бизнеса стандартом остается СУБД MS Access. Тем не менее, Oracle продолжает занимать значительную долю рынка, являясь пожалуй самой передовой СУБД.

Первая коммерческая версия СУБД Oracle 2.0 (версия 1.0 не поставлялась заказчикам) была создана фирмой Relational Software Incorporated (позднее переименована в Oracle Corporation) на миникомпьютере PDP-ll фирмы Digital Equipment Corporation в операционной среде RSX-II. Отличительные черты этой и последующих версий представлены в таблице 14.

3.2.2 Инструменты и технологии, реализованные в СУБД Oracle.

Поддержка языка PL/SQL

Язык PL/SQL является диалектом SPL-SQL (Stored Procedure Language - SQL) - модифицированным вариантом языка SQL, использующимся в СУБД для описания триггеров, хранимых процедур и функций. Если в традиционном SQL отсутствуют циклы, операторы IF..THEN..ELSE и т.д., то в языках SPL-SQL все эти возможности имеются, что позволяет писать на них "полноценные" программы, хранимые в самой СУБД. Другим широко известным диалектом SPL-SQL является язык Transact-SQL, реализованный в СУБД MS SQL Server и СУБД Sybase.

Программа SOL *NET

Программа SQL *Net позволяет пользователю взаимодействовать с удаленной базой данных (или несколькими базами данных) через сеть так, как будто бы это была локальная база данных, находящаяся на компьютере пользователя. SQL *Net устанавливается и на сервере и на клиенте. Задача администратора – настроить ее, создать именованный канал связи между локальной и удаленной базой данной. Именованный канал связи определяет: местоположение удаленного сервера БД в сети, используемый коммуникационный протокол, имя удаленной базы данных, имя и пароль пользователя для подключения к БД. После создания именованного канала связи доступ к удаленной базе данных из инструкций SQL осуществляется в виде:

ИМЯ _ ТАБЛИЦЫ @ имя_канала.

SQL*Net поддерживает практически все сетевые протоколы: TCP/IP, SPXlIPX, DECnet, APPC, ISO!OSI, Named Pipes, NetBIOS, AppleTalk, Banyan Vines,. XODIAC, Wang Systems Network и др. Механизм многопротокольного обмена - MPI (Multiple Protocol Interchange) осуществляет преобразование данных из одного протокола в другой и позволяет

Таблица 14. История развития СУБД Oracle

Версия	Отличительные черты
Oracle 2.0 1979 г	Большая часть СУБД написана на ассемблере PDP-ll, а отдельные компоненты - на языке Си. Реализована межплатформенная переносимость (например, помимо "родной" ОС RSX- 11, работала в ОС UNIX). Полная реализация нового (на то время) языка запросов SQL (по стандарту IBM). Отсутствие поддержки транзакций (как следствие, пользователи были вынуждены часто делать резервные копии базы данных во избежание потерь информации при сбоях в процессе обновления базы данных).
Oracle 3.0 1983 г.	Полностью написана на языке Си, что обеспечило последующий перенос Oracle на широкий спектр платформ. Существенно улучшена эффективность системы и введена поддержка транзакций.
Oracle 4.0 1984 г.	Существенно повышена надежность и проработан механизм блокировок, для обеспечения непротиворечивости данных, при одновременной работе с СУБД нескольких пользователей Выполнен перенос Oracle 4.0 на большие компьютеры в ОС VM и MVS, а также на персональный компьютер с 640 Кбайтами оперативной памяти.
Oracle 5.0 1985 г.	Впервые введена архитектура "клиент/сервер" и реализована программа SQL *Net, поддерживающая неоднородные среды.
Oracle 5.1 1986 г.	Впервые разрешены распределенные запросы: у пользователя создавалась иллюзия работы с единой базой данных, хотя физически его запрос направлялся к различным базам данных, находящимся на различных компьютерах.
Oracle 6.0	Ориентирована на построение крупномасштабных информационных систем, позволяющих обрабатывать транзакций в режиме реального времени. Существенно улучшен механизм блокировок (введена блокировка на уровне записи, введено несколько уровней блокировки, что позволило процессам чтения и записи больше не блокировать друг друга). Частично реализован механизм описания ограничений ссылочной целостности на уровне схемы базы данных. Введена возможность совместного использования языка SQL с процедурным язы ком PL/SQL (получающиеся в результате утверждения могли посылаться для выполнения на сервер, как анонимные процедуры). В язык PL/SQL введен язык четвертого поколения на системы OS/2, Xenix, Вапуап Vines и Macintosh.
Oracle 7.0	Повышение эффективности ввода/вывода, использования центрального процессора и работы с памятью, оптимизатор запросов, разделяемый SQL-кэш на сервере (SQL-инструкции, ко торые ранее уже были получены от клиентов, проанализированы и скомпилированы, хранят ся некоторое время в кэш-памяти, экономя время и память при обработке последующих аналогичных SQL-инструкций). Полностью реализован механизм описания ограничений ссылочной целостности на уровне схемы базы данных (каскадные удаления и обновления записей). При помощи PL/SQL реализованы хранимые процедуры и триггеры. В области администрирования введены: зеркальный журнал транзакций, динамическое создание журналов данных, команда ANALYZE, позволяющая собирать статистику об использовании таблиц, индексов и других физических объектах (используется оптимизатором запросов), профили пользователей на использование системных ресурсов (процессора, ввода/вывода и т.д.), роли пользователей. Роль - это совокупность прав доступа к объектам базы данных (1NSERT, UPDATE, SELECT и др.) и системных прав

	(CREATE TABLE, ALTER SYSTEM и т.д.). Определив роль, администратор базы данных может с помощью одной команды GRANT дать пользователю привилегии для работы с некоторым приложением.
Oracle 7.1	Полностью поддерживает стандарт ANSl SQL 89 и входной уровень ANSl SQL 92. Достигнута полная интеграция PL/SQL и SQL. Если ранее можно было использовать конструкции SQL в текстах PL/SQL (но не наоборот), то теперь функции PL/SQL могут вызываться из SQL как обычные встроенные функции в любых SQL-инструкциях (SELECT, WHERE, GROUP BY, CREATE TABLE, ALTER TABLE и др.). Реализована возможность динамически формировать и выполнять инструкции SQL в программах PL/SQL, т.е. реализо ван динамический SQL (благодаря встроенному пакету DBMS_SQL). Реализована возмож ность параллельной обработки запросов (опция paralle1 query), параллельного создания ин дексов и параллельной загрузки данных (параллельная работа сразу нескольких серверных процессов, под управлением процесса-координатора). В области распределенной обработки данных введена асинхронная симметричная репликация данных и асинхронный вызов удаленных процедур.
Oracle 8.0 1997 г.	Первая объектно-ориентированная версия СУБД Oracle (использование объектной модели для доступа к данным, вызова внешних процедур на языках C, C++, Java, как методов объек тов и т.д.). Поддержка языка Java: встроенная в СУБД виртуальная Java-машина и техноло гия SQLJ дЛЯ возможности внедрения инструкций SQL в Java-код. Как следствие - хранимые процедуры и триггеры на языке Java, взаимодействие Java - SQL - PL/SQL. Драйвер JDBC - набор стандартных API-функций, позволяющих внешним программам на языке Java взаимодействовать с базой данных при помощи SQL-инструкций. Упрощены процедуры резервного копирования, восстановления и репликации базы данных. Введена возможность разделять большие таблицы и индексы на несколько небольших самостоятельных частей (partitioning). В результате появляется возможность обслужива ния/восстановления крупных таблиц "по частям", в случае утраты одной части (из-за физи ческих проблем диска) остальные части таблицы продолжают оставаться корректными - данные всей таблицы не теряются. Улучшена параллельная обработка данных, возможность оперативной обработки транзакций (OL TP, Online Transaction Processing), улучшена работа с данными типа мультимедиа.
Oracle 8.1 1999 г.	Версия, ориентированная на более тесную интеграцию в сеть Intemet. Интегрированный в СУБД Web-сервер (на основе Apache) -СУБД может сама принимать HTTP-запросы и обрабатывать их без помощи внешнего Web-сервера. Акселератор (JIT-компилятор) для Java-машины - байт-код Java компилируется в выполнимый код для конкретной платформы, что значительно увеличивает быстродействие. Поддержка JavaServer Pages, сервлетов Java и компонентов Java Beans. Поддержка языка XML. PL/SQL шлюз для протокола HTTP. Защита информации при помощи криптографических алгоритмов RC4 (256-бит) и TripleDes, поддержка протокола SSL, значительно повышена защищенность и стабильность самой СУБД.
Oracle 9 2002 г	Было сделано более 400 изменений, по сравнению с предыдущей версией, обеспечивающих улучшение работы СУБД.

существенно снизить число, используемых протоколов в сети.

SQL *Net поддерживает также множественные сетевые интерфейсы для одного узла.

Другой существенной функцией SQL *Net является поддержка национальных языков, если на клиентах и серверах используются разные кодовые страницы, то SQL *Net обеспечивает корректную перекодировку данных.

Программа SQL *PLUS

Программа SQL *Plus позволяет интерактивно, из командной строки, взаимодействовать с БД Oracle при помощи инструкций SQL (рис. 19). Программа позволяет:

1) выполнять в построчном режиме большинство SQL-операторов и блоки PL/SQL;

2)создавать или изменять таблицы и представления, а также просматривать их содержимое в окне SQL *Plus в форме отчета;

3) просматривать определения столбцов для любой таблицы;

4) выполнять отладку SQL-операторов, хранимых процедур и функций, блоков PL/SQL перед их использованием в разрабатываемых приложениях;

5) получать доступ и копировать данные между различными SQL-базами данных;

выполнять встроенные команды SQL *Plus.

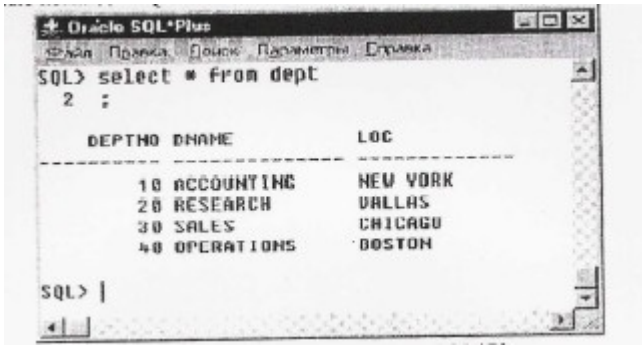


Рис. 19 Пример запроса в окне SQL *Plus.

ORACLE Developer

ORACLE Developer - это мощная интегрированная среда разработки приложений, взаимодействующих с БД Oracle по архитектуре клиент/сервер. Причем поддерживается создание приложений, как по архитектуре клиент - сервер, так и по трехуровневой архитектуре (клиент-сервер приложений - сервер БД). Пример трехуровневой архитектуры, с использованием Oracle Fonns Server приведен на рис. 20. Наличие интегрированного в Oracle Web-сервера позволяет реализовать трехуровневую архитектуру и без использования Forms Server, однако это потребует больших затрат времени программистов.

Рис. 20 Трехзвенная архитектура, с использованием Oracle Fonns Server.

В состав среды ORACLE Oeveloper входят следующие программы:
- Form Builder
- Report Bu]der
- Graphics Builder
- Procedure Builder
- Project Builder

Form Builder - это интегрированная среда для разработки программ, которые позволяют получить доступ к БД Oracle, при помощи удобных форм пользователя. Такой подход чем-то напоминает формы MS Access, с той разницей, что Form Builder позволяет создавать самостоятельные программы-клиенты, выполняющиеся на компьютере пользователя. Эти программы содержат собственное меню, окна, элементы управления, триггеры, срабатывающие при определенных действиях пользователя, и процедуры, написанные на PL/SQL. Процедуры могут определяться и быть доступными как внутри одной формы, так и группироваться в библиотеки процедур, используемые несколькими формами.

Report Builder - это интегрированная среда разработки отчетов, позволяющая при помощи удобных мастеров строить практически любые отчеты (табличные, главный-подчиненный и др.), с использованием различных текстовых и графических объектов и процедур PL/SQL. С помощью шлюзов можно строить отчеты из данных, находящихся в базах данных других производителей (например, OB2).

Graphics Builder - средство визуализации данных БД Oracle, поддерживает языки SQL и PL/SQL, импорт и экспорт данных и графики из различных источников.

Procedure Builder - позволяет создавать и отлаживать процедуры на языке PL/SQL.

Project Builder - позволяет объединить в едином проекте все средства Oracle Developer (Form Builder, Report Builder. Graphics Builder, Procedure Builder и др.). Он содержит информацию о всех файлах, относящихся к одному проекту и позволяет группе разработчиков скоординировано работать над проектом. Для редактирования файла формы Project Builder вызовет Form Builder, для файла отчета - Report Builder.

Все компоненты (Form Builder, Report Builder, Graphics Builder и др.) полностью интегрированы, и могут передавать и получать данные друг - друга. Все файлы, входящие в проект, компилируются совместно.

К другим инструментам разработчика можно отнести Oracle Book - инструмент создания электронной гипертекстовой документации, которая может вызываться из приложений. Созданные с помощью Oracle Book документы могут находиться в базе данных или в файловой системе, и содержать, помимо текста и графики, видео- и аудио- данные. Oracle Book автоматически конвертирует файлы в более чем 80 форматах текстовых процессоров (MS Word,

WordPerfect и др.), при этом сохраняются шрифты, цвета и выравнивание. Инструмент Oracle Glue ("клей") позволяет включать внешние прикладные программы (например, Excel) в среду разработки Oracle. Доступ к базе данных Oracle из программ на других языках программирования возможен либо с использованием препроцессоров встроенного SQL (Oracle Precompilers), либо через ODBC (или JDBC), либо через интерфейс вызовов функций Oracle (Oracle Call Interface - OCI).

3.2.3 Структура базы данных Oracle.

В базе данных Oracle все таблицы хранятся в единой базе данных, причем таблицы, представления, индексы и др. объекты БД разных пользователей организованы по различным табличным пространствам (tabIespace). При создании объекта базы данных (таблица, представление, индекс и т.д.) внутри табличного пространства, ему выделяется некоторая область - сегмент. При увеличении объекта размер сегмента может увеличиваться на заданный размер, называемый экстентом (extents). Каждый сегмент может состоять из одного или из нескольких экстентов. Каждое табличное пространство может содержать следующие сегменты:

- сегменты данных (data segments);
- сегменты индексов (index segments);
- сегменты отката (roHback segments);
- временные сегменты (temporary segments).

Сегменты отката (roHback segments) используются для хранения первоначального состояния данных, изменяемых при транзакциях. Они позволяют в любой точке транзакции выполнить откат транзакции возврат к первоначальному состоянию. В начале транзакции и в каждой контрольной точке текущее состояние данных копируется в сегмент отката. После завершения транзакции соответствующий элемент сегмента отката становится недействительным. Все изменения данных, выполняемые в рамках транзакции, сохраняются в специальных блоках кэш-буфера и пере носятся в базу данных только при завершении транзакции.

При инсталляции Oracle по умолчанию создается табличное пространство SYSTEM, в котором размещаются служебные таблицы (структура базы данных схематично приведена на рис. 21). Также, для улучшения производительности, сегменты отката и временные сегменты выделяются в отдельные табличные пространства ROLLBACK (сегменты отката) и TEMP (временные сегменты).

Рис. 21 Физическая структура базы данных Oracle

3.2.4 Основные объекты и термины базы данных.

База данных Oracle содержит различные типы объектов. Эти объекты можно подразделить на две категории: объекты схемы и объекты, не принадлежащие схемам. Схема (schema) - это набор объектов различной логической структуры данных. Каждая схема принадлежит пользователю базы данных и имеет одинаковое с ним имя. Каждый пользователь владеет одной схемой. Схема может содержать следующие объекты:

- таблицы;
- индексы;
- кластеры;
- представления (виды);
- снимки - snapshots, журналы репликаций;
- связи базы данных (содержат информацию о подключении к удаленной базе данных);
- последовательности;
- синонимы;
- пакеты, хранимые процедуры, функции, триггеры;
- библиотеки внешних процедур.

Объекты схемы могут состоять из других объектов, называемых подобъектами схемы. К ним относятся разбиения таблиц и видов (partitions), ограничения целостности таблиц, пакетные процедуры и функции и другие элементы, хранимые в пакетах.

К объектам, не принадлежащим схеме, но хранимым в базе данных, относятся:

- роли;
- каталоги;
- профили;

- пользователи;
- сегменты отката;
- табличные пространства.

Таблицы.

Таблицы содержат данные и состоят из столбцов и строк. Для ссылки на строку таблицы можно использовать псевдостолбец ROWID, формируемый Oracle, который содержит уникальный адрес каждой строки. Определить значения ROWID для строк таблицы можно выполнением инструкции SELECT. Использование ROWID для поиска или изменения строки является самым быстрым способом доступа к строке.

Индексы.

Индексы создаются на базе таблиц для ускорения поиска и чтения данных. Если строки таблицы не упорядочены, то значения индекса, созданного по одному или нескольким столбцам, упорядочены. Это позволяет быстрее производить поиск данных по индексу, избегая полного перебора всех строк. После нахождения нужного значения в индексе, искомая строка определяется по значению ROWID строки, хранимому в индексе. Индексы также служат для обеспечения целостности данных и позволяют гарантировать их уникальность.

Кластеры.

Кластер - это объект схемы, состоящий из одной или нескольких таблиц, которые имеют один или несколько общих столбцов. В кластере строки из одной или нескольких таблиц, которые имеют одинаковые значения в их общих столбцах, физически хранятся вместе в базе данных в пределах одного блока данных. Кластеры уменьшают время доступа к таблице и пространство, необходимое для хранения таблицы. Кластеры являются "внутренним делом" самой СУБД и с точки зрения инструкций SQL нет никакой разницы входит таблица в кластер или нет.

Представления (виды).

Представления (виды, view) - это логическая таблица, получаемая из других таблиц или представлений. Иногда говорят, что вид - это сохраняемый запрос, результатом которого является таблица. Вид позволяет:

- ограничивать доступ к таблице выборкой только ее отдельных столбцов/строк;
- объединять несколько таблиц в одну структуру;

Снимки.

Снимок (snapshot) - это таблица, содержащая результат запроса из одной или нескольких таблиц или представлений, как правило с удаленной базы данных. Снимки позволяют получать копию удаленных данных только для чтения, но с возможностью ее автоматического обновления. Снимки позволяют:

· использовать копии таблиц и представлений с удаленных серверов баз данных, доступ к которым является ненадежным или может быть прерван;

· получить более быстрый доступ к данным, расположенным в снимке, чем к их оригиналу на удаленном сервере.

Применение снимков имеет ряд ограничений:

· невозможность изменения удаленных данных, используемых в снимке, на локальном компьютере, так как снимок - это запрос с режимом доступа только для чтения;

· для автоматического обновления снимка требуется соединение с удаленным сервером;

· снимки не могут содержать столбцов типа LONG (большие объекты).

· в снимке нельзя использовать таблицы или представления, принадлежащие пользователю SYS. Процесс синхронизации в распределенной базе данных снимков и таблиц (представлений), на основе которых они созданы, называется репликацией.

Последовательности.

Последовательность (sequence) - это объект базы данных, аналогичный объекту "Счетчик"

СУБД MS Access. Каждое следующее или текущее значение последовательности, с учетом заданного шага приращения, получается, используя псевдостолбцы NEXTVAL или CURVAL. Как правило, последовательности создаются для формирования значений столбца, который используется как первичный ключ.

Синонимы.

Синоним (synonym) определяет дополнительное имя, используемое для доступа к объекту. Синонимы могут быть общедоступные или индивидуальные, которые может применять только пользователь, владеющий объектом. Все общедоступные синонимы принадлежат схеме PUBLIC, объекты которой доступны всем зарегистрированным пользователям базы данных Oracle.

Пакеты, хранимые процедуры, функции, триггеры.

Триггер - это блок инструкций PL/SQL, связанный с определенной таблицей и автоматически запускаемый Oracle при попытке выполнить один из следующих SQL-операторов: DELETE, INSERT или UPDATE. Хранимые процедуры/функции - это программы на PL/SQL, хранимые в самой базе данных и выполняющие определенные действия над ней. Хранимые процедуры могут использоваться наравне с инструкциями SQL, при работе с данной конкретной базой данных, а хранимые функции могут также использоваться в самих инструкциях SQL, в условиях отбора. Возможен вызов одной хранимой процедуры/функции из другой, или вызов хранимой процедуры из триггера.

Пакет - это инкапсулированный набор хранимых процедур, функций, переменных, констант, курсоров (отбор данных), исключений (обработка ошибок), хранимых в базе данных вместе. Использование пакетов имеет ряд преимуществ:

· более удобное управление полномочиями;

· возможность изменения объектов пакета без изменения зависимых объектов схемы;

· использование глобальных переменных и курсоров всеми процедурами и функциями пакета;

· возможность перегрузки (overloading) процедур и функций, имеющих одинаковые имена, но разные списки аргументов (по количеству и/или по типу).

Словарь данных.

В словаре данных (Data Dictionary) хранится вся служебная информация об объектах базы данных Oracle. Словарь данных - это представления, созданные на основании системных таблиц базы данных. Название этих представлений и их столбцов четко определены, поэтому пользователь, знающий эти названия и обладающий соответствующими привилегиями доступа, может всегда получить системную информацию об объектах базы данных.

Распределенные транзакции.

Сервер Oracle позволяет выполнять распределенные транзакции, или транзакции, которые изменяют данные в нескольких базах данных. Для завершения распределенной транзакции достаточно выполнить оператор COMMIT, как и для обычной транзакции. После этого каждый компонент распределенной транзакции завершается на каждой включенной в транзакцию базе данных. При возникновении сбоя в сети или на одной из машин во время процесса завершения транзакции состояние такой транзакции становится неопределенно. Говорят, что распределенная транзакция находится в сомнительном состоянии (in-doubi). После консультаций с администраторами других баз данных, вовлеченных в транзакцию, можно выполнить принудительное завершение или откат транзакции на отдельной локальной базе данных. Для принудительного отката транзакции используется оператор COMMIT с фразой FORCE.

3.3 СУБД Progress.

3.3.1 Архитектура СУБД Progress.

СУБД Progress - это SQL-совместимая реляционная система управления базами данных, поддерживающая многосерверную и многопоточную организацию доступа к данным. СУБД имеет шлюзовую архитектуру, показанную на рис. 21.

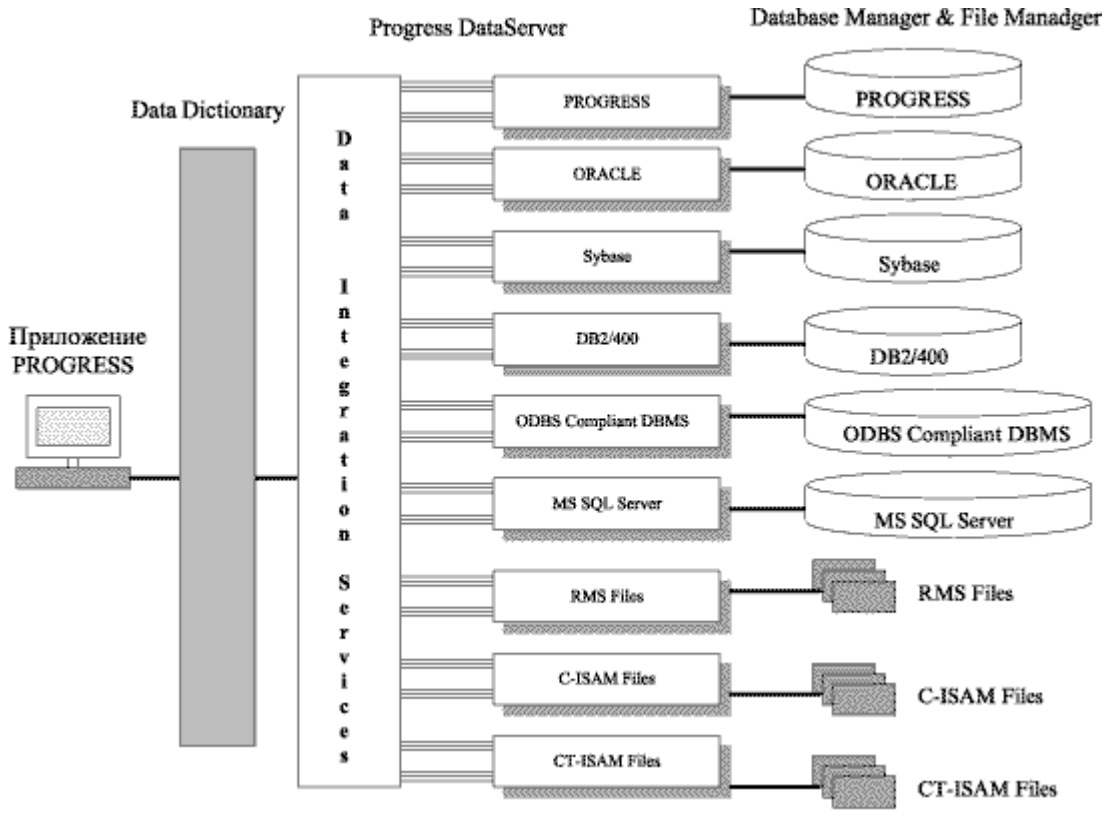


Рис. 21 Архитектура СУБД Progress

Приложения Progress могут работать с данными из базы данных Progress, любой базы данных, для которой есть Progress DataServer или их комбинации, причем тип источника данных определяется только на этапе инсталляции и не требует изменений в приложении. Для доступа к своей базе данных Progress предоставляет ODBC Driver.

DataServer - шлюз нижнего уровня для доступа к другим источникам данных, в том числе и к распределенным.

При работе приложения через DataServer можно использовать специфические для конкретного сервера СУБД возможности, синтаксис и команды (например, использовать преимущества хранимых процедур Oracle, синтаксис PL-SQL, интерфейс SQL*Net) и/или расширять функциональность, используя специфические средства Progress, например триггеры.

Собственный сервер баз данных Progress.

На рис. 22 показана структура собственного сервера баз данных Progress в общем случае.

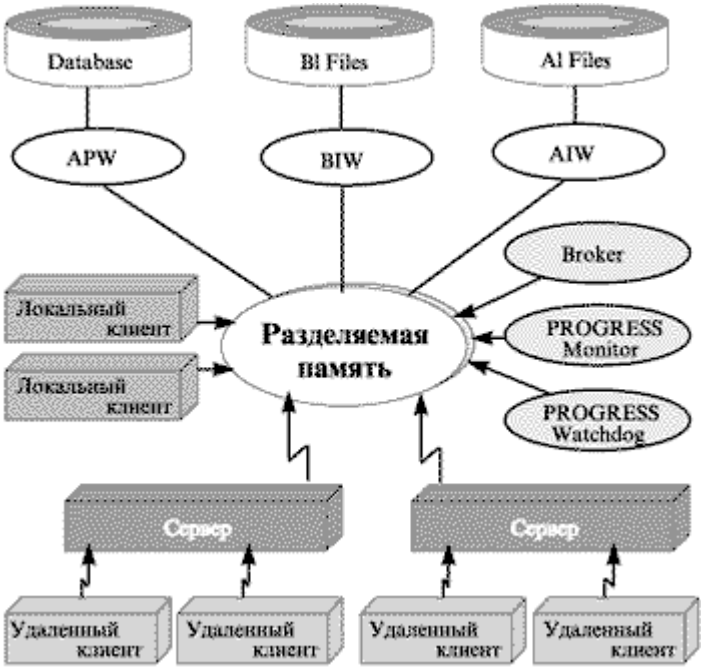


Рис. 22 Сервер СУБД Progress.

3.3.2 Основные компоненты СУБД Progress.

Брокер - процесс, управляющий многопользовательской базой данных. Он управляет разделяемыми ресурсами и выполняет восстановительные работы в базе данных после перезагрузки. Брокер также отвечает за связь с удаленными клиентами, прикрепляя их к существующим серверам удаленных клиентов или, при необходимости, запуская новые серверы. Брокер останавливает процессы удаленных клиентов при завершении работы базы.

Сервер - процесс, который обращается к базе данных через разделяемую память от имени одного или нескольких удаленных клиентов. Сервер выполняется на хост-машине и взаимодействует с удаленными клиентами, используя сетевые протоколы.

Progress Монитор - утилита, показывающая производительность и полезную информацию о статусе базы данных и активности.

Progress Watchdog - утилита, определяющая некорректно завершенные процессы и разблокирующая записи базы данных, структуры в разделяемой памяти и т. д.

При работе с разделяемой памятью Progress использует фоновые процессы, которые постоянно выполняют текущие функции в фоновом режиме. Асинхронная запись буферов позволяет распараллелить обработку данных и запись информации в журналы активных и завершенных транзакций. Сброс буферов осуществляют отдельные процессы, что позволяет значительно увеличить производительность системы, особенно в многопроцессорных системах. Существует три вида фоновых процессов.

- **APWs** (Asynchronous page writers) постоянно записывают измененные блоки базы данных на диск.
- **BIWs** (Before-image writers) постоянно записывают на диск заполненные буферы журнала активных транзакций, служащего для отката базы назад в случае системных сбоев.
- **AIWs** (After-image writers) постоянно записывают на диск заполненные буферы журнала завершенных транзакций. Используя резервную копию базы и журнал завершенных транзакций, всегда можно восстановить текущее состояние базы в случае аппаратных сбоев.

Создание резервных копий баз данных и восстановление баз может производиться как в off-line-, так и on-line-режимах.

В случае распределенных транзакций автоматически используется протокол двухфазной фиксации.

- **Удаленный клиент -** клиентский процесс на удаленном сетевом узле (на компьютере, который не содержит базы данных или брокера базы данных). Удаленные клиенты не обращаются к базе напрямую, а работают через серверы, запускаемые брокером.
- **Самообслуживающийся клиент -** многопользовательская Progress-сессия, запущенная на той же машине, что и брокер. Самообслуживающиеся клиенты работают с базой напрямую через разделяемую память. В случае, когда клиент и сервер функционируют на одном компьютере, Progress позволяет объединить их функции в одном процессе в терминах операционной системы, что сокращает накладные расходы по производительности. Высокая производительность сервера обеспечивается также большим кэшем базы, до 500 000 блоков базы.

Блокировки в Progress поддерживаются вплоть до уровня записей. Существует два типа блокировок: разделяемая (shared) и монопольная (exclusive). Тип и время, на которое блокируется запись, определяются по умолчанию, но могут быть и явно указаны разработчиком. Блокирование доступа возможно на уровне записи, таблицы и базы данных.

3.3.3 Возможности СУБД Progress

Администрирование СУБД.

Сервер баз данных Progress достаточно прост в администрировании, благодаря большому количеству умолчаний и тому, что многие процессы, такие как APWs, автоматически настраивают свое поведение, основываясь на активности использования базы данных, даже если активность принципиально изменяется при переходе с дневной обработки в режиме on-line на пакетную обработку в ночное время. Всего СУБД имеет около 150 конфигурационных опций и параметров, позволяющих оптимально настроить систему для конкретной программно-аппаратной платформы и сетевого взаимодействия.

Некоторые особенности функционирования СУБД Progress.

Каждое приложение работает со словарем базы данных (Data Dictionary), в котором описывается структура базы данных, характеристики отдельных таблиц, полей, индексы, определены триггеры, выполняемые при работе с данными. Все остальные компоненты среды разработки используют по умолчанию информацию, хранимую в словаре, и любые изменения, сделанные в Data Dictionary, наследуются всеми компонентами приложения, в результате чего приложения не зависят ни от физического расположения данных в распределенных и гетерогенных сетях, ни даже от формата источника данных.

Для эффективной работы с документами в Progress существует механизм word indexing - возможность построения индексов по словам в символьном поле.

Приложение может также использовать данные из мета-схемы базы (т. е. само описание базы данных может быть доступно в виде обычной базы) и виртуальные run-time-таблицы (magic tables), содержащие данные об активных блокировках записей, работающих пользователях, параметрах базы данных. Для поддержания семантической целостности базы данных, для различных событий, таких как добавление, удаление, нахождение записи в базе и т. д., можно определить триггеры, выполняемые в ответ на эти события. Механизм триггеров можно использовать для дополнительной журнализации событий в базе, ограничения видимости записей, соответствующих некоторым

критериям для определенных групп пользователей. Например, для события find record для таблицы "Зарплата" можно создать триггер, который позволит просматривать записи о зарплате только тех сотрудников, сумма зарплаты которых не превышает определенного для данного пользователя значения, хранимого в другой таблице. Пользователь может даже не догадываться, что просматривает только часть записей. Все триггеры пишутся в виде обычных 4GL-процедур, могут работать с любым источником данных, поддерживаемым Progress, и до сих пор обладали только тем недостатком, что в отличие от хранимых процедур исполнялись на клиентской стороне. Но с появлением в сентябре 1996 года нового продукта Progress AppServer - компонентного сервера приложений - триггеры могут выполняться в любом узле вычислительной сети, в том числе и на самом сервере базы данных.

Компонентный сервер приложений AppServer.

Одним из важных шагов, предпринятых Progress Software в направлении создания распределенных приложений в N-уровневой архитектуре клиент-сервер, является введение в язык понятия удаленных процедур (remote procedures), разрабатываемых на языке 4GL и используемых для разделения приложения на компоненты. Каждый из этих компонентов запускается на своем сервере приложений в непосредственной близости от данных, с которыми он работает, тем самым выделяя сегмент бизнес-логики из приложения - по сути инкапсулируя его функции в различных компонентах, каждый из которых может быть запущен на разных серверах в различных узлах сети. В приложении, написанном с использованием запускаемых на компонентном сервере удаленных процедур, клиентское приложение может вызвать удаленную процедуру и, не дожидаясь возврата из нее, сразу же перейти к следующей процедуре. Работа клиента будет прервана только по окончании выполнения удаленной процедуры для возвращения результата. Очевидно, что при этом общая производительность приложения значительно возрастает.

Progress/4GL

Компонентный язык 4GL - ядро разработки критически важных приложений, которое позволяет управлять всеми уровнями приложения: пользовательским интерфейсом, бизнес-логикой приложения и данными. Он обеспечивает управление транзакциями, генерацию отчетов, определение механизмов безопасности и целостности данных. В приложении на 4GL для работы с данными можно использовать SQL. Progress/SQL удовлетворяет стандарту ANSI SQL89 Level 2. Также в приложение могут быть интегрированы вставки C, на объекты ActiveX, OLE, механизмы DDL, DDE, Unix named pipes и другие.

Переносимость.

Progress функционирует более чем на 160 программно-аппаратных платформах и поддерживает различные сетевые протоколы - TCP/IP, SPX/IPX, SNA, DecNet, NetBIOS. Приложения, разработанные на одной платформе, могут эксплуатироваться на любой другой из поддерживаемых платформ. Среди них MS-DOS, Novell, Windows, OS/2, Unix (UnixWare, SCO Unix, Interactive, Sun Solaris, AIX, HP-UX, IRIX), Windows NT (Intel, Alpha), OSF/1, OpenVMS, OS/400. Одним из уникальных свойств Progress является то, что одно и то же приложение может работать как в графической среде (Motif, Windows), так и на алфавитно-цифровых терминалах. Разработчик может создавать и различные экранные формы для каждого из типов пользовательского интерфейса.

Интернационализация.

Вопрос интернационализации приложений и поддержки национальных алфавитов является одним из принципов открытости системы. С другой стороны, реализация этого требования довольна сложна, так как требует решения большого количества проблем: сортировки, форматного отображения дат и денежных единиц, календарных дней, телефонных номеров и т. д. Опыт работы с Progress и программными продуктами, разработанными с его помощью, показал, что Progress удовлетворяет самым жестким требованиям, предъявляемым к открытой системе с точки зрения интернационализации.

В каждой стране используется свой алфавит - набор символов (немецкий, французский, испанский, русский). Для их представления обычно используются таблицы кодировок, содержащие коды от 0 до 255. Однако в некоторых случаях одного байта не хватает для представления всех символов алфавита (китайский, корейский, японский). Progress предоставляет возможность работы как с однобайтовыми, так и двухбайтовыми таблицами кодировок.

Системные сообщения.

В ряде случаев, особенно при эксплуатации готовой системы неспециалистами в области информационных технологий, требуется исключить появление на экране любой информации на языке, отличном от используемого в данной стране. Основную проблему вызывают системные сообщения, появляющиеся при различных сбоях; как правило, они выдаются только на английском языке. В Progress можно выбрать язык, на котором будут выдаваться системные сообщения, среди них есть, естественно, и русский.

Одновременная поддержка нескольких языков.

Все более актуальной становится возможность использования одного и того же приложения в различных странах на разных языках с учетом национальных традиций и особенностей. В России разработчикам также приходится решать такого рода задачи. Российские приложения хотят эксплуатировать на Украине, в Казахстане, в других странах бывшего Советского Союза. При этом встает вопрос поддержки национальных алфавитов каждой из этих стран. Progress Software разработала специальный продукт Translation Manager 2 для решения этих вопросов. В его состав входят два основных компонента - Translation Manager и Visual Translator.

Описание процесса перевода приложения.

В процессе перевода выделяются две ключевые фигуры: руководитель проекта и собственно переводчик. Каждый из них решает свои задачи. Руководитель проекта формирует задание для переводчика, определяет стоимость работы, распределяет ее между переводчиками, осуществляет контроль за выполнением работы, готовит окончательный вариант приложения. Переводчик получает задание, осуществляет перевод и возвращает выполненную работу руководителю проекта. В соответствии с этими ролями распределены функции между двумя компонентами.

Руководитель проекта определяет те процедуры, которые должны быть переведены на другой язык. После этого он указывает фильтр для типов сообщений, которые необходимо перевести в каждой процедуре. Такими типами могут быть: меню, метки, наименования кнопок и т. д. После этого происходит формирование задания для переводчиков. В задание могут быть включены

стандартные словари, содержащие перевод основных терминов, и допускается использование стандартных словарей фирмы Microsoft, которые применялись для перевода Word и Excel. При этом выдается вся статистика, позволяющая оценить трудоемкость работы и возможные сроки ее окончания. Переводчику передается только информация, необходимая для перевода, без логики самого приложения. Задание может посылаться по почте, на дискете или иным способом.

Получив задание, переводчик приступает к его переводу на другой язык. Возможны два режима перевода: поэкранный либо построчный. После окончания перевода и его проверки переводчик посылает выполненное задание обратно руководителю проекта.

Руководитель проекта, собрав полученные работы от переводчиков, консолидирует их работу и выполняет компиляцию приложений. После этого с данным приложением можно работать с использованием нового языка. Язык задается стартовым параметром.

Разработчиком Progress является американская компания Progress Software Corporation (Бедфорд), имеющая представительства в 64 странах мира. На Progress создано более 3000 приложений, среди которых крупные банковские системы, системы управления промышленными, торговыми предприятиями. На территории СНГ дистрибьютором Progress Software является компания CSBI EE.

Для создания отчетов в интерактивном режиме существует специальная компонента Progress - Report Builder (до версии 8.1 включительно, в последующих версиях - Actuate). Заметим, что Report Builder требует отдельного connect с базой данных, поэтому в однопользовательском сеансе перед входом в среду Report Builder следует отсоединиться от базы данных.

Список литературы.

1. *Бобровски С.* Oracle8. Архитектура. — М.: ЛОРИ, 1999.
2. Вейскас Д. Эффективная работа с Microsoft Access 97 – СПб: ЗАО «Издательство «Питер», 1999. – 976 с.
3. М. Ф. Гарсиа, Дж. Рединг, Э. Уолен, С. А. ДеЛюк «Microsoft SQL Server 2000. Справочник администратора». ЭКОМ, 2002
4. Д. Дэвитт, Д. Грей. Параллельные системы баз данных: будущее высокоэффективных баз данных. СУБД, вып. 2, 1995
5. А.Г. Ляхевич Обзор баз данных. Учебное пособие. Минск, 2002
6. Мамаев Е., Шкарина Л. «Microsoft SQl Server 2000 для профессионалов». СПб:Питер, 2001
7. Ю.Пуха Объектные технологии построения распределенных информационных систем. СУБД вып.3, 1997.
8. Пейдж В., Хьюз Н., Остин Д. Использование Oracle8. — К.; М.; СПб.: Вильямс, 1998.
9. Хоторн Роб «Разработка баз данных, Micrososoft SQL Server 2000». Вильямс, 2001
10. Шарон Б., Мэйбл Грэг «Sql Server 2000, Энциклопедия программиста» ДиаСофт, 2001.