Базы данных. Общие сведения

ОСНОВНЫЕ ПОНЯТИЯ.

В начале года мы говорили об информационных системах (ИС), предназначенных для обработки больших массивов данных. Основа ИС, объект ее обработки – база данных.

Базы данных представляют собой информационные модели. Содержащие данные об объектах и их свойствах. БД хранят информацию о группах объектов с одинаковым набором свойств.

Например, БД Записная книжка хранит информацию о людях, каждый из которых имеет фамилию, имя, телефон и т.д. Библиотечный каталог хранит информацию о книгах, каждая из которых имеет название, автора, год издания и т.д.

Информация в БД хранится в упорядоченном виде. Так, в записной книжке все записи упорядочены по алфавиту, а в библиотечном каталоге либо по фамилиям авторов (алфавитный каталог) либо по области знания (предметный каталог).

Таким образом,

База данных – это информационная модель, позволяющая упорядочено хранить данные о группе объектов, обладающих одинаковым набором свойств.

1. СУБД

Развитие информационных технологий привело к созданию компьютерных БД. Создание БД, а также операции поиска и сортировки данных выполняются специальными программами – СУБД.

Системы управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Таким образом, следует различать собственно БД, которые являются упорядоченными наборами данных и СУБД – программы, управляющие хранением м обработкой данных.

Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Программы, с помощью которых пользователи работают с базой данных, называются *приложениями.* В общем случае с одной базой данных могут работать множество различных приложений. Например, если база данных моделирует некоторое предприятие, то для работы с ней может быть создано приложение, которое обслуживает подсистему учета кадров, другое приложение может быть посвящено работе подсистемы расчета заработной платы сотрудников, третье приложение работает как подсистемы складского учета, четвертое приложение посвящено планированию производственного процесса. При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно и независимо друг от друга, и именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями.

Приложение представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи. В данном случае имеются в виду приложения, использующие БД.

Приложения разрабатывают главным образом в тех случаях, когда требуется обеспечить удобство работы с БД неквалифицированным пользователям или интерфейс СУБД не устраивает пользователей.

Приложения могут создаваться в среде или вне среды СУБД — с помощью системы программирования, использующей средства доступа к БД, например, Delphi или C++ Builder. Приложения, разработанные в среде СУБД называют *приложениями СУБД,* а приложения, разработанные вне СУБД, - *внешними приложениями.*

В современном мире системы обработки информации играют огромную роль, поскольку от них во многом зависит эффективность работы любого предприятия или учреждения.

1.1. Основные функции СУБД:

• **Ввод информации в БД и обеспечение его логического контроля.** Под логическим контролем здесь понимается проверка на допустимость вводимых данных: нельзя, например, вводить дату рождения 31 июня 1057 года.

• **Исправление информации** (также с контролем правильности ввода).

• **Удаление устаревшей информации.**

• **Контроль целостности и непротиворечивости данных.** Здесь имеется в виду, что данные, хранящиеся в разных частях базы данных, не противоречат друг другу, например, **дата поступления в** школу **явно** не может быть позже даты ее окончания.

• **Защита данных от разрушения.** СУБД должна иметь средства защиты данных от выключения электропитания, сбоев оборудования и других аварийных ситуаций, а также возможности последующего восстановления информации.

• **Поиск информации с необходимыми свойствами.** Одна из наиболее важных в практическом отношении задач, ради которой ставятся все остальные.

• **Автоматическое упорядочивание информации в соответствии с требованиями человека.** Сюда относится сортировка данных, распределение их между несколькими базами и другие подобные процедуры.

• **Получение общих и/или детализированных отчетов по итогам работы.**

• **Обеспечение коллективного доступа к данным.** В современных информационных системах возможен параллельный доступ к одним и тем же данным нескольких пользователей, поэтому СУБД должны поддерживать такой режим.

• **Защита от несанкционированного доступа.** Не только ввод новой информации, но даже ее просмотр должны быть разрешены только тем пользователям, у которых есть на это права.

• **Удобный и интуитивно понятный пользователю интерфейс.**

Говоря о БД, нельзя обойти стороной вопрос, связанный с организацией в них данных. Помимо собственно данных, в любой базе имеется информация о ее строении, которую чаще всего называют ***структурой*.** В простейшем случае структура просто указывает тип информации и объем требуемой для нее памяти. Сведения о структуре позволяют СУБД легко рассчитывать местоположение требуемых данных на внешнем носителе и, следовательно, быстро получить к ним доступ.

Связанные между собой данные, например об одном человеке или объекте, объединяются в БД в единую конструкцию, которая называется з***апись***. При этом части, образующие запись, принято называть ***полями*** или реже — элементами данных. Примерами полей могут служить фамилия, номер паспорта, семейное положение, наличие или отсутствие детей и т.д.

С появлением компьютерных сетей отпала необходимость хранения данных в одной машине и даже в одной стране, возникли так называемые ***распределенные БД*.**

1.2. Классификация СУБД

В качестве основных классификационных признаков используют: вид программы, характер использования, модель данных.

К СУБД относятся следующие **основные виды программ**:

1. Полнофункциональные СУБД.

Полнофункциональные СУБД представляют собой традиционные СУБД, которые являются наиболее многочисленными и мощными по своим возможностям. Например, dBase IV, Microsoft Access, Microsoft FoxPro и Paradox R:BASE.

Обычно они имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД: создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчеты, выводить их на печать и т. п. ((Для создания запросов и отчетов не обязательно программирование. Включают средства программирования для профессиональных разработчиков.))

2. Серверы БД.

Серверы БД предназначены для организации центров обработки данных в сетях ЭВМ. Эта группа БД в настоящее время менее многочисленна, но им количество постепенно растет. Серверы БД реали-

зуют функции управлении битами данных, запрашиваемые другими (клиентскими) программами обычно с помощью операторов SQL.

Примерами серверов БД являются следующие программы: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland).

3. Клиенты БД.

В роли ***клиентских программ*** для серверов БД в общем случае могут использоваться различные программы: СУБД 1-гго типа, электронные таблицы, текстовые процессоры, программы электронной почты и т. д.

Сервер – компьютер (программа) управляющая определенным ресурсом (например, БД) в компьютерной сети. ***Клиент*** – компьютер (программа), использующий этот ресурс.

((В случае, когда клиентская и серверная части выполнены одной фирмой, естественно ожидать, что распределение функций между ними выполнено рационально. В остальных случаях обычно преследуется цель обеспечения доступа к данным «любой ценой»)).

4. Средства разработки программ работы с БД.

К средствам разработки пользовательских приложений относятся системы программирования, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок (в том числе систем типа клиент-сервер). В числе наиболее распространенных можно назвать следующие инструментальные системы: Delphi (Borland), Visual Basic (Microsoft) и др..

По характеру использования СУБД делят на:

1. ***Персональные СУБД*** обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД и разработанные с их помощью приложения могут выступать в роли клиентской части многопользовательской СУБД. К персональным СУБД, например, относятся Visual FoxPro, Paradox, Clipper, dBase, Access и др.

2. ***Многопользовательские СУБД*** включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle и Informix.

Логическую структуру хранимых в базе данных называют ***моделью представления данных***. К основным моделям представления данных относятся следующие: иерархическая, сетевая, реляционная, постреляционная, многомерная и объектно-ориентированная.

Хранимые в базе данные имеют определенную логическую структуру — иными словами, описываются некоторой ***моделью представления данных*** (моделью данных), поддерживаемой СУБД.

По используемой **модели данных СУБД (как и БД), разделяют на**:

1. Иерархическая,
2. Сетевая,
3. Реляционная.
4. Постреляционная,
5. Многомерная,
6. Объектно-Ориентированная.

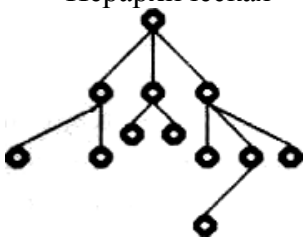
(1-3) классические, (4-6) появились в последнее время.

Некоторые СУБД могут одновременно поддерживать несколько моделей данных.

2. Модели БД и СУБД

2.1. Иерархическая БД

Иерархическая



Иерархические БД графически могут быть представлены как перевернутое дерево, состоящее из объектов различных уровней. ((Верхний уровень занимает один объект, второй - объекты второго уровня и т.д.))

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении *предка (родителя* или более правильно *владельца отношения)* к *потомку*, при этом объект-предок может не иметь потомков или иметь их несколько, тогда как объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются *близнецами*.

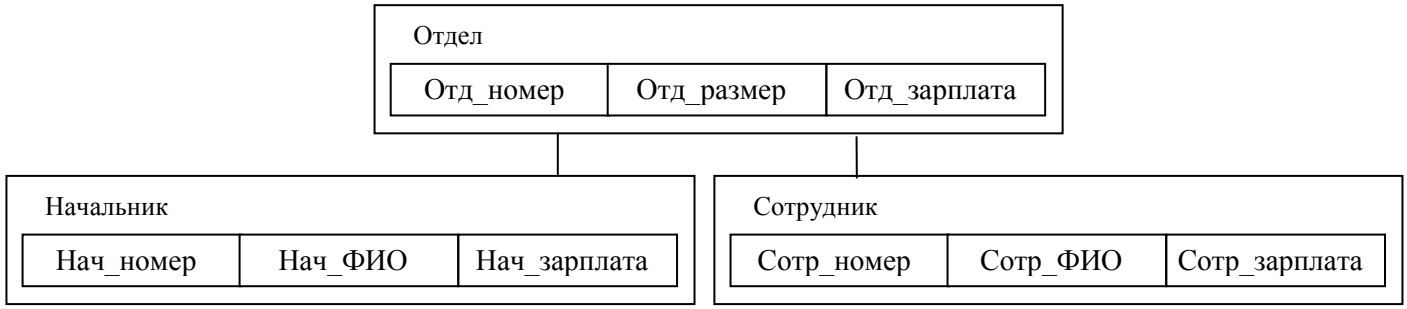
Объект, который имеет потомков, но не имеет предков называется *корневым*.

Иерархическая БД представляет собой упорядоченную совокупность экземпляров данных типа «дерево», содержащих экземпляры типа «запись». Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание БД. Обход всех элементов иерархической БД обычно производится сверху вниз и слева направо.

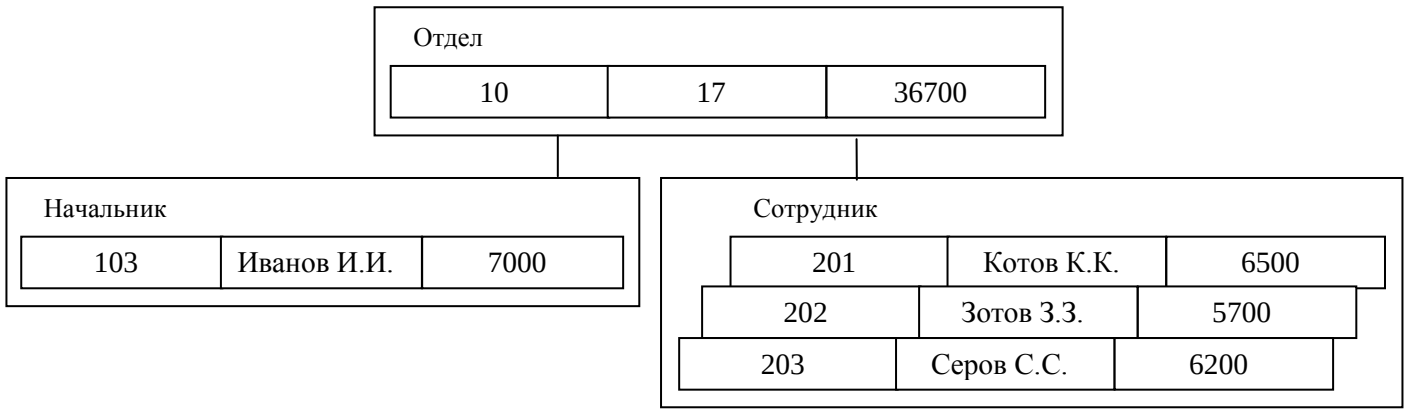
Типичными примерами иерархического способа организации является хорошо известная система вложенных каталогов в операционной системе, или так называемое "генеалогическое дерево", представляющее собой графическое представление родословной.

Например,

Структура:



Данные:



(Например, Иерархической БД является Каталог папок Windows. Верхний уровень занимает Рабочий стол. На втором уровне находятся папки Мой компьютер, Мои документы, Сетевое окружение, Корзина, которые являются потомками папки Рабочий стол, а между собой – близнецами. В свою очередь папка Мой компьютер является предком по отношению к папкам 3-го уровня, папкам дисков (Диск 3,5(A:), Локальный диск (C:), Компакт-диск (D:)) и системным папкам (Принтеры, Панель управления и т.п.)).

((Основные операции манипулирования иерархически организованными данными:
- поиск указанного экземпляра БД (например, дерева со значением 10 в поле Отд_номер);
- переход от одного дерева к другому;
- переход от одной записи к другой внутри дерева (например, к следующей записи типа Сотрудники);
- вставка новой записи в указанную позицию;
- удаление текущей записи и т. д.))

В соответствии с определением типа «дерево», можно заключить, что между предками и потомками автоматически поддерживается контроль целостности связей. Основное правило контроля целостности формулируется следующим образом: потомок не может существовать без родителя, а у некоторых родителей может не быть потомков. Механизмы поддержания целостности связей между записями различных деревьев отсутствуют.

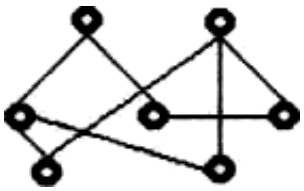
Достоинства:
- эффективное использование памяти ЭВМ;
- неплохие показатели времени выполнения основных операций над данными.

Недостатки:
- громоздкость для обработки информации с достаточно сложными логическими связями;
- сложность понимания ((для обычного пользователя)).

Примеры СУБД: PC/Focus, Team-Up, Data Edge, а также отечественные Ока, ИНЭС и МИРИС.

2.2. Сетевая БД

Сетевая



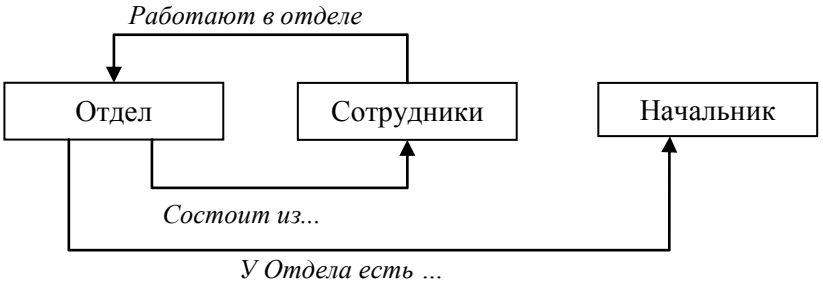
Сетевая БД является обобщением иерархической за счет допущения объектов, имеющих более одного предка, т.е. каждый элемент вышестоящего уровня может быть связан одновременно с любыми элементами следующего уровня. Вообще, в **сетевой** базе данных связи разрешено устанавливать произвольным образом, без всяких ограничений, поэтому запись может быть найдена значительно быстрее **(по** наиболее короткому пути).

Такая модель лучше всего соответствует реальной жизни: один и тот же человек является одновременно и работником, и клиентом банка, и покупателем, т.е. запись с информацией о нем образует довольно густую сеть сложных связей.

Сетевая БД состоит из набора записей и набора соответствующих связей.

Например,

Схема простейшей сетевой БД (типы связей обозначены надписями на соединяющих типы записей линиях).



Сетевой БД фактически является Всемирная паутина глобальной компьютерной сети Интернет. Гиперссылки связывают между собой сотни миллионов документов в единую сетевую БД.

((Операции манипулирования данными баз сетевого типа:
- поиск записи в БД;
- переход от предка к первому потомку;
- переход от потомки к предку;

- создание новой записи;
- удаление и обновление текущей записи;
- включение и исключение записи в связи;
- изменение связей и т.д.))

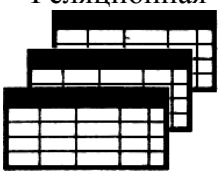
Достоинства:
- возможность эффективной реализации по показателям затрат памяти и оперативности;
- допускает большие возможности в образовании связей.

Недостатки:
- высокая сложность и жесткость схемы БД, построенной на ее основе;
- сложность для понимания и выполнения обработки информации в БД обычным пользователем.
- указанную организацию БД сложно реализовать на компьютере.
- ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

Примеры СУБД: IDMS, db_VistaIII, а также отечественные СЕТЬ, СЕТОР и КОМПАС.

2.3. Реляционная модель

Реляционная



Хотя описанные выше способы являются более универсальными, на практике распространен самый простой тип организации данных — ***реляционный.*** Слово *реляционный* происходит от английского *relation* (*отношение*). Строгое определение отношения достаточно математизировано, поэтому на практике обычно пользуются следствием из него: поскольку отношения удобно представлять в виде таблиц, то говорят, что реляционные базы — это базы с табличной формой организации.

Таблица имеет строки (записи) и столбцы (колонки). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам — атрибуты отношения.

С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно деление одного объекта (явления, сущности, системы и проч.), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей.

Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют *связывание* таблиц.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Например,

Таблица 1. Накладные

Ном_Накладной	Ном_Покупки
044	1234
055	9119
077	1234

Таблица 2. Накладные_товары

Ном_Накладной	Товар	Количество
044	Сыр	3
044	Рыба	2
055	Мясо	1
055	Сок	6
055	Торт	2
077	Масло	1

Достоинство табличной БД заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно это явилось основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Примеры СУБД: dBase, FoxPro, FoxBase, Paradox, Access, Oracle, а также отечественные ПАЛЬМА и HyTech.

2.4. Постреляционная модель

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц. Постреляционная модель данных допускает многозначные поля — поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Например, данные приведенной выше реляционной БД в постреляционной модели:

Таблица 2. Накладные_товары

Ном_Накладной	Ном_Покупки	Товар	Количество
044	1234	Сыр	3
		Рыба	2
055	9119	Мясо	1
		Сок	6
		Торт	2
077	1234	Масло	1

Достоинства:

- возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей, что позволяет ((по сравнению с реляционной моделью)) хранить данные более эффективно, а при обработке не требуется выполнять операцию соединения данных из двух таблиц.
- высокая наглядность представления информации и эффективность ее обработки.

Недостаток

- сложность решения проблемы целостности и непротиворечивости хранимых данных.

Примеры СУБД: uniVers, Budda, Dasdb.

2.5. Многомерная модель

Многомерный подход к представлению данных в базе появился практически одновременно с реляционным, но реально работающих многомерных СУБД до настоящего времени было очень мало. С середины 90-х годов интерес к ним стал приобретать массовый характер.

((Толчком послужила в 1993 году программная статья одного из основоположников реляционного подхода Э. Кодда. В ней сформулированы 12 основных требований к системам класса OLAP (OnLine Analytical Processing — оперативная аналитическая обработка), важнейшие из которых связаны с возможностями концептуального представления и обработки многомерных данных. Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

В развитии концепций ИС можно выделить следующие два направления:

• системы оперативной (транзакционной) обработки;

• системы аналитической обработки (системы поддержки принятия решений).))

Реляционные СУБД предназначались для информационных систем *оперативной* обработки информации и в этой области были весьма эффективны. В системах аналитической обработки (системы поддержки принятия решений) более эффективными здесь оказываются многомерные СУБД.

***Многомерные** СУБД* являются узкоспециализированными СУБД, предназначенными для интерактивной аналитической обработки информации.

((Основные понятия, используемые в этих СУБД:

- *Агрегируемость* данных означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь-оператор, управляющий, руководитель.
- *Историчность* данных предполагает обеспечение высокого уровня статичности (неизменности) собственно

данных и их взаимосвязей, а также обязательность привязки данных ко времени. Статичность данных позволяет использовать при их обработке специализированные методы загрузки, хранения, индексации и выборки. Так, для уменьшения времени обработки запросов желательно, чтобы данные всегда были отсортированы в том порядке, в котором они наиболее часто запрашиваются.

- *Прогнозируемость* данных подразумевает задание функций прогнозирования и применение их к различным временным интервалам.))

По сравнению с реляционной моделью многомерная организация данных обладает более высокой наглядностью и информативностью.

Например, данные об объемах продаж автомобилей представим в виде реляционной и многомерной модели.

Реляционная		
Модель	Месяц	Объем
Жигули	Январь	12
Жигули	Февраль	24
Жигули	Март	10
Ока	Январь	5
Ока	Февраль	2
Волга	Январь	19
Волга	Март	15

Многомерная			
Модель	Январь	Февраль	Март
Жигули	12	24	10
Ока	5	2	0
Волга	19	0	15

Пример 1.

Многомерные БД могут иметь размерность больше двух. Например (см.рис.).

Термины:

Измерение – это подмножество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются Дни, Месяцы, Кварталы и Годы. В качестве географических измерений широко употребляются Города, Районы, Регионы и Страны. В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка или ***показатель*** — это поле, значение которого однозначно определяется фиксированным набором измерений.

В примере 1 каждое значение ячейки *Объем продаж* однозначно определяется комбинацией временного измерения (Месяц продаж) и модели автомобиля.

Срез - представляет собой подмножество гиперкуба, полученное в результате фиксации одного или нескольких измерений.

Например, если ограничить значения измерения Модель автомобиля в гиперкубе маркой «Жигули», то получится двухмерная таблица продаж этой марки автомобиля различными менеджерами по годам.

Пример 2.

Трехмерная модель данных:

Измерения:

Время (год) -2006, 2007, 2008

Менеджер – Котов, Орлов, Попов

Модель – Волга, Ока, Жигули

Показатель: Объем продаж

Достоинство: удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. ((При организации обработки аналогичных данных на основе реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размерности БД и существенное увеличение затрат оперативной памяти на индексацию)).

Недостаток: громоздкость для простейших задач обычной оперативной обработки информации.

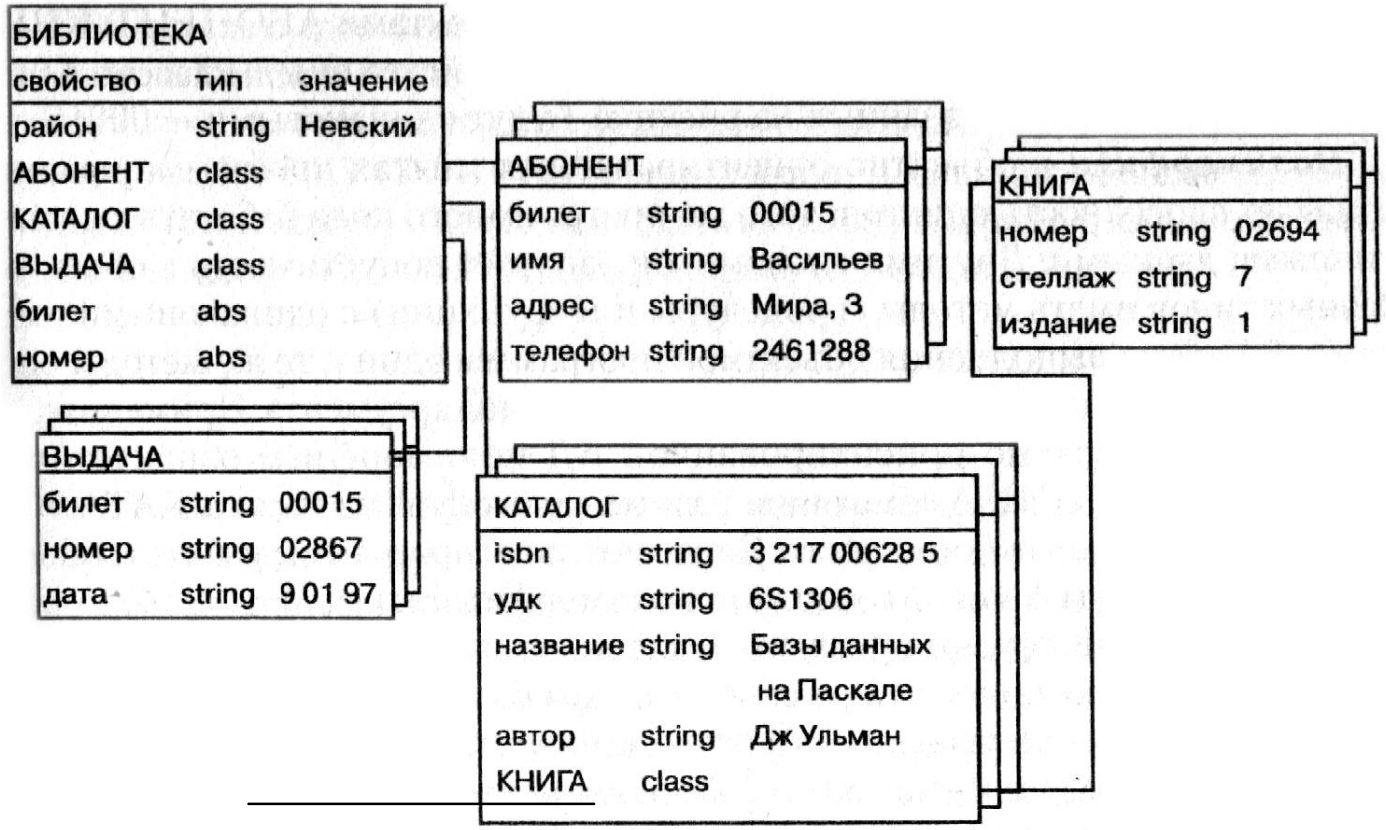
СУБД: Oracle Express Server, Essbase, Media Multi-matrix, Cache.

2.6. Объектно-ориентированная модель

((В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями БД и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.))

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, строковым — siring) или типом, конструируемым пользователем (определяется как class). Значением свойства типа string является строка символов. Значением свойства типа class есть объект, являющийся экземпляром соответствующего класса. Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.

Например, Логическая структура объектно-ориентированной БД библиотечного дела:



Здесь объект типа БИБЛИОТЕКА является родительским для объектов-экземпляров классов АБОНЕНТ, КАТАЛОГ и ВЫДАЧА. Различные объекты типа КНИГА могут иметь одного или разных родителей. Объекты типа КНИГА, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств *isbn, удк, название* и *автор.*

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными.

((Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, усиленные объектно-ориентированным механизмами инкапсуляции, наследования и полиморфизма)).

((Термины.

Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено.

Так, если в объект типа КАТАЛОГ добавить свойство, задающее телефон автора книги и имеющее название *Телефон,* то это будет совершенно другое свойство чем одноименные свойство объектов АБОНЕНТ.

Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта.

Так, всем объектам типа КНИГА, являющимся потомками объекта типа КАТАЛОГ, можно приписать свойства объекта-родителя *isbn, удк, название* и *автор.*

Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками, то в их общем предке определяется абстрактное свойство типа abs.

Так, определение абстрактных свойств *билет* и *номер* в объекте БИБЛИОТЕКА приводит к наследованию этих свойств всеми объектами АБОНЕНТ, КНИГА и ВЫДАЧА.

Полиморфизм означает способность одного и того же программного кода работать с разнотипными данными.

??? Например, объекты класса КНИГА, имеющие разных родителей из класса КАТАЛОГ, могут иметь разный набор свойств. Следовательно, программы работы с объектами класса КНИГА могут содержать полиморфный код.))

Основное достоинство объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации в сложных взаимосвязях объектов.

Недостатки:

- высокая понятийная сложность,
- неудобство обработки данных,
- низкая скорость выполнения запросов.

СУБД: POET, Jasmine, Versant, O2. Iris, Orion и др.

3. Типы данных.

- *Счетчик* – содержит последовательность целых чисел, которые задаются автоматически при вводе записей. Эти числа не могут быть изменены пользователем.
- *Числовой* – содержит числа (примеры значений данных: 0.43,328, 2E+5).
- *Денежный* – содержит числа в денежном формате.
- *Символьный* (алфавитно-цифровые) – содержит до 255 символов (примеры значений данных: «пятница», «строка», «программист»).
- *Символьные переменной длины*, предназначенные для хранения текстовой информации большой длины, например, документа (в Access – поле MEMO).
- *Дата/Время* – предназначены для хранения информации о времени и/или дате (примеры значений данных: 31.01.85 (дата), 10:03 (время), 6.03.1960 12:00 (дата и время)).
- *Логический* – содержит значения Истина или Ложь.
- *Двоичный* – предназначены для хранения графических объектов, аудио- и видеоинформации, пространственной, хронологической и другой специальной информации (например, в MS Access тип Поле объекта OLE, который позволяет хранить в БД графические данные в формате BMP (Bitmap) и автоматически их отображать при работе с БД).
- *Гиперссылки* – предназначены для хранения ссылок на различные ресурсы (узлы, файлы, документы и т.д.), находящиеся вне базы данных, например, в сети Интернет, корпоративной сети интранет или на жестком диске компьютера (примеры значений данных: http:\\www.chat.ru, ftp:\\chance4u.teens.com.).

4. Реляционные (табличные) БД

Данная модель представления данных является самой распространенной.

Термины и определения

Реляционная модель данных (РМД) некоторой предметной области представляет собой набор отношений, изменяющихся во времени.

Слово *реляционный* происходит от английского relation (отношение).

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные.

Сущность есть объект любой природы, данные о котором хранятся в отношении БД.

Атрибуты представляют собой свойства, характеризующие сущность. В структуре таблицы каждый атрибут имеет имя и ему соответствует заголовок некоторого столбца таблицы.

Кортеж (запись) - строка таблицы

Домен представляет собой множество всех возможных значений определенного атрибута отношения.

Тип данных - тип значений элементов таблицы

Схема отношения (заголовок таблицы) - строка заголовков столбцов (имен атрибутов) таблицы.

Первичный ключ (ключ отношения, ключевой атрибут) - один или несколько атрибутов однозначно идентифицирующий (определяющий) каждый из его кортежей.

Если ключ состоит из нескольких атрибутов, то он называется ***составным*** (***сложным***).

Ключи обычно используют для достижения следующих целей:

- исключения дублирования значений в ключевых атрибутах;
- автоматическая сортировка кортежей;
- ускорения работы (например, поиска) с кортежами отношения;
- организации связывания таблиц.

Внешние ключи служат для связи отношений (таблиц БД) между собой. При этом атрибут одного отношения (родительского) называется *внешним ключом* данного отношения, если он является *первичным* ключом другого отношения (дочернего). Говорят, что отношение, в котором определен внешний ключ, ссылается на отношение, в котором этот же атрибут является первичным ключом.

((Поскольку не всякой таблице можно поставить в соответствие отношение, приведем условия, выполнение которых позволяет таблицу считать отношением.))

Таблицу можно считать отношением если соблюдаются условия:

1. Все строки таблицы должны быть уникальны, то есть не может быть строк с одинаковыми первичными ключами.
2. Имена столбцов таблицы должны быть различны, а значения их простыми, то есть недопустима группа значений в одном столбце одной строки.
3. Все строки одной таблицы должны иметь одну структуру, соответствующую именам и типам столбцов.
4. Порядок размещения строк в таблице может быть произвольным.

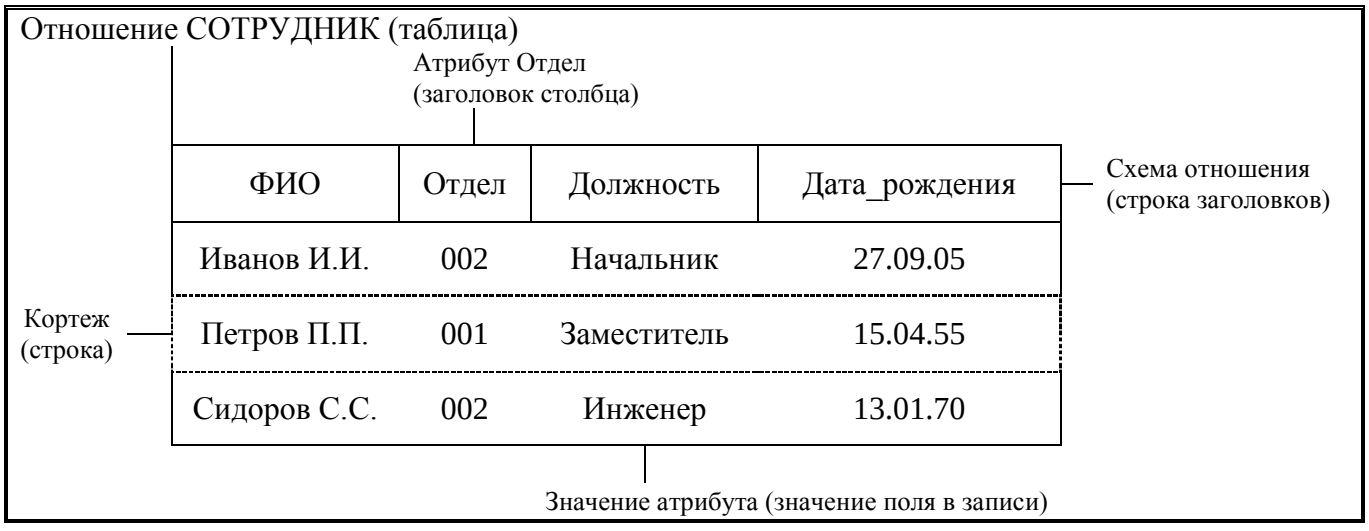
В общем случае можно считать, что БД включает одну или несколько таблиц, объединенных смысловым содержанием, а также процедурами контроля целостности и обработки информации в интересах решения некоторой прикладной задачи.

К отношениям можно применять систему операций, позволяющую получать одни отношения из других. Например, результатом запроса к реляционной БД может быть новое отношение, вычисленное на основе имеющихся отношений. Поэтому можно разделить обрабатываемые данные на хранимую и вычисляемую части.

Основной единицей обработки данных в реляционных БД является таблица, а не отдельные его кортежи (записи).

((Пример представления отношения СОТРУДНИК.

В общем случае порядок кортежей в отношении, как и в любом множестве, не определен. Однако в реляционных СУБД для удобства кортежи все же упорядочивают. Чаще всего для этого выбирают некоторый атрибут, по которому система автоматически сортирует кортежи по возрастанию или убыванию. Если пользователь не назначает атрибута упорядочивания, система автоматически присваивает номер кортежам в порядке ввода.



Отношение СОТРУДНИК включает 4 домена: домен 1 содержит фамилии всех сотрудников, домен 2 — номера всех отделов фирмы, домен 3 – названия должностей, домен 4 - даты рождения. Каждый домен образует значения одного типа данных, например числовые или символьные.

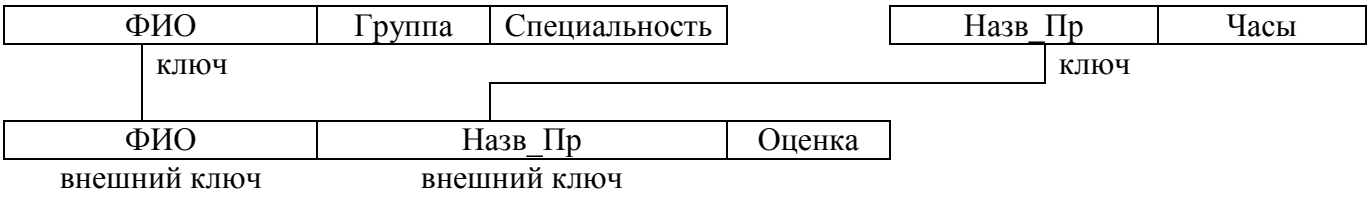
Отношение СОТРУДНИК содержит 3 кортежа. Кортеж рассматриваемого отношения состоит из 4 элементов, каждый из которых выбирается из соответствующего домена. Каждому кортежу соответствует строка таблицы.

Схема отношения имеет вид СОТРУДНИК(ФИО, Отдел, Должность, Дата_рождения).

В отношении СОТРУДНИК ключевым является атрибут «ФИО».))

С помощью внешних ключей устанавливаются связи между отношениями.

Например, имеются два отношения СТУДЕНТ (ФИО, Группа, Специальность) и ПРЕДМЕТ (Назв_Пр, Часы), которые связаны отношением СТУДЕНТ_ПРЕДМЕТ (ФИО, Назв_Пр, Оценка):



В связующем отношении атрибуты ФИО и Назв_Пр образуют составной ключ. Эти атрибуты представляют собой внешние ключи, являющиеся первичными ключами других отношений.

4.2. Индексирование

Термин «индекс» тесно связан с понятием «ключ», хотя между ними есть некоторое отличие.

Под ***индексом*** понимают средство *ускорения* операции поиска записей в таблице, а следовательно, и других операций, использующих поиск: извлечение, модификация, сортировка и т. д. Таблицу, для которой используется индекс, называют ***индексированной*.**

Индекс выполняет роль *оглавления* таблицы, просмотр которого предшествует обращению к записям таблицы.

На практике для создания индекса для некоторой таблицы БД пользователь указывает поле таблицы, которое требует индексации. Ключевые поля таблицы во многих СУБД как правило индексируются автоматически, такие индексы называют *первичными* **.**

Индексы, создаваемые пользователем для не ключевых полей, называют вторичными (пользовательскими) индексами. Введение таких индексов не изменяет физического расположения записей таблицы, но влияет на последовательность просмотра записей.

((Некоторыми СУБД, например Access, деление индексов на первичные и вторичные не производится. В этом случае используются автоматически создаваемые индексы и индексы, определяемые пользователем по любому из неключевых полей.))

Главная причина повышения скорости выполнения различных операций в индексированных таблицах состоит в том, что основная часть работы производится с небольшими индексными файлами, а не с самими таблицами. Чем больше объем таблицы, тем заметней эффект повышения производительности работы.

4.3. Связывание таблиц

При проектировании БД информацию обычно размещают в нескольких таблицах. В реляционных СУБД для указания связей таблиц производят операцию *связывания***.**

Преимущества связанных таблиц:

1. автоматически выполняется контроль целостности вводимых в БД данных в соответствии с установленными связями, что повышает достоверность хранимой в БД информации;
2. доступ к данным становится проще, так как появляется возможность обращения к произвольным полям связанных записей.

Основные виды связи таблиц

Между таблицами могут устанавливаться бинарные (между двумя таблицами), тернарные (между тремя таблицами) и, в общем случае, n-арные связи. Рассмотрим наиболее часто встречающиеся бинарные связи.

При установке бинарных связей (между двумя таблицами) выделяют основную и дополнительную

(подчиненную) таблицы. Логическое связывание таблиц производится с помощью ***ключа связи***.

Ключ связи, по аналогии с обычным ключом таблицы, состоит из одного или нескольких полей, которые называют ***полями связи*** (ПС).

Суть связывания состоит в установлении соответствия полей связи основной и дополнительной таблиц. Поля связи основной таблицы могут быть обычными и ключевыми. Поля связи подчиненной таблицы чаще всего ключевые.

В зависимости от того, как определены поля связи основной и дополнительной таблиц выделяют четыре основных вида связи:

Характеристики видов связей таблиц

Вид связи	Поля связи	
	основной таблицы	дополнительной таблицы
1:1 (один-один)	Являются ключом	ключ
1:M (один-много)	ключ	Не ключ
M:1 (много-один)	Не ключ	ключ
M:M или M:N(много-много)	Не ключ	Не ключ

Дадим характеристику названным видам связи между двумя таблицами приведем примеры их использования.

Связь вида 1:1

Связь вида 1:1 образуется в случае, когда все поля связи основной и дополнительной таблиц являются ключевыми. Поскольку значения в ключевых полях обеих таблиц не повторяются, обеспечивается взаимнооднозначное соответствие записей из этих таблиц. Сами таблицы, по сути, становятся равноправными.

Пример 1.

В некоторых ВУЗах при сдаче письменных вступительных экзаменов работы для обеспечении большей объективности не подписывают, а дают шифр (код). В данной ситуации необходимо две таблицы, одна с секретной информацией (код_абитуриента и данные о нем), а другая для служебного пользования (код_работы и оценка). Связать таблицы можно по коду. Первую таблицу целесообразно защитить от несанкционированного доступа.

Ключевое поле, поле связи — ТАБЛ. АБИТУРИЕНТ

Код_абитуриента	ФИО
101	Алов А.А.
102	Белов Б.Б.
103	Серов С.С.

Ключевое поле, поле связи — ТАБЛ. ЭКЗАМЕН

Код_работы	Оценка
101	5
103	3

В приведенных таблицах установлена связь между записью (101, Алов А.А.) таблицы Абитуриент и записью (101, 5) таблицы Экзамен. Основанием этого является совпадение значений в полях связи. Аналогичная связь существует и между записями (103, Серов С.С.) и (103, 3) этих же таблиц. В таблицах записи отсортированы по значениям в ключевых полях.

Сопоставление записей двух таблиц по существу означает образование новых «виртуальных записей» (псевдозаписей). Так, первую пару записей можно считать новой псевдозаписью вида (101, Алов А.А., 5).

На практике связи вида 1:1 используются сравнительно редко, так как хранимую в таких таблицах информацию легко объединить в одну таблицу, которая занимает гораздо меньше места в памяти ЭВМ.

Данную связь применяют, чтобы:

- ускорить обработку,
- повысить удобство работы нескольких пользователей с общей информацией,
- обеспечить более высокую степень защиты информации (приведенный пример).

Связь вида 1:M

Связь 1:M имеет место в том случае, когда одной записи основной таблицы соответствует несколько записей дополнительной таблицы.

Пример 2.

Имеются две связанные таблицы основная «Устройства» с информацией о видах мультимедиа-устройств КП и дополнительная «Склад» со сведениями о фирмах-производителях этих устройств, а также о наличии на складе хотя бы одного устройства.

Ключевое поле, поле связи	ТАБЛИЦА УСТРОЙСТВА
Код	Вид устройства
A	CD-ROM
B	CD-Recorder
C	DVD-ROM
D	DVD-Recorder

Ключевое поле, поле связи	Ключевое поле	ТАБЛИЦА СКЛАД
Код	Фирма-производитель	Наличие
A	Sony	Да
A	Teac	Нет
B	Teac	Нет
B	Acer	Да
C	Sony	Да

Таблица «Склад» имеет два ключевых поля, так как одна и та же фирма может производить устройства различных видов.

Сопоставление записей обеих таблиц по полю Код порождает прсевдозаписи вида: (A, CD-ROM, Sony, Да), (A, CD-ROM, Teac, Нет) и т. д.

Если свести псевдозаписи в новую таблицу, то получим полную информацию обо всех видах мультимедиа-устройств, фирмах-производителях, а также сведения о наличии конкретных видов устройств на складе.

Связь вида M:1

Связь M:1 имеет место в случае, когда одной или нескольким записям основной таблицы ставится в соответствие одна запись дополнительной таблицы.

Пример 4.

Основная таблица РАСПИСАНИЕ содержит информацию о преподавателях, работающих в одном из двух компьютерных классах. Дополнительная таблица КАБИНЕТ содержит информацию о лаборантах и техниках, обслуживающих тот или иной кабинет.

Поле связи	ТАБЛИЦА РАСПИСАНИЕ
Кабинет	Преподаватель
10	Иванов И.И.
10	Зотов З.З.
11	Егоров Е.Е.
11	Перов П.П.

Ключевое поле, поле связи	Ключевое поле	ТАБЛИЦА КАБИНЕТ
Кабинет	Лаборант	Техник
10	Львов Л.Л.	Лыков Л.Л.
11	Котов К.К.	Быков Б.Б.

Связывание этих таблиц обеспечивает такое установление соответствия между записями, которое эквивалентно образованию следующих псевдозаписей (10, Иванов И.И., Львов Л.Л., Лыков Л.Л.); (10, Зотов З.З., Львов Л.Л., Лыков Л.Л.) и т.п.

Полученная псевдотаблица может быть полезна при планировании или принятия управленческих решений, когда не обходимо иметь все возможные варианты.

Если таблицу КАБИНЕТ сделать основной, а таблицу РАСПИСАНИЕ - дополнительной, получим связь вида в 1:M. Отсюда следует, что вид связи (1:M или M: 1) зависит от того, какая таблица является основной, а какая дополнительной.

Связь вида M:M

Самый общий вид связи M:M возникает в случаях, когда нескольким записям основной таблицы соответствует несколько записей дополнительной таблицы.

Пример 5.

Пусть основная таблица РАБОТА содержит информацию о том, на каких станках могут работать рабочие некоторой бригады. Дополнительная таблица РЕМОНТ содержит сведения о том, кто из бригады ремонтников какие станки обслуживает.

ТАБЛИЦА РАБОТА	
Ключевое поле	Ключевое поле, поле связи
Работает	На станке
Алов А.А.	Станок1
Алов А.А.	Станок2
Бобов Б.Б.	Станок1
Волков В.В.	Станок2
Волков В.В.	Станок3

ТАБЛИЦА РЕМОНТ	
Ключевое поле	Ключевое поле, поле связи
Обслуживает	Станок
Гришин Г.Г.	Станок1
Гришин Г.Г.	Станок3
Данилов Д.Д.	Станок2
Данилов Д.Д.	Станок3

1-й и 3-й записям таблицы РАБОТА соответствует 1-ая запись таблицы РЕМОНТ (у всех этих записей значение второго поля — «станок 1»). 5-й записи таблицы РАБОТА соответствуют 2-я и 4-я записи таблицы РЕМОНТ (во втором поле этих записей содержится «станок3»)

Исходя из определения полей связи этих таблиц можно оставить новую таблицу с именем «РАБОТА+РЕМОНТ», записями которой будут псевдозаписи. Записям полученной таблицы можно придать смысл возможных смен, составляемых при планировании работы. Для удобства, поля новой таблицы переименованы.

Работа	Станок	Ремонт
Алов А.А.	Станок1	Гришин Г.Г.
Алов А.А.	Станок2	Данилов Д.Д.
Бобов Б.Б.	Станок1	Гришин Г.Г.
Волков В.В.	Станок2	Данилов Д.Д.
Волков В.В.	Станок3	Гришин Г.Г.
Волков В.В.	Станок3	Данилов Д.Д.

Аналогично связи 1:1, связь М:М не устанавливает подчиненности таблиц, т.е. основную и дополнительную таблицы можно поменять местами и выполнить объединение информации путем связывания - результирующие таблицы будут отличаться только порядком расположения полей и записей.

Примечание. На практике в связь обычно вовлекается сразу несколько таблиц. При этом одна таблица может иметь различного рода связи с несколькими таблицами. В случаях, когда связанные таблицы, в свою очередь, имеют связи с другими таблицами, образуется иерархия или дерево связей.

Из перечисленных видов связи чаще используется связь вида 1:М. Связь 1:1 можно считать частным случаем связи 1:М, когда одной записи главной таблицы соответствует одна запись вспомогательной таблицы. Связь М:1, по сути, является зеркальным отображением» связи 1:М. Оставшийся вид М:М характеризуется как слабый вид связи или даже как отсутствие связи.

4.4. Обеспечение целостности

Под ***целостностью*** понимают свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию.

Различают *физическую* и *логическую* целостность.

Физическая целостность означает наличие физического доступа к данным и то, что данные не утрачены.

Логическая целостность означает отсутствие логических ошибок в базе данных, к которым относятся нарушение структуры БД или ее объектов, удаление или изменение установленных связей между объектами и т. д.

Поддержание целостности БД включает проверку (контроль) целостности и ее восстановление в случае обнаружения противоречий в базе.

Целостное состояние БД задается с помощью *ограничений целостности* в виде условий, которым должны удовлетворять хранимые в БД данные.

Среди ограничений целостности можно выделить два основных типа ограничений:

1. *Ограничения значений* атрибутов отношений.
   Например, требование недопустимости пустых или повторяющихся значений в атрибутах, а также контроль принадлежности значений атрибутов заданному диапазону.
   Так, в записях отношений о кадрах значения атрибута Дата_рождения не могут превышать значений атрибута Дата_приема.
2. *Структурные ограничения* на кортежи отношений.
   2.1. Требование *целостности сущностей.*
   ((сущность – объект любой природы, данные о котором хранятся в отношении (таблице) БД))
   Каждому экземпляру сущности, представленному в отношении, соответствует только один его кортеж. Требование *целостности сущностей* состоит в том, что любой кортеж отношения должен быть отличим от любого другого кортежа этого отношения, т. е., иными словами, любое отношение должно обладать первичным ключом.
   2.2. Требование *целостности ссылок* состоит в том, что для каждого значения внешнего ключа родительской таблицы должна найтись строка в дочерней таблице с таким же значением первичного ключа.

Например, если в отношении R1 содержатся сведения о сотрудниках кафедры, а атрибут этого отношения Должность является первичным ключом отношения R2, то в этом отношении для каждой должности из R1 должна быть строка с соответствующим ей окладом.

Во многих современных СУБД имеются средства обеспечения контроля целостности БД.

R1. Родительская таблица

ФИО	Должность	Кафедра	Стаж
Иванов И.М.	преп	25	5
Петров М.И.	ст.преп	25	7
Сидоров Н.П	преп	25	10
Егоров ВВ.	преп	24	5

R2. Дочерняя таблица

Должность	Оклад
преп	500
ст.преп	800

Внешний ключ — Первичный ключ

Основные операции над данными двух таблиц с контролем целостности:

1. Ввод новых записей. Чтобы не допустить нарушение целостности, данные сначала вводятся в основную таблицу, а потом в дополнительную. В процессе заполнения основной таблицы контроль значений полей связи ведется как контроль обычного ключа (на совпадение полей других записей). Заполнение полей связи дополнительной таблицы контролируется на предмет совпадения со значениями полей связи основной таблицы. Если вновь вводимое значение в поле связи дополнительной таблицы не совпадет ни с одним соответствующим значением в записях основной таблицы, то ввод такого значения должен блокироваться.
2. Модификация записей. При редактировании полей связи дополнительной таблицы основным требованием является то, чтобы новое значение поля связи совпадало с соответствующим значением какой-либо записи основной таблицы, то есть дополнительная запись может сменить родителя, но остаться без него не должна.
   Редактирование поля связи основной таблицы должно подчинятся одному из правил: 1) блокировать модификацию полей связи, если есть подчиненные записи; 2) изменения в полях связи основной записи мгновенно передавать во все поля связи всех записей дополнительной таблицы (каскадное обновление).
3. Удаление записей. Удаление записей основной таблицы должно подчиняться одному из правил: 1) удалять можно запись, которая не имеет подчиненных записей; 2) запретить (блокировать) удаление записи при наличии подчиненных записей, либо удалять ее вместе со всеми подчиненными записями (каскадное удаление). Записи дополнительной таблицы можно удалять бесконтрольно.

4.5. Теоретические языки запросов

((Хранимые в БД данные можно обрабатывать вручную, последовательно просматривая и редактируя данные в таблицах с помощью имеющихся в СУБД соответствующих средств. Для повышения эффективности применяют запросы, по-

зволяющие производить множественную обработку данных, то есть одновременно вводить, редактировать и удалять множество записей, а также выбирать данные из таблиц)).

Запрос представляет собой специальным образом описанное требование, определяющее состав производимых над БД операций по выборке, удалению или модификации хранимых данных.

Для подготовки запросов чаще всего используют два основных языка описания запросов:

- язык QBE (Query By Example) – язык запросов по образцу;
- язык SQL (Structured Query Language) – структурированный язык запросов.

Главное отличие между ними заключается в способе формирования запросов: язык QBE предполагает ручное или визуальное формирование запроса, а язык SQL - с помощью программирования.

Характеристика языка QBE

Язык QBE позволяет задавать сложные запросы к БД путем заполнения предлагаемой СУБД запросной формы. Такой способ задания запросов обеспечивает высокую наглядность и не требует указания алгоритма выполнения операции — достаточно описать образец ожидаемого результата. В каждой из современных реляционных СУБД имеется свой вариант языка QBE.

На языке QBE можно задавать запросы ***однотабличные*** и ***многотабличные*** (выбирающие или обрабатывающие данные из нескольких связанных таблиц).

С помощью запросов на языке QBE можно выполнять следующие основные операции:

- выборку данных;
- вычисление над данными;
- вставку новых записей;
- удаление записей;
- модификацию (изменение) данных.

Результатом выполнения запроса является новая таблица, называемая ***ответной*** (первые две операции), или обновленная исходная таблица (остальные операции).

Выборка, вставка, удаление и модификация могут производиться безусловно или в соответствии с условиями, задаваемыми с помощью логических выражений. Вычисления над данными задаются с помощью арифметических выражений и порождают в ответных таблицах новые поля, называемые *вычисляемыми.*

Запросная форма имеет вид таблицы, имя и названия полей которой совпадают с именем и названиями полей соответствующей исходной таблицы. Чтобы узнать имена доступных таблиц БД, в языке QBE предусмотрен запрос на выборку имен таблиц. Названия полей исходной таблицы могут вводиться в шаблон вручную или автоматически. Во втором случае используется запрос на выборку заголовков столбцов.

В современных СУБД, например в Access и Visual FoxPro, многие действия по подготовке запросов с помощью языка QBE выполняются визуально с помощью мыши. В частности, визуальное связывание таблиц при подготовке запроса выполняется не элементами примеров, а просто «протаскиванием» мышью поля одной таблицы к полю другой.

Характеристика языка SQL

Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а также некоторых сопутствующих операций. SQL является *непроцедурным* языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т. п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД (например, FoxPro СУБД Visual FoxPro, ObjectPAL СУБД Paradox, Visual Basic for Applications СУБД Access).

((В современных СУБД с интерактивным интерфейсом можно создавать запросы, используя другие средства, например QBE. Однако применение SQL позволяет повысить эффективность обработки данных в базе. Например, при подготовке запроса в среде Access можно перейти из окна Конструктора запросов (формулировки запроса по образцу на языке QBE) в окно SQL, где, например, можно подготовить новый запрос путем редактирования уже имеющегося)).

Основным назначением языка SQL является подготовка и выполнение запросов.

Типы данных

В общем случае в разных СУБД могут использоваться различные типы данных.

Тип данных	**Размер**	**Описание**
BINARY	1 байт на	В поле этого типа могут храниться данные любого типа. Данные

	знак	не преобразуются (например, в текстовые). Данные отображаются в том же виде, в каком они вводятся в это поле.
BIT	1 байт	Значения «Да» (Yes) и «Нет» (No), а также поля, содержащие одно из двух возможных значений.
TINYINT	1 байт	Целое значение от 0 до 255.
MONEY	8 байтов	Масштабируемое целое от –922 337 203 685 477,5808 до 922 337 203 685 477,5807.
DATETIME (см. DOUBLE)	8 байтов	Дата или время; допустим любой год от 100 до 9999.
UNIQUEIDENTIFIER	128 битов	Уникальный идентификатор, используемый при вызовах удаленных процедур.
REAL, Single	4 байта	Число с плавающей точкой и одинарной точностью от –3,402823E38 до –1,401298E-45 для отрицательных значений, от 1,401298E-45 до 3,402823E38 для положительных значений или значение 0.
FLOAT	8 байтов	Число с плавающей точкой и двойной точностью от –1,79769313486232E308 до –4,94065645841247E-324 для отрицательных значений, от 4,94065645841247E-324 до 1,79769313486232E308 для положительных значений или значение 0.
SMALLINT	2 байта	Короткое целое от –32 768 до 32 767 (см. «Примечания»).
INTEGER	4 байта	Длинное целое от –2 147 483 648 до 2 147 483 647.
DECIMAL	17 байтов	Тип данных для хранения точных числовых значений от -10^28 - 1 до 10^28 - 1. Точность (1 - 28) и фактор масштабирования (от 0 до заданной точности) определяются пользователем. По умолчанию точность и фактор масштабирования равны соответственно 18 и 0.
TEXT (он же MEMO)	2 байта на знак	От 0 до 2,14 Гбайт.
IMAGE	Не ограничено	От 0 до 2,14 Гбайт. Используется для объектов OLE.
CHARACTER (?CHAR, он же TEXT(n) с указанной длиной	2 байта на знак	От 0 до 255 знаков.

Опишем язык SQL на примере версии, используемой в СУБД Microsoft Access.

Для того, чтобы перейти в окно языка SQL надо:
1. В главном меню выбрать команду *Запросы*.
2. В появившемся окне выбрать *Создать*.
3. Выбрать режим *Конструктор*.
4. Закрыть появившееся окно.
5. В верхнем меню выбрать команду *Вид \ Режим SQL*.

Основные (не все) операторы языка

Операторы языка SQL можно условно разделить на два вида:
1. Язык определения данных (Data Definition Language — DDL). Операторы:
   - Create Table - создание таблицы
   - Drop Table – удаление таблицы
   - Alter Table – изменение структуры таблицы
   - Create Index – создание индекса
   - Drop Index – удаление индекса
   - Create View – создание представления
   - Drop View – удаление представления

2. Язык манипулирования данными (Data Manipulation Language - DML). Операторы:
   - Select – выборка записей
   - Update – изменение записей
   - Insert – вставка новых записей
   - Delete - удаление записей

Рассмотрим формат и основные возможности операторов (в квадратных скобках указаны необязательные поля).

1. Создание таблицы.

Create Table *Имя таблицы* (*Имя поля Тип данных* [Not Null] [, *Имя поля Тип данных* [Not Null]]);

Обязательными операндами являются имя создаваемой таблицы и имя хотя бы одного поля с указанием типа данных.

При создании таблиц для отдельных полей могут указываться дополнительные правила контроля вводимых в них значений. Например, опция Not Null указывает, что данное поле является обязательным для заполнения.

Пример 1. Создать таблицу Товары с полями *Название*, *Производитель*, *Цена*.

Create table *Товары* (*Название* Char(12) not Null, *Производитель* Char(12) not Null, *Цена* Single);

2. Изменение структуры таблицы.

Alter Table *Имя таблицы* {Add, Modify, Drop} *Имя поля* [*Тип данных*] [Not Null]
[,{Add, Modify, Drop} *Имя поля* [*Тип данных*] [Not Null]]…];

Добавление (Add), изменение (Modify) или удаление (Drop) одного или нескольких столбцов.

При удалении столбца указывать тип данных не надо.

Пример 2. Добавить поле *Количество* в таблицу *Товары*.

Alter Table *Товары* add *Количество* Real;

Пример 3. Удалить в таблице *Товары* поле *Производитель*.

Alter Table *Товары* Drop *Производитель*;

3**. Удаление таблицы.**

Drop Table *Имя таблицы*;

Пример 4. Удалить таблицу Товары.

Drop Table *Товары*;

4. Создания индекса.

Create [Unique] Index *Имя индекса* ON *Имя таблицы*
(*Имя столбца* [ASC | DESC] [, *Имя столбца* [ASC | DESC]...);

Оператор позволяет создать индекс для одного или нескольких столбцов данной таблицы.

Задав необязательную опцию Unique, можно обеспечить уникальность значений во всех указанных в операторе столбцах. По существу, создание индекса с указанием признака Unique означает определение ключа в созданной ранее таблице.

При создании индекса можно задать порядок автоматической сортировки значений в столбцах — в порядке возрастания ASC (по умолчанию)или в порядке убывания DESC. Для разных столбцов можно задавать различный порядок сортировки.

Пример 5. Для таблицы *Сотрудники*, имеющей поля: *ФИО*, *Отдел*, *Оклад* создать индекс *Код* для сортировки имен в алфавитном порядке и убыванию размеров зарплаты.

Create Index *Код* ON *Сотрудники* (*ФИО*, *Оклад* DESC);

Или

Create Index *Код* ON *Сотрудники* (*ФИО* ASC, *Оклад* DESC);

5. Удаление индекса:

DROP INDEX *Имя индекса*;

6. Добавление записей.

Оператор *вставки новых записей* имеет форматы двух видов:

1) Добавление одной записи:

Insert Into *Имя таблицы* [(*Список столбцов*)] Values (*Список значений*);

Предназначен для ввода новых записей с заданными значениями в столбцах. Порядок перечисления имен столбцов должен соответствовать порядку значений, перечисленных в списке операнда Values. Если *Список столбцов* опущен, то в *Списке значений* должны быть перечислены все значения в порядке столбцов структуры таблицы. Строковые значения пишутся в апострофах.

2) Добавление нескольких записей:

Insert Into *Имя таблицы* [(*Список столбцов*)] *Предложение*_Select;

Предназначен для ввода в заданную таблицу новых строк, отобранных из другой таблицы с помощью предложения Select.

Пример 6. Ввести в таблицу *Сотрудники* запись о новом сотруднике.

Insert Into *Сотрудники* (*ФИО*, *Отдел*, *Руководитель*, *Оклад*)
Values (*'Иванов И.И.', 'ИВЦ', 'Кротов К.К.', 7500*);
Или
Insert Into *Сотрудники*
Values (*'Иванов И.И.', 'ИВЦ', 'Кротов К.К.', 7500*);
Или
Insert Into *Сотрудники* (*ФИО*, *Оклад*)
Values (*'Перов П.П.', 6500*);

7. Изменение записей:

Update *Имя таблицы*
Set *Имя столбца*={*Выражение* |Null}
[, Set *Имя столбца*={*Выражение* |Null}, …]
[Where Условие]

Выполнение оператора Update состоит в изменении значений в определенных операндом Set столбцах таблицы для тех записей, которые удовлетворяют условию, заданному операндом Where.

Новые значения полей в записях могут быть пустыми (Null), либо вычисляться в соответствии с арифметическим выражением.

((Правила записи арифметических и логических выражений аналогичны соответствующим правилам оператора Select))

Пример 7. Увеличить на 500 единиц зарплату тем служащим, которые получают не более 6000 (по таблице *Сотрудники*).

Update *Сотрудники* Set *Оклад*=*Оклад*+500 Where *Оклад*<= 6000;

8. Удаление записей.

Delete From *Имя таблицы* [Where *Условие*]

Результатом выполнения оператора Delete является удаление из указанной таблицы строк, которые удовлетворяют условию, определенному операндом Where. Если операнд Where опущен ((, то есть условие отбора удаляемых записей отсутствует,)) удалению подлежат все записи таблицы.

Отличие от оператора Drop – структура таблицы остается, удаляются только записи.

Пример 8. В связи с ликвидацией отдела *ИВЦ*, требуется удалить из таблицы *Сотрудники* всех сотрудников этого отдела.

Delete From *Сотрудники* Where *Отдел='ИВЦ'*;

9. Выборка записей:

Select [All | Distinct] *Список_данных* From *Список_таблиц*
[Where *Условие выборки*]
[Group By *Имя столбца* [, *Имя столбца*]...]
[Having *Условие поиска*]
[Order By *Спецификация* [, *Спецификация*]...]

Это наиболее важный оператор из всех операторов SQL. Функциональные возможности его огромны. Рассмотрим основные из них.

9.1. Оператор *Select* позволяет производить выборку и вычисления над данными из одной или нескольких таблиц. Результатом выполнения оператора является ответная таблица, которая может иметь (*ALL*), или не иметь (*DISTINCT*) повторяющиеся строки. По умолчанию в ответную таблицу включаются все строки, в том числе и повторяющиеся.

9.2. *Список данных* может содержать имена столбцов, участвующих в запросе, а также выражения над столбцами. ((В простейшем случае в выражениях можно записывать имена столбцов, знаки арифметических операций (+, — ,*,/), константы и круглые скобки. Если в списке данных записано выражение, то наряду с выборкой данных выполняются вычисления, результаты которого попадают в новый (создаваемый) столбец ответной таблицы.))

При использовании в списках данных имен столбцов нескольких таблиц для указания принадлежности столбца некоторой таблице применяют конструкцию вида: *Имя таблицы*.*Имя столбца*.

9.3. В отборе данных участвуют записи одной или нескольких таблиц, перечисленных в списке *From*.

Пример 9. Выбор записей.
Для таблицы *Сотрудники*, имеющей поля: *ФИО*, *Отдел*, *Руководитель*, *Оклад* вывести все поля и записи таблицы.

Select * From *Сотрудники*;

Пример 10. Выбор записей с вычислением.
Вывести имена сотрудников и размер оклада, увеличенный на 100 единиц.

Select *ФИО, Оклад+100* From *Сотрудники*;

9.4. Операнд *Where* задает условия, которым должны удовлетворять записи в результирующей таблице.

9.5. *Условие выборки* является логическим. Его элементами могут быть имена столбцов, операции сравнения, арифметические операции, логические связки (И, ИЛИ, НЕТ), скобки, специальные функции:

Like – подобный (например вывести сотрудников на А),
Null – нет значения,
In - принадлежит множеству значений,
Not In – не принадлежит множеству значений,
Between *Значение* - между какими-то значениями и т. д.

Пример 11. Выбор с условием.
Вывести названия таких отделов таблицы *Сотрудники*, в которых в данный момент нет руководителя.

```
Select Отдел
From Сотрудники
Where Руководитель is Null;
```

Пример 12. Выбор с условием.
Вывести ФИО и оклад сотрудников, у которых оклад от 5000-7500.

```
Select ФИО, Оклад
From Сотрудники
Where Оклад Between 5000 and 7500;
```

9.6. В логических и арифметических выражениях можно использовать функции:

AVG - среднее значение в группе,
MAX - максимальное значение в группе,
MIN - минимальное значение в группе,
SUM - сумма значений и группе,
COUNT - число значений в группе.

Пример 13. Подсчитать количество сотрудников в отделе, которым руководит Иванов.

```
Select COUNT(Руководитель)
From Сотрудники
Where Руководитель='Иванов';
```

9.7. Операнд *Group By* позволяет выделять в результирующем множестве записей *группы* – ((*Группой* являются)) записи с совпадающими значениями в столбцах, перечисленных за ключевыми словами *Group By*. В *Group By* указывают все поля, перечисленные после операнда *Select* ((Выделение групп требуется для использования в логических выражениях операндов Where и Having, а также для выполнения операций (вычислений) группами)).

Пример 14. Выбор с группированием.

```
Select Отдел, MIN(Оклад), MAX(Оклад)
```

Найти минимальную и максимальную зарплаты для каждого из отделов таблицы *Сотрудники*.	From *Сотрудники* Group by *Отдел*;

9.8. Операнд *Having* действует совместно с операндом *Group By* и используется для дополнительного отбора записей во время определения групп. Правила записи *Условия поиска* аналогичны правилам формирования *Условия выборки* операнда *Where*.

Пример 15. Выбор с группированием и отбором. Вывести те отделы, у которых минимальный оклад меньше 5000.	Select *Отдел,* MIN(*Оклад*) From *Сотрудники* Group by *Отдел* Having MIN(*Оклад*)<5000;

9.9. Операнд *Order By* задает порядок сортировки результирующего множества. Каждая *Спецификация* ((аналогична соответствующей конструкции оператора *Create Index* и)) представляет собой пару вида: *Имя столбца* [*ASC* | *DESC*]. (возрастнаие/убывание)

Пример 16. Сортировка. Вывести всю информацию по сотрудникам, отсортировав ФИО по алфавиту.	Select * From *Сотрудники* Order By *ФИО* ASC;

Замечание.

Оператор *Select* может иметь и другие более сложные синтаксические конструкции:

- *Подзапросы* позволяют формулировать *вложенные запросы,* когда результаты одного оператора *Select* используются в логическом выражении условия выборки операнда *Where* другого оператора *Select*, в этом случае вложенные подзапросы пишутся в ();

Пример 17. Вложенный подзапрос.

Вывести всю информацию о сотрудниках, оклад которых больше среднего.

```
Select *
From Сотрудники
Where Оклад > (Select AVG (Оклад) From Сотрудники);
```

- выполнить объединение результирующих таблиц при выполнении нескольких операторов *Select* (операнд *Union*);

Пример 18. Вложенный подзапрос.

Вывести всю информацию о клиентах и поставщиках из двух соответствующих таблиц БД, правление которых находится в Москве.

```
Select Название
From Поставщики
Where Город='Москва'
Union Select Название
      From Клиенты
      Where Город='Москва';
```

БАЗЫ ДАННЫХ ACCESS

Термины и определения.

В MS Access **базой данных** называется совокупность таблиц, форм, отчетов, запросов, модулей, макросов. Вся эта совокупность запоминается в одном файле базы данных, имеющем расширение *.mdb*.

Для работы с базами данных в Access имеется стандартное окно, из которого можно вызвать любой ее объект просмотра, выполнения, разработки и модификации. Пользователь для работы с базой данных может разработать свой интерфейс, основу которого обычно составляют *формы*

((На формах размещаются различные элементы, такие как: поля таблиц, поля со списком, кнопки, раскрывающиеся списки, выключатели, переключатели, флажки, рисунки, подчиненные формы и т. д.))

((За кнопками обычно закрепляют вызов функций. Функции обработки информации во время работы с базой данных задаются с помощью макросов или программ на Visual Basic for Application (VBA) — VBA-программ.))

Таблица представляет собой основную единицу хранения данных в базе. В произвольной БД обычно имеется совокупность связанных между собой таблиц. Между двумя таблицами можно устанавливать связи типа 1:1, 1:M, M:1 с помощью окна описания схемы данных. Основными операциями над таблицами являются: просмотр и обновление (ввод, модификация и удаление), сортировка, фильтрация, и печать.

Форма представляет собой объект базы данных Access, в котором разработчик размещает элементы управления, принимающие действия пользователей или служащие для ввода, отображения и изменения данных в полях.

Запрос представляет собой формализованное требование на отбор данных из таблиц или на выполнение определенных действий с данными. Запрос позволяет создать набор записей из данных, находящихся в разных таблицах, и использовать его как источник данных для формы или отчета. В Access можно создавать и выполнять следующие основные типы запросов: на выборку, обновление, удаление, или добавление данных. ((С помощью запросов можно также создавать новые таблицы, используя данные из одной или нескольких существующих таблиц)).

Описание запроса можно выполнить с помощью бланка QBE или инструкции языка SQL.

Макрос представляет последовательность макрокоманд встроенного языка Access, задающих автоматическое выполнение некоторых операций, например: «ОткрытьТаблицу» (OpenTable), «Закрыть» (Close), «НайтиЗапись» (FindRecord) и «Печать» (PrintOut).

Модуль представляет совокупность описаний, инструкций и процедур на языке VBA, сохраненную под общим именем. В Access используются модули трех типов: формы, отчета и стандартный. ((Модули форм и отчетов содержат программы, являющиеся локальными для этих объектов. Процедуры из стандартного модуля, если они не описаны явно как локальные для содержащего их модуля, распознаются и могут вызываться процедурами из других модулей в той же базе данных или из адресуемых баз данных.

Access поддерживает традиционные для офисных приложений механизмов связывания и встраивания объектов OLE (Object Linking and Embedding) и динамического обмена данными DDE (Dynamic Data Exchange).))

1. Создание БД

Access предоставляет два способа создания базы данных:

1) Создание пустой БД (в последующем можно добавить нужные объекты).
   Данный способ отличается большей гибкостью и трудоемкостью, так как требует отдельного определения каждого элемента базы данных.
   Чтобы создать пустую БД надо выбрать *Файл\Создать новую БД*, в появившемся окне на вкладке «*Общие*» производится переход к созданию новой БД, для чего нужно будет задать папку и имя базы данных.
2) Создание непустой БД с помощью *Мастера*. Данный способ ускоряет процесс создания БД и позволяет получить БД с образцами информации в таблицах. Он применим в случаях, когда пользователю подходит одна из предлагаемых типов БД.
   Чтобы создать непустую БД надо выбрать *Файл \ Создать* и пойти по ссылке «*Создание с помо-*

щью шаблона и на вкладке *Базы данных* выбрать как основу одну из многих готовых БД (например, Контакты. Склад и т.п.)

2. Создание основных элементов БД

К основным элементам базы данных можно отнести таблицы, запросы, формы, отчеты, макросы и модули.

2.1. Создание таблиц

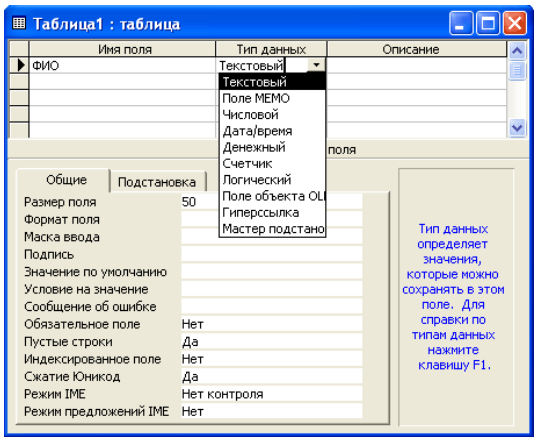
((Перед созданием таблицы нужно открыть базу данных, в которой таблица будет находиться.
В открытой БД следует выбрать вкладку *Таблицы* и нажать кнопку *Создать* (New). Либо *Вставка* \ *Таблица*.))

Возможны пять вариантов создания таблиц:

1) *Режим таблицы* - путем ввода данных в пустую таблицу, при сохранении данных в которой Access анализирует данные и автоматически присваивает соответствующий тип данных и формат каждому полю;
2) С помощью Конструктора;
3) С помощью Мастера*;*
4) Путем копирования структуры ранее созданной таблицы;
5) *Импорт таблиц* - импортируемых таблиц*;*
6) *Связь с таблицами* - путем создания таблиц, связанных с таблицами, находящимися во внешнем файле.

2.1.1. Создание таблиц в режиме конструктора

Для создания новой таблицы в режиме *Конструктор* в окне *Базы Данных* надо выбрать объект *Таблица* и соответствующий режим ее создания, после чего появится окно для описания структуры таблицы и других ее характеристик.



В этом окне надо последовательно описать все поля создаваемой таблицы: задать имя, выбрать тип данных, задать требуемые свойства.

При желании можно воспользоваться готовыми структурами таблиц, выбрав в контекстном меню *Построить*. Появляется окно, в котором можно выбрать готовые описания полей.

Примечание. Независимо от способа создания, изменять структуру таблицы придется делать в режиме *Конструктор.*

При описании структуры таблицы следует обратить внимание на свойство *Индексированное поле {Indexed).* Оно может принимать значения: *Нет (No)* - не индексированное, *Да (Допускаются совпадения (Duplicates OK))* и *Да (Совпадения не допускаются) (Yes (No Duplicates).* Индексация поля в системе Access еще не означает, что поле является ключевым. Чтобы сделать поле ключевым, нужно сначала задать свойства поля (полноценно ключевым оно станет только при индексации этого поля, не допускающей совпадения), затем выделить строку описания поля и нажать кнопку *Ключ* на панели инструментов.

Считается нормой, когда таблица имеет хотя бы одно ключевое поле.

Ключ может создаваться системой автоматически или определяться вручную при описании таблицы.

В первом случае ключ первоначально описывать не надо. При завершении описания таблицы надо подтвердить необходимость его создания, ответив «*Да»* на вопрос «*Создать ключевое поле сейчас?*». Ключевое поле будет создано автоматически, иметь имя *Код*, тип данных - *Счетчик* и иметь признак ключа. Для удобства дальнейшей работы рекомендуется переименовывать ключевое поле так, чтобы было понятно кодом чего является данное поле (например, «Код_сотрудника», «Код_кафедры» и т.п.). Для этого необходимо загрузить только что созданную таблицу в режиме конструктора и переименовать соответствующее поле.

Во втором случае, когда ключ определяется самостоятельно, необходимо:

1) открыть таблицу в режиме конструктора;
2) выделить одно или несколько полей, которые требуется определить как поля первичного ключа;
3) выбрать один из возможных путей:

- позиционироваться на соответствующее поле и нажать на кнопку *Ключевое поле*  ;
- выбрать команду меню *Правка* \ *Ключевое поле*;
- воспользоваться правой кнопкой мыши для вызова контекстного меню.

Примечание. Для выделения одного поля надо щелкнуть область выделения строки (прямоугольник с левого края строки или полоса на левом крае окна, при выборе которых выделяется вся строка в таблице или в режиме конструктора) нужного поля.

Для выделения нескольких полей (для создания *составного ключа*) необходимо выделить строку с первым полем, а все остальные поля, входящие в ключ выделить, удерживая нажатой клавишу CTRL.

Использование мастера подстановок

При создании некоторых полей можно использовать мастер подстановки.

Для использования возможности подстановки можно либо, описывая поле, указать тип данных *Мастер подстановок* (см. последнюю строку в ниспадающем списке типов данных), либо выбрать команду *Вставка* \ *Поле подстановки*. После чего появится окно «*Создание подстановки*».

Возможно два способа подстановки:

1) Подстановка из фиксированного набора значений:

- В окне *Создание подстановок* выбрать *Будет введен фиксированный набор значений*.
- В появившейся таблице ввести список значений.
- Задать имя поля.

2) Подстановка из другой таблицы/запроса.

Данный вариант подстановки следует использовать, если число значений поля подстановки достаточно велико и сами значения могут меняться со временем.

В этом случае таблица, из которой будет производиться подстановка, должна быть уже создана и обе таблицы должны быть связаны между собой.

После этого надо открыть таблицу в режиме конструктора и в окне *Создание подстановок* выбрать *Объект* «*столбец подстановки*» *будет использовать значения из таблицы или запроса*; затем выбрать таблицу или запрос в качестве источника и определить поле таблицы-источника, значения из которого будут подставляться в описываемую колонку. На последнем шаге задать имя поля.

2.1.2. Создание таблиц с помощью мастера

Создание таблиц с помощью мастера в Access представляет собой выбор из имеющегося в системе списка таблиц делового и личного применения. Список таблиц высвечивается в окне «Образцы таблиц». После выбора таблицы пользователь может определить те поля, которые нужны ему для данного применения.

Выбранные поля можно переименовать. Чтобы ввести какие-либо другие изменения в структуру создаваемой таблицы, следует завершить ее формирование с помощью мастера, после чего откорректировать структуру в обычном порядке в режиме «Конструктор».

Создавая таблицу с использованием мастера, нельзя отказаться от ключа, а также создать составной ключ.

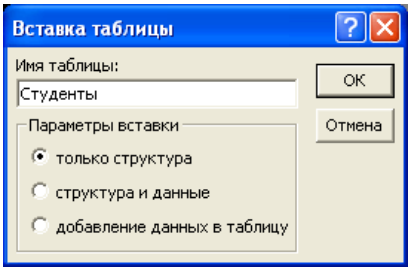
2.1.3. Создание таблиц путем ввода данных

Создание таблиц путем ввода данных является не очень удобным способом создания таблиц, так как может повлечь за собой неочевидные недостатки в проектировании (например, неточное определение типа поля), а также требует большой последующей корректировки полученной структуры таблицы (переименование полей, определение ключа, если ключ задается не автоматически).

2.1.4. Создание таблиц путем копирования структуры существующей таблицы

Данный способ используют, если структура вновь создаваемой таблицы совпадает либо незначительно отличается от какой-либо уже существующей таблицы.

Для этого надо позиционироваться на таблице, структура которой будет копироваться, и выбрать команду меню *Правка* / *Копировать*, потом *Правка* / *Вставить*, после чего в появившемся окне ввести имя вновь создаваемой таблицы, а в качестве параметра вставки выбрать либо *Только структура* либо Структура и данные. Структура созданной таким образом таблицы может быть впоследствии скорректирована обычным способом.



2.1.5. Создание таблиц другими способами

Создать таблицу можно и путем импорта ее из других систем. Кроме того, в виде таблицы можно сохранить результат запроса.

I. Импорт данных.

1. Команда верхнего меню *Файл* \ *Внешние данные* \ *Импорт*.
2. В появившемся окне изменить тип файла на *Microsoft Excel* после чего загрузить файл с таблицей.
3. Запустить мастер *Импорт электронной таблицы*, с помощью которого последовательно переходя из одного окна в другое (кнопка *Далее*) выполнить действия:
   - Выбрать лист, с которого будут импортированы данные (данное окно может отсутствовать, если в книге только один лист).
   - Установить флажок *Первая строка содержит заголовки столбцов*.
   - Установить переключатель в положение *Данные необходимо сохранять в новой таблице*.
   - Следующий шаг можно пропустить.
   - Выбрать один из вариантов создания ключа *Автоматически создать ключ* (если в таблице нет столбца с подходящими данными) или *Определить ключ*, после чего в окне правее указать название столбца.
   - На пятом шаге в окне *Импорт в таблицу* задать имя таблицы.
4. После того, как таблица будет создана ее можно изменить, загрузив в режиме конструктора.

3. Фильтрация данных

Работая с таблицей можно установить фильтр, т.е. задать логическое выражение, которое позволит выдавать на экран только те записи, для которых это выражение выполняется.

В Access существует три вида фильтрации данных:

1. Обычный фильтр - отбор записей по содержимому нескольких полей;
2. Фильтр по выделенному фрагменту - отбор записей путем выделения данных;
3. Расширенный фильтр - построение более сложного фильтра.

Чтобы установить ***обычный*** фильтр надо:

1. Открыть таблицу.
2. *Записи* \ *Фильтр* \ *Изменить фильтр.*
3. В появившемся окне в нужных полях указать критерий отбора (например, в поле *Пол* установить значение "*ж"* или в поле *Оклад* >5000).
4. *Фильтр* \ *Применить фильтр.*

В результате фильтрации будут показаны только те записи, для которых заданное условие выполняется.

Чтобы установить ***фильтр по выделенному*** фрагменту надо:

1. Выделить данные, которые будут использоваться в качестве критерия фильтрации (например, выделим строку со значением «*ж»* в поле *Пол*).
2. *Записи* \ *Фильтр* \ *Фильтр по выделенному.*

Чтобы установить ***расширенный фильтр*** надо:

1. Открыть таблицу.
2. *Записи* \ *Фильтр* \ *Расширенный фильтр*.
3. В нижнюю часть раскрывшегося окна перетащить с помощью мыши поля таблицы, используемые в фильтре.

4. Для каждого из полей можно установить в строке *Сортировка* тип сортировки, а в строке *Условие отбора* – критерий.
5. *Фильтр \ Применить фильтр.*

Примечания.

- Все команды работы с фильтром продублированы кнопками на панели инструментов.
- При фильтрации нельзя отключить отображение отдельных полей и выполнить вычисления, в этих случаях необходимо использовать запрос на выборку.
- Для удаления фильтра и вывода на экран всех записей таблицы предназначена команда *Записи \ Удалить фильтр*.

4. Связывание таблиц. Обеспечение целостности

После создания в базе данных отдельных таблиц по каждой теме необходимо выбрать способ, которым Microsoft Access будет вновь объединять сведения таблиц. Первым делом следует определить связи между таблицами. После этого можно создать запросы, формы и отчеты для одновременного отображения сведений из нескольких таблиц.

4.1. Связывание таблиц

Чтобы связать таблицы надо:

1. Выбрать команду меню *Сервис \ Схема данных* (либо нажать на соответствующую кнопку  на панели инструментов).
2. В открывшемся окне *Схема данных* добавить в окно те таблицы, между которыми будет определяться связь.
3. Таблицы, между которыми определяется связь, чаще всего связаны отношением 1:М. Для установления связи надо позиционироваться на поле связи (обычно это первичный ключ) в основной таблице, нажать на левую клавишу мыши и, не отпуская ее, перетащить появившийся значок на соответствующее поле в «зависимой» таблице.
4. В появившемся окне *Изменение связи* следует определить, надо ли задавать ограничения целостности связи, и если да, то выбрать режимы корректировок (обновления и удаления).
   Примечание. Если задается ограничения целостности, то поле связи основной записи должно быть проиндексировано (для ключей индексация задается автоматически).
5. В окне *Изменение связи*, нажав на кнопку *Объединение* можно выбрать один из трех видов связи:
   1) 1:1, объединение записей, в которых связанные поля таблиц совпадают;
   2) 1:М, объединение всех записей из дополнительной таблицы и тех записей основной таблицы, в которых связанные поля совпадают;
   3) М:1, объединение всех записей из основной таблицы и тех записей дополнительной таблицы, в которых связанные поля совпадают.

4.2. Удаление связи

1. Закрыть все открытые таблицы - удалять связи между открытыми таблицами нельзя.
2. Для перехода в окно базы данных нажать клавишу F11.
3. Нажать кнопку *Схема данных*  на панели инструментов.
4. Если таблиц, связи которых нужно удалить, нет на экране, нажать кнопку *Отобразить таблицу*  на панели инструментов и дважды щелкнуть все таблицы, которые нужно добавить, затем нажать кнопку *Закрыть*.
5. Выделить линию связи, которую необходимо удалить (выделенная линия становится жирной), а затем нажать клавишу Delete.

4.3. Задание ограничений целостности

Обеспечение целостности БД является одной из важнейших задач при создании БД.

При изложении вопросов создания и связывания таблиц мы уже касались некоторых аспектов обеспечения целостности БД. Рассмотрим другие возможности ограничений целостность.

В Access многие ограничения целостности могут задаваться при создании таблицы.

4.3.1. Ограничения, относящиеся к полю

Тип поля. Тип поля определяет допустимые символы, которые могут быть использованы при его заполнении; например, не допускается ввод текста в числовые поля.

Для некоторых типов полей, например поля типа «Дата», осуществляется и более сложная проверка. Если допущена ошибка в типе данных или неправильно введена дата, то пользователь должен обязательно исправить ошибку, так как СУБД не дает других возможностей продолжить работу.

Многие из свойств полей также позволяют обеспечивать контроль целостности. Такие свойства полей, как:

- размер поля
- формат поля
- маска ввода
- значение по умолчанию
- условия на значения
- сообщение об ошибке
- обязательное поле
- пустые строки
- индексированное поле

в той или иной степени связаны с ограничениями целостности.

Поясним использование некоторых из перечисленных выше свойств в целях обеспечения контроля целостности на отдельных примерах.

Размер поля. В поле нельзя ввести больше символов, чем это и зафиксировано в свойстве «Размер поля» или предопределено типом поля.

Условия на значения. Одной из самых гибких возможностей определения ограничений целостности является задание «Условия на значения». Условия вводятся как выражения. Выражения могут быть простыми или сложными. Используя их, можно задавать и диапазоны.

Пользователь может задать в свойстве «Сообщение об ошибке» свой текст.

После задания такого условия поле становится Обязательным.

Пример 1. Организация была создана 1 января 2000 года. Определить условие на значение для поля «Дата_приема_на_работу».

Ответ: >#1.1.2000#

(Значения-даты необходимо заключать в символы номера (#)).

Пример 2. Для поля «Пол» задать ограничение на ввод значений.

Ответ: «м» OR «ж»

Пример 3. Для поля «Оклад» задать ограничение.

Ответ: >=500 And <=20000

Маска ввода.

Маски ввода могут быть использованы для того, чтобы ограничить символы, которые могут быть использованы при заполнении значения поля, и это может рассматриваться как один из способов задания ограничений целостности.

При написании маски в Microsoft Access используют следующие знаки:

Знак	Описание
0	Цифра (от 0 до 9, ввод обязателен; знаки плюс [+] и минус [-] не допускаются).
9	Цифра или пробел (ввод не обязателен; знаки плюс и минус не допускаются).
#	Цифра или пробел (ввод не обязателен; пустые знаки преобразуются в пробелы, допускаются знаки плюс и минус).
L	Буква (от A до Z или от А до Я, ввод обязателен).
?	Буква (от A до Z или от А до Я, ввод не обязателен).
A	Буква или цифра (ввод обязателен).
a	Буква или цифра (ввод необязателен).

&	Любой знак или пробел (ввод обязателен).
C	Любой знак или пробел (ввод необязателен).
. , : ; - /	Десятичный разделитель и разделители тысяч, значений дат и времени. (Отображаемый знак зависит от настроек языка и стандартов на панели управления Microsoft Windows.)
<	Указывает перевод всех следующих знаков на нижний регистр.
>	Указывает перевод всех следующих знаков на верхний регистр.
!	Указывает заполнение маски ввода справа налево, а не слева направо. Заполнение маски знаками всегда происходит слева направо. Восклицательный знак в маске ввода можно помещать в любую позицию.
\	Указывает ввод любого следующего знака в качестве текстовой константы. Используется для отображения всех перечисленных в данной таблице знаков как текстовых констант (например, \A выводится как знак «A»).
Пароль	Значение **Пароль** создает поле для ввода пароля. Любой знак, введенный в поле, сохраняется как знак, но отображается как звездочка (*).

Примечания.

1) Чтобы включить в маску текстовые константы, отличные от представленных в таблице, в том числе знаки и пробелы, следует просто ввести их в нужную позицию.
2) Чтобы включить один из следующих знаков в качестве текстовой константы, необходимо перед ним ввести знак обратной косой черты (\).

Задания.

№ 1. Создать для поля «Телефон» маску ввода, обеспечивающую обязательный ввод четырехзначного цифрового кода города в круглых скобках и далее шестизначный номер по три цифры с дефисом.
Маска: (0000) 999-999
Пример: (3843) 741-147

№ 2. Создать для поля «Мобильный телефон» маску ввода, обеспечивающую автоматический вывод первых трех символов номера «8-9» и обязательный ввод оставшихся цифр сгруппированных по 3 и 2 цифры, разделенных дефисом.
Маска: 8-\900-000-00-00
Пример: 8-901-044-74-47

№ 3. Создать для поля «Фамилия» маску ввода, которая позволит использовать только буквы при вводе, обеспечит преобразование первого символа в верхний регистр, а всех остальных - в нижний.
Маска: >L<??????????????????
Пример: Мария

№ 4. Создать для поля «Номер автомобиля» маску ввода, которая позволит вводить номер в формате: в круглых скобках код страны из двух символов, затем три заглавные буквы и четырехзначный номер разбитый на две группы разделенных дефисом, все символы обязательны.
Маска: (00)>LLL 0-000
Пример: (72)КЕМ 4-605

Индексированное поле можно использовать для контроля на уникальность. В Access при определении значения свойства «Уникальный индекс» в это поле не допускается ввод повторяющихся значений.

Ключевое поле индексируются автоматически, после чего запрещается ввод повторяющихся или пустых значений ключа.

4.3.2. Ограничения, относящиеся к записи

Если ограничения целостности затрагивают несколько полей одной записи, то они должны задаваться не как свойство записи, а как свойство таблицы. Для этого надо в режиме «Конструктор» щелкнуть правой кнопкой мыши по заголовку таблицы, выбрать «Свойства» и в появившемся окне задать условие.

Например, [Дата_окончания_института]>[Дата_рождения]

5. Создание запросов

Для создания запроса надо выбрать вкладку *Запросы* нажать кнопку *Создать* или *Вставка \ Запрос*.

В результате появится окно, в котором предлагается выбор варианта запроса:

1. *Конструктор* – создать вручную «с нуля».
2. *Простой запрос* служит для создания простых запросов на основе выбранных полей.
3. *Перекрестный запрос* автоматическое создание перекрестного запроса для компактного представления данных в виде сводной (перекрестной) таблицы.
4. *Повторяющиеся записи* -автоматическое создание запроса на поиск записей с повторяющимися значениями полей.
5. *Записи подчиненных* - автоматическое создание запроса на поиск записей в одной таблице, которые не имеют подчиненных записей в другой таблице.

Создаваемые запросы основаны на полях таблиц и/или запросов из БД. Все способы, кроме первого, реализуются с помощью Мастеров.

Если созданный запрос не удовлетворяет требованиям, то можно воспользоваться Конструктором, либо создать заготовку запроса с помощью Мастера, которую затем подправить в режиме Конструктора.

5.1. Создание запросов

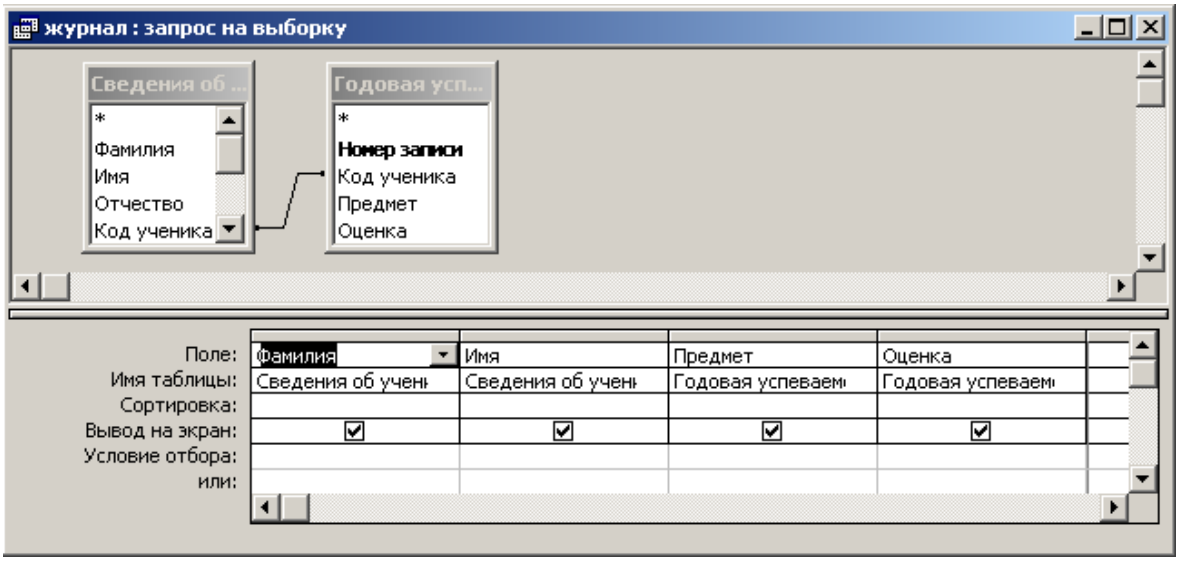
При создании макета запроса в общем случае необходимо выполнить следующие базовые операции:

1) В основном меню выбрать вкладку *Запросы* и нажать кнопку *Создать*, после этого надо выбрать способ создания запроса;
2) указать системе, какие поля и из каких таблиц мы хотим включить в запрос;
3) указать тип запроса (по умолчанию установлен запрос на выборку);
4) при необходимости описать вычисляемые поля, то есть поля, значения которых являются функциями значений существующих полей;
5) описать групповые операции над записями исходных таблиц;
6) описать условия отбора, то есть сформулировать логическое выражение, которое позволит включить в выборку только записи, удовлетворяющие определенному условию.

При разработке конкретного запроса допускается любое сочетание перечисленных операций.

5.2. Конструктор запросов

Окно конструктора запроса - это основное средство работы с запросами. Оно позволяет не только сформировать новый запрос, но и понять, по какому принципу построен любой из уже существующих.



В верхней половине показаны связи между таблицами. Схему можно редактировать, добавляя в нее новые объекты: таблицы или запросы.

Нижняя часть окна - бланк запроса - содержит описание запроса в табличной форме. Каждая колонка в нем отвечает одному полю.

Строки *Поле* и *Имя таблицы* содержат списки, которые позволяют определить нужное поле.

Добавить нужные поля в бланк запроса можно перетаскиванием их имен из списка, находящегося в верхней части окна конструктора, в строку бланка *Поле.*

Строка *Сортировка* позволяет отсортировать полученные в результате запроса данные.

Строки *Условие отбора* предназначены для задания критериев отбора.

Если в строке *Условие отбора* добавить текст в квадратных скобках, то можно организовать параметрический запрос, т.е. запрос, который сначала требует ввести некоторые данные, по которым он будет строить запрос (например, Введите [Фамилию]).

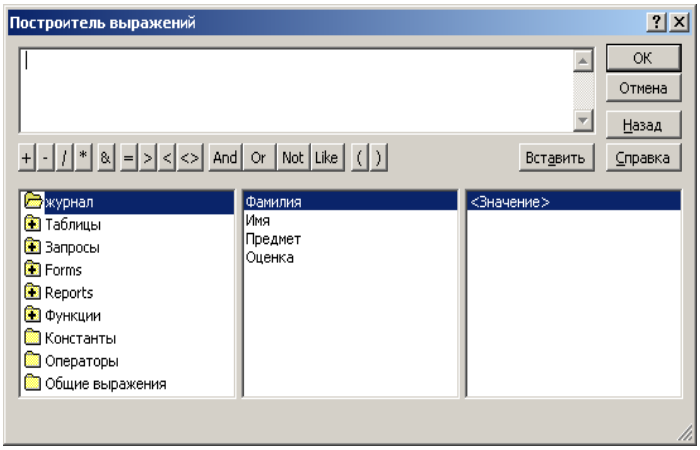
Строка *Групповая операция* содержит список функций для обработки и обобщение значений данного поля при помощи определенной функции. Данная строка появляется после нажатия кнопки с греческой буквой сигма (Σ), расположенной на панели инструментов, или выбора команды *Вид \ Групповые операции*.

В строке *Групповая операция* можно задать одну из следующих статистических функций:

Функция	*Выполняемая операция*
Sum	Суммирование значений определенного поля
Avg	Вычисление среднего значения
Min	Вычисление минимального значения
Max	Вычисление максимального значения
Count	Вычисление количества записей в определенном поле
First	Определяется первое значение в указанном поле
Last	Определяется последнее значение в указанном поле
StDev	Вычисляется стандартное отклонение значений данного поля
Var	Вычисляется вариация значений данного поля

5.3. Построение выражений с помощью построителя выражений

Microsoft Access предоставляет пользователю возможность создавать более сложные выражения с помощью построителя выражений.



Для запуска построителя выражений необходимо выполнить следующие действия:

1. Открыть запрос в режиме конструктора
2. Установить указатель в позицию, в которую требуется ввести выражение.
3. В контекстном меню выбрать команду *Построить.*
4. Построитель выражений состоит из трех разделов:

Поле выражения – окно, в котором создается выражение.

Кнопки операторов: раздел основных действий и операций, предназначенный для вставки элементов в поле выражения, а именно:

+, -, /, * - знаки арифметических действий;

& - сложение данных символьного типа; можно использовать знак «плюс»;

\ - деление целых частей делимого и делителя, результат округляется до целых;

^ - возведение в степень;

Mod - остаток от деления целых частей аргументов;

Like - создание масок при определении строк с неизвестными символами, в маске ? означает любой одиночный символ, * - любую последовательность символов, # - любую неизвестную цифру;

And, Or, Not - логические операторы, могут применяться к двум или нескольким выражениям и используются со скобками.

Элементы выражения расположены в трех полях:

- в левом выводятся папки, содержащие таблицы, запросы, формы, объекты базы данных, встроенные и определенные пользователем функции, константы, операторы и общие выражения;
- в среднем задаются определенные элементы или типы элементов для папки, заданной в левом поле. Например, если выбрать в левом поле **Встроенные функции,** то в среднем поле появится список всех типов функций Microsoft Access.
- в правом выводится список значений (если они существуют) для элементов, заданных левым и средним полями. ((Например, если выбрать в левом поле **Встроенные функции** и тип функции в среднем, то в правом поле будет выведен список всех встроенных функций выбранного типа.))

5.4. Корректирующие запросы

5.4.1. Запрос на обновление

Используя запрос на обновление, пользователь может изменить группу записей, отобранную на основе определенных критериев.

Для создания запроса на обновление надо:

1. Составить и выполнить запрос на выборку.
2. В режиме конструктора активизировать команду ***Запрос \ Обновление.***
3. Access добавит в бланк запроса строку ***Обновление,*** которая предназначена для указания новых значений полей таблицы. В качестве таковых могут выступать и вычисляемые выражения.
4. В специальном диалоговом окне Access укажет, сколько записей будет изменено в таблице, и потребует подтвердить выполнение этой операции.

5.4.2. Запрос на создание таблиц

На основе записей, отобранных запросом можно построить новую таблицу с помощью запроса на создание таблицы.

Для этого надо:

1. Подготовить запрос на выборку.
2. Выполнить составленный запрос для проверки его правильности.
3. В режиме конструктора выбрать команду ***Запрос \ Создание таблицы.***
4. Задать имя новой таблицы.
5. Сохранить и выполнить запрос.
6. В специальном окне Access потребует подтвердить выполнение операции.

5.4.3. Запрос на добавление

С помощью запроса на добавление записи одной таблицы (все или отобранные запросом) можно поместить в конец другой таблицы.

Для этого надо:

1. Составить и проверить запрос на выборку.
2. В режиме конструктора активизируйте команду ***Запрос \ Добавление.***
3. В появившемся окне в поле ***Имя таблицы*** задать имя таблицы, к которой надо присоединить отобранные данные.
4. После нажатия кнопки ***OK*** Access добавляет в бланк запроса строку ***Добавление.*** В эту строку автоматически или в ручную вставляются имена тех полей целевой таблицы, которые совпадают с именами полей запроса.
5. Выполнить запрос, подтвердив выполнение этой операции.

5.4.4. Запрос на удаление

Запросы этого типа служат для удаления из таблицы групп записей, соответствующих некоторому критерию отбора. Поскольку записи, удаленные посредством запроса, нельзя восстановить, следует тщательно анализировать критерии отбора.

Для создания запроса на удаление надо:

1. Составить и проверить запрос на выборку.
2. В режиме конструктора активизировать команду ***Запрос \ Удаление.***

3. Access добавит в бланк запроса строку ***Удаление*** и введет в ее ячейки значение *Условие.* Это означает, что пользователь может установить дополнительные критерии отбора.
4. Выполнить запрос и в специальном диалоговом окне подтвердить выполнение этой операции.

6. Создание форм

((Работа с данными в режиме таблицы имеет существенный недостаток: если полей слишком много, они не умещаются на экране и приходится прибегать к различным манипуляциям, чтобы оптимизировать представление: например, убирать некоторые столбцы, менять их положение.))

После создания БД можно создать формы для просмотра данных в более удобном виде. Форма служит средством защиты базы данных от неквалифицированных пользователей, а также ширмой, заслоняющей от любопытных глаз конфиденциальную информацию.

Любая форма строится на основе Access-таблицы или запроса. Имена полей извлекаются из спецификации таблицы, а поля в форме можно расположить по своему усмотрению. На основе одной таблицы можно построить несколько форм.

Создание формы производится в окне открытой БД путем выбора вкладки *Формы* и нажатия кнопки *Создать*

При этом появляется окно *Новая форма*, в котором можно выбрать один из 9 вариантов создания формы:

1. С помощью конструктора – *Конструктор*;
2. С помощью мастера – *Мастер*.
3. Автоматическое создание стандартной формы с различным расположением полей:

3.1. *Автоформа: В столбец* (поля размещены в один столбец);
3.2. *Автоформа: Табличная*;
3.3. *Автоформа: ленточная* (автоматическое создание стандартной формы, незначительно отличающиеся по виду от табличной формы);
3.4. *Автоформа: сводная таблица* (автоматическое создание формы в виде сводной таблицы с помощью Мастера);
3.5. *Автоформа: сводная диаграмма* (автоматическое создание формы в виде сводной диаграммы с помощью мастера);

4. Создание формы с диаграммой — *Диаграмма*;
5. Создание формы со сводной таблицей Microsoft Excel — *Сводная таблица*.

Построение стандартной формы осуществляется следующим образом:

1. В окне БД выбрать вкладку ***Формы*** и нажать кнопку ***Создать***.
2. В предложенном меню выбрать способ создания формы.
3. Выбрать имя таблицы или запроса, содержащих данные, на основе которых будет создана форма. При использовании мастера форм источник данных для формы следует указывать в диалоговом окне мастера.
4. Если были выбраны ***Мастер форм, Диаграмма*** или ***Сводная таблица,*** то при создании формы надо следовать инструкциям, выводимым в диалоговых окнах соответствующего мастера. При выборе элементов ***Автоформа: в столбец, Автоформа: ленточная*** **или** ***Автоформа: табличная*** форма создается автоматически.

   Изменить созданную форму можно в режиме конструктора.

6.1. Конструктор форм

В готовый проект формы можно внести небольшие изменения, чтобы сделать ее более привлекательной. Можно, например, добавить рисунки, поля или изменить расположение отдельных полей. В режиме конструктора можно также выбрать новый вид и/или размер шрифта, выровнять содержимое элементов формы, выбрать цвет текста и/или фона, определить ширину и/или цвет границы, выбрать специальные эффекты.

6.1.1. Элементы формы:

1. **Область данных.**

Основные объекты расположены в Области данных. Это — надписи полей и поля. Надпись и на-

звание поля могут не совпадать. Информация, расположенная в рамке надписи, не меняется при просмотре записей. Это аналог неизменяемой части карточки.

Поле предназначено для ввода данных. В окне конструктора оно представляет собой белый прямоугольник с одной или несколькими строками.

((Первоначально в режиме конструктора и для надписи, и для поля указано имя соответствующего поля. Можно отредактировать надпись. ! Название поля менять не рекомендуется, так как это может привести к ошибке.))

Примечание.

- Чтобы переместить поле на форме надо установить на него курсор таким образом, чтобы курсор принял вид раскрытой ладони, после чего, удерживая левую кнопку мыши, переместить в нужное место. Чтобы переместить одну из двух составных частей поля (например только название) надо выделить объект после чего навести курсор на большой черный квадрат в левом верхнем углу таким образом, чтобы курсор принял вид указательного пальца, и удерживая левую кнопку мыши - переместить.
- Чтобы изменить формат поля или полей надо его выделить (если выбирается более одного элемента, то удерживать при этом нажатой клавишу SHIFT), а затем на панели инструментов выбрать тип и размер шрифта, начертание, тип выравнивания.
- Чтобы изменить размеры поля, надо его выделить; вокруг поля появятся маркировочная рамка с маркерами изменения размеров (маленькие черные прямоугольники), а в левом углу рамки - большой черный квадрат (маркер перемещения); установить на нужный маркер указатель мыши (он примет вид двусторонней стрелки) и не отпуская кнопку мыши, растянуть до нужных размеров.
- Чтобы фотография полностью помещалась внутри рамки независимо от ее исходного размера, надо правой кнопкой мыши щелкнуть внутри рамки и в контекстном меню выбрать ***Свойства***; в открывшемся окне свойств рамки перейти на вкладку *Макет* и в строке *Установка размеров* установить переключатель *По размеру рамки***.**
- Для удобства позиционирования в поле формы выведена сетка. Программа позволяет автоматически выравнивать объекты по сетке. Соответствующие команды находятся в меню ***Формат.***
- Для создания в форме новых объектов - элементов управления - в режиме конструктора на экран выводится *Панель элементов*. Элементы управления недостаточно просто создать - их необходимо запрограммировать.

2. Панель элементов.

используется для добавления элементов управления (создание надписи, добавление поля, создание группы переключателей, добавление флажка, кнопки, рамки рисунка и т.д.).

Пиктограмма	*Название*	***Функция***
	Выбор объектов	Выделение объектов (позволяет маркировать и перемещать поля, а так же изменять их размеры, установленные по умолчанию).
	Мастера	Мастера создания элементов управления.
	Надпись	Вставка в форму названия нового поля.
	Поле	Отображает содержимое некоторого поля записи базы данных или вычисляемого поля.
	Группа переключателей	Создание и размещение группы, в которую можно ввести контрольные переключатели или селекторные кнопки.
	Выключатель	Создание выключателя, кнопки с фиксацией.
	Переключатель	Создание селекторного переключателя.
	Флажок	Создание контрольного переключателя.
	Поле со списком	Создание комбинированного списка.
	Список	Создание поля списка.
	Кнопка	Создание командной кнопки.
	Рисунок	Встраивание иллюстраций в форму.
	Свободная рамка объекта	Создание рамки объекта, для которого нельзя установить связь.

	Присоединенная рамка объекта	Создание рамки объекта, для которого будет объекта установлена связь с файлом-источником.
	Разрыв таблицы	Установка принудительного конца страницы формы.
	Набор вкладок	Создание формы или диалогового окна с несколькими вкладками.
	Подчиненная форма/отчет	Встраивание подчиненной формы в главную форму.
	Линия	Проведение в форме разделительной линии.
	Прямоугольник	Создание в форме прямоугольной рамки для группы полей.
	Дополнительные элементы	Встраивание в форму элементов, не представленных на панели инструментов.

3. **Список полей** содержит список всех полей в форме - источнике записей и используется для добавления полей в форму. Если список полей не отображается, то ***Вид \ Список полей.***
   Чтобы добавить поле в форму – надо, захватив ее левой кнопкой мыши в списке полей, переместить в область данных.

6.2. Диспетчер кнопочных форм

Когда форм и отчетов становится слишком много, полезно иметь систему указателей, которая позволит ориентироваться среди множества объектов.

В Access существует надстройка, которая позволяет создать своеобразный путеводитель по формам и отчетам базы данных - кнопочную форму.

Для создания кнопочной формы надо:

1. ***Сервис \ Служебные программы \ Диспетчер кнопочных форм*** ((в старых версиях *Сервис \ Надстройки: Диспетчер кнопочных форм.*))
2. Программа не найдет ни одной кнопочной формы в базе данных и выдаст соответствующее сообщение. Подтвердить, что надо создать новую кнопочную форму;
3. В окне ***Диспетчер кнопочных форм*** нажать кнопку ***Изменить.***
4. В появившемся окне ***Изменение страницы кнопочной формы*** определить элементы, щелкнув на кнопке ***Создать.***
5. Откроется окно ***Изменение элемента кнопочной формы.*** Здесь надо указать команду и объект базы данных - форму или отчет. Например, выбрать команду ***Открытие формы в режиме редактирования*** и объект - форма ***Сведения о студентах.***
6. Сохранить и закрыть форму.

Примечание. Чтобы форма запускалась автоматически при открытии базы данных, надо выбрать меню команду ***Сервис \ Параметры запуска*** и в списке ***Форма*** найти имя созданной формы.

6.3. Кнопочная форма

Кнопочную форму можно создать «с нуля». Для этого надо в окне БД выбрать вкладку ***Формы*** и кнопку *Конструктор.* Появится чистая форма, на которую можно добавлять различные элементы формы, в том числе кнопки, при нажатие на которые будет выполняться то или иное действие: загрузка таблицы, выполнение запроса, отчета или макроса и т.п.

Чтобы создать такую кнопку надо:

1. На панели элементов выбрать пиктограмму *Кнопка* и добавить ее на форму.
2. В появившемся окне выбрать Категорию и Действие.
   Например,
   Категория: *Работа с формой* Действия: *Открыть форму*;
   Категория: *Разное* Действия: *Выполнить макрос*;
   Категория: *Разное* Действия: *Выполнить запрос*;
   Категория: *Работа с отчетом* Действия: *Просмотр отчета*;
   Категория: *Работа с формой* Действия: *Закрыть форму*;
3. Выполнить дополнительные инструкции по каждой из категорий.
4. Задать текст, который должен располагаться на кнопке.
5. Выполнить действия по завершению создания кнопки.

Чтобы подкорректировать оформления окна, надо: щелкнуть правой кнопкой мыши в левой части окна формы (поле со стрелкой) и в контекстном меню выбрать *Свойства*; перейти на вкладку *Макет* и установить необходимые значения,
Например,
Область выделения: *Нет*
Кнопки перехода: *Нет*
Полосы прокрутки: *Отсутствуют*

6.4. Диаграмма в формах

В Access диаграммы создаются при помощи мастера диаграмм. Этот мастер используется в двух случаях:

- при создании новой формы или отчета «вручную»;
- при использовании существующих форм или отчетов в режиме конструктора.

Первый способ является наиболее простым и позволяет быстро создавать новые диаграммы. Когда диаграмма создается таким образом, Access представит форму или отчет с одной диаграммой.

1. В окне БД выбрать вкладку ***Форма*** и кнопку ***Создать.***
2. Выбрать элемент ***Диаграмма*** и источник данных.
3. В появившемся окне мастера диаграмм перенести поля, необходимые для создания диаграммы.
4. В следующем окне выбрать тип диаграммы.
5. В следующем диалоговом окне Access отображает выбранные поля в виде кнопок и показывает приблизительный вариант получающегося результата. Можно внести следующие изменения:
   - Захватив мышью поле, представленное в виде кнопки, перетащить его на соответствующую ось координат диаграммы.
   - Изменить вид вычислений: дважды щелкнуть на диаграмме числовое поле, после чего в окне *Вычисление итоговых значений* выбрать в списке одну из следующих операций:

Отсутствует	Не выполнять действий
Сумма	Суммирование данных
Среднее	Среднее
Минимум	Нахождение минимального значения
Максимум	Нахождение максимального значения
Число	Подсчет количества записей

6. В последнем окне мастера указать заголовок диаграммы.

Примечание. Чтобы изменить диаграмму (тип, параметры, ориентацию данных, а также добавить в нее новые или удалить ненужные элементы) надо перейти в режим конструктора и выполнить двойной щелчок на диаграмме.

((В режиме конструктора формы на диаграмме часто отображены данные из примеров или старые данные. Чтобы убедиться, что выведенные на экран данные являются текущими, необходимо переключиться в режим формы или в режим предварительного просмотра.))

7. Создание отчетов

Отчеты используются для отображения данных таблицы или запроса в удобном для пользователя формате (с заголовками и номерами страниц).

Больше всего сведений в отчете берется из базовой таблицы и запроса, являющихся источниками данных для отчета. Другие сведения вводятся при разработке отчета. При создании отчета можно использовать несколько таблиц и запросов.

Возможны следующие варианты создания отчета:

1. С помощью Конструктора - Конструктор;
2. С помощью Мастера — Мастер отчетов;
3. Автоматическое создание отчета стандартного вида, в котором каждая запись базового запроса или таблицы представлена в виде названия и значения поля — Автоотчет: в столбец;
4. Автоматическое создание стандартного отчета, в котором данные записи базового запроса или таблицы выводятся в одной строке — Автоотчет: ленточный;
5. Создание отчета с диаграммой — Диаграмма;
6. Создание отчета для печати почтовых наклеек — Почтовые наклейки.

((Проще всего создать отчет по вариантам 3 и 4, требующим наименьшее число параметров. В режиме Конструктора предоставляются более мощные средства, требующие больше знаний и времени для разработки отчета.

Методика работы с Конструктором отчетов мало чем отличается от работы с конструктором форм. В частности, при этом используется такая же панель элементов.))

Составные части отчета:

1. Область заголовка. Заголовок выводится один раз в начале отчета.
2. Область примечаний. Примечание выводится один раз в конце отчета.
3. Область колонтитулов. Верхний/нижний колонтитулы помещаются в начало каждой страницы отчета. Нижний колонтитул используется для вывода данных, таких как итоговые значения, даты или номер страницы. По умолчанию в нижний колонтитул добавляется поле с текущей датой, а также номер страницы и количество страниц.

   Примечание. Для добавления или удаления колонтитулов выбрать команду в меню *Вид* \ *Колонтитулы.* Чтобы скрыть один из колонтитулов установить для него в свойстве *Высота* значение 0.
4. Область данных. Содержимое области данных выводится один раз для каждой записи исходной таблицы или запроса.

Примечания.

- Если пользователь задал группировку записей отчета, то по каждому полю, по которому проводится группировка данных, Access формирует заголовок и примечание группы. Для создания в отчете области группировки нужно при открытом в режиме конструктора отчете выбрать пункт меню *Вид*/*Сортировка и группировка*.
- Существенное различие между отчетом и формой заключается в том, что отчеты предназначены исключительно для вывода данных на печать.
- Все элементы отчета представлены в режиме конструктора в качестве объектов. После того как объект выбран, он окружается рамкой с маркерами. Маркеры служат для изменения размеров объекта.

8. Создание макросов

Создание макроса производится в окне открытой БД путем выбора вкладки *Макросы* и нажатия кнопки *Создать* или *Вставка* \ *Макрос*.

В результате открывается окно создания макроса.

Оно включает четыре столбца:

1. *Имя макроса* – необязательный параметр. Следует задать, если окно содержит несколько макросов.
2. *Условие* - осуществляется ввод условия для выполнения той или иной макрокоманды макроса. Условие – логическое выражение или многоточие «…».

   Примечание. Если логическое выражение в строке макрокоманды истинно, то выполняется эта макрокоманда и все последующие, в поле *Условие* которых стоит многоточие. Если ложно, то пропускается текущая макрокоманда и все нижеследующие макрокоманды, содержащие в поле *Условие* многоточие.
3. *Макрокоманда*. Здесь перечисляются подлежащие выполнению действия в нужной последовательности.

Множество макрокоманд Access по функциональному принципу можно условно разделить на следующие группы:

- открытие и закрытие таблиц, форм и отчетов;
- вывод данных;
- выполнение запроса;
- проверка истинности условий и управление выполнением макрокоманд;
- установка значений;
- поиск данных;
- построение специального меню и выполнение команд меню;
- управление выводом на экран;
- сообщение пользователю о выполняемых действиях;
- переименование, копирование, удаление, импорт и экспорт объектов;
- запуск других приложений.

Ввод макрокоманд максимально облегчен, поскольку названия самих макрокоманд, а также значе-

ния многих аргументов можно не только вводить с клавиатуры, но и выбирать из списка. Выражения в области аргументов и условий выполнения макрокоманд можно вводить с клавиатуры или использовать *Построитель выражений*.

Выполнение каждой макрокоманды зависит от значений ее аргументов, которые вводятся в специально отведенные для этого поля, расположенные в нижней части окна макроса. Аргументы можно вводить с клавиатуры, однако лучше выбирать их из списка, чтобы не допустить ошибки при вводе.

Для добавления в макрос других макрокоманд нужно в области ввода макрокоманд перейти на следующую строку. Макрокоманды выполняются в порядке их расположения в бланке.

4. *Примечание.* Столбец *Примечание* содержит комментарий к программе и делает текст макроса понятнее.

Примечание. При создании нового макроса по умолчанию отображается только столбцы *Макрокоманда* и *Примечание.* Показ остальных столбцов устанавливается посредством опций *Имена макросов* и *Условия* из меню *Вид.*

8.1. Создание макроса

Существует несколько способов создания макросов. Рассмотрим один из них.

Задача. Создать макрос для открытия таблицы *Сотрудники*:

1. В окне БД на вкладке *Макросы* и выбрать кнопку *Создать*.
2. В появившемся окне в столбце таблицы *Макрокоманда* с помощью выпадающего меню выбрать *Открыть таблицу*.
3. В нижней части окна в поле *Имя таблицы* указать имя загружаемой таблицы – *Сотрудники*; остальные параметры оставить по умолчанию;
4. Закрыть и сохранить макрос под именем, например *Макрос1*.

8.2. Запуск макроса

Запустить макрос можно одним из следующих способов:

1. В окне БД базы на вкладке *Макросы* выделить имя макроса и нажать кнопку *Запуск* либо выполнить двойной щелчок на имени запускаемого макроса.
2. В режиме конструктора или в окне создания макросов надо щелкнуть на кнопке *Запуск* (на ней изображен восклицательный знак).
3. С помощью кнопки, размещенной на форме.

Задание. Создать кнопку для открытия таблицы *Сотрудники*.

1. Открыть форму в режиме конструктора.
2. На панели элементов выбрать пиктограмму *Кнопка* и добавить ее на форму.
3. В появившемся окне выбрать
   Категория: *Разное* Действия: *Выполнить макрос*;
4. На следующем шаге выбрать название макроса, например *Макрос1*.
5. На следующем шаге задать текст, который должен располагаться на кнопке.

9. Экспортирование таблиц

Пользователь может экспортировать данные из Access-таблиц в текстовые файлы, электронные таблицы, файлы других баз данных, а также в другую базу данных Access .

Для выполнения экспортирования надо:

1) Открыть БД, содержащую экспортируемую таблицу;
2) Выделить нужную таблицу.
3) *Файл \ Сохранить как экспорт*.
4) В появившемся окне выбрать нужный тип файла и имя файла.

10. Защита баз данных

Термины и определения.

Целью защиты информации является обеспечении безопасности хранимой и обрабатываемой информации, а также используемых программных средств.

Для эффективного построения системы защиты необходимо:

- Выделить уязвимые элементы вычислительной системы;
- Выявить угрозы для выделенных элементов;
- Сформировать требования к системе защиты;
- Выбрать методы и средства удовлетворения предъявляемым требованиям.

Безопасность вычислительной системы нарушается вследствие реализации одной или нескольких потенциальных угроз.

Под ***угрозой*** понимается возможность преднамеренного или случайного действия, которое может привести к нарушению безопасности хранимой и обрабатываемой информации.

Основные виды угроз:

В Access реализованы следующие способы защиты баз данных:

1. Парольная защита,
2. Защита на уровне пользователей
3. Шифрование.

10.1. Парольная защита БД

Парольная защита является простым и часто достаточным средством обеспечения защиты БД от открытия несанкционированными пользователем.

Установка пароля может быть запрещена в случае, если для БД установлена защита на уровне пользователя и наложен запрет на парольную защиту.

Парольная защита может использоваться в дополнение к защите на уровне пользователя. В этом случае сначала БД надо защитить на уровне пользователя и только потом устанавливают пароль. Но не наоборот.

Если парольная защита действует наряду с защитой на уровне пользователя, то пользователю предоставляется возможность выполнять над объектами БД действия, предусмотренные правами доступа.

Метод парольной защиты достаточно надежен, так как пароль Access шифрует, и к паролю нет прямого доступа. Он хранится вместе с защищаемой БД и поэтому открыть БД на другом компьютере не удастся.

((Увидеть пароль или найти место в файле БД, которое он занимает, дело не перспективное. При использовании парольной защиты от пользователя требуется подобрать удачный пароль и надежно его сохранить от потери и от хищения.))

Процедура установки парольной защиты БД включает следующие шаги:

1. Закрытие базы данных, если она открыта. Если база данных используются в сети, следует проверить, что все остальные пользователи тоже закрыли ее.
2. *Файл* \ *Открыть*.
3. В появившемся окне *Открытие файла базы данных* надо щелкнуть по стрелке на кнопке *Открыть* и выбрать режим монопольного доступа (*Монопольно*).
4. Сервис \ Защита \ Задать пароль БД.
5. Ввести пароль в поле Пароль с учетом регистра клавиатуры.
6. Подтвердить введенный пароль путем повторного его ввода в поле Подтверждение и нажатием кнопки ОК.

Удалить пароль намного проще, главное — знать его при открытии БД.

Для удаления пароля надо:

1. Открыть БД в режиме монопольного доступа.
2. Сервис \ Защита \ Удалить пароль БД. Команда доступна, если пароль базы данных уже установлен.
3. В поле *Пароль* появившегося окна ввести текущий пароль.
4. Нажать ОК.

((База данных по-прежнему остается открытой. При очередном ее открытии система Access запрашивать пароль не будет.))

Примечания.

1. Средств изменения пароля БД в Access нет, поэтому для изменения пароля следует удалить пароль, а затем определить новый.
2. Парольная защита не защищает БД от удаления.
3. Если пароль утерян, то доступ к защищенной БД не получить.
4. Защищая паролем, следует иметь в виду, что в случае связывания таблиц БД, защищенных паролем, могут быть проблемы. Пусть таблица БД (назовем ее подключаемой БД) связывается с таблицей защищенной паролем БД. При' образовании связи система Access потребует от пользователя ввода пароля защищенной БД. Введенный пароль системой в некотором месте запоминается. Если в последующем пароль защищенной БД изменится, то доступ к защищенным таблицам подключаемой БД будет невозможен. Система Access информирует об изменении пароля.

10.2. Защита на уровне пользователя

Защита на уровне пользователя применяется в случаях, когда с одной БД работают несколько пользователей или групп пользователей, имеющих разные права доступа к объектам БД.

((Использовать защиту на уровне пользователя можно на отдельном компьютере и при коллективной работе в составе локальной сети. Этот способ защиты подобен способам ограничения доступа в локальной сети)).

Установка защиты на уровне пользователя сложнее парольной защиты. Она предполагает создание файлов РГ, учетных записей пользователей и групп, включение пользователей в группы, активизацию процедуры подключения к Access, а также защиту каждой из БД с помощью Мастера защиты. Рассмотрим коротко эти операции.

Для организации защиты на уровне пользователя в системе Access создаются рабочие группы (РГ). Каждая рабочая группа определяет единую технологию работы совокупности пользователей. Система Access в произвольный момент времени может работать с одной РГ.

Информация о каждой РГ хранится в соответствующем файле (System.mdw), который автоматически создается при установке системы. Для управления рабочими группами в Access 2002 имеется программа «Администратор рабочих групп» (АРГ), запустить которую можно *Сервис \ Защита*.

Файл РГ описывает группы пользователей и отдельных пользователей входящих в эту РГ. Он содержит учетные записи групп пользователей и отдельных пользователей. По каждой учетной записи система Access хранит права доступа к объектам базы данных.

По умолчанию в каждую рабочую группу входит две группы пользователей: администраторы (имя группы Admins) и обычные пользователи (имя группы Users). Причем, в группу Admins первоначально включен один администратор под именем Admin.

При необходимости можно создать дополнительные группы пользователей, удалить или вставить новые учетные записи: *Сервис \ Защита \ Пользователи и группы*.

При создании групп указывается имя (идентификатор) группы и код, представляющий собой последовательность от 4 до 20 символов.

Каждому пользователю можно задать свой пароль, который хранится в учетной записи в файле РГ. Первоначально все пользователи РГ имеют пустой пароль или, можно считать, что не имеют пароля.

((При регистрации (создании) пользователей в системе защиты им присваивается имя, код и необязательный пароль. Имена групп и пользователей, их коды, а также пароли пользователей (если они заданы) Access скрывает от пользователей. Поэтому, если пользователь их забудет, найти их практически невозможно. Коды группы и пользователя используются для шифрования системой учетных записей в файле РГ.))

При создании рабочих групп и регистрации пользователей действуют следующие правила и ограничения:

1. Группы Admins и Users удалить невозможно.
2. В группе Admins должен быть хотя бы один пользователь. Первоначально таким пользователем является пользователь Admin (администратор).
   Удалить пользователя Admin из этой группы можно после включения в нее еще одного пользователя.
3. Все регистрируемые пользователи автоматически становятся членами группы Users. Удалить их из этой группы нельзя.

4. Один пользователь может входить в состав нескольких групп.
5. Удалить пользователя Admin из рабочей группы нельзя (из группы Admins его можно удалить, а из группы Users — нет).
6. Создаваемые группы не могут быть вложены в другие группы, другими словами, нельзя создавать иерархию групп пользователей.
7. В системе защиты могут быть пустые группы, но не может быть пользователей, не входящих ни в одну группу (они обязательно войдут в группу Users).
8. Каждой из групп приписываются определенные права на объекты БД. У группы Admins максимальные права
9. При подключении пользователя, зарегистрированного в нескольких группах, действуют минимальные из установленных в разных группах ограничения на доступ

Основное назначение системы защиты на уровне пользователя состоит в контроле прав доступа к объектам базы данных: *Сервис* \ *Защита* \ *Разрешения*.

Изменить права других пользователей могут следующие пользователи:

- члены группы Admins, определенной в файле РГ, использовавшемся при создании базы данных;
- владелец объекта;
- пользователь, получивший на объект права администратора.

Изменить свои собственные права в сторону их расширения могут пользователи, являющиеся членами группы Admins и владельцы объекта.

Владельцем объекта считается пользователь, создавший объект. Владельца объекта можно изменить путем передачи прав другому пользователю: *Сервис* \ *Защита* \ *Разрешения*.

Установка защиты на уровне пользователя выполняется с помощью Мастера защиты: *Сервис* \ *Защита* \ *Мастер*. Создание структуры рабочей группы надо завершить до вызова Мастера защиты. Результат защиты – создание новой защищенной БД, для объектов которой можно изменить права доступа других пользователей.

Снятие защиты:

в текущей рабочей группе

1. Запустить Access и подключиться к ней от имени администратора (группа Admins).
2. Открыть базы данных.
3. Предоставить группе «Users» полные прав на все объекты в БД.
4. Выйти из системы.

во всех рабочих группах включает все перечисленные действия, а также следующее:

5. Подключиться к системе с именем «Admin».
6. Создать пустую БД.
7. Импортировать в новую открытую БД все объекты из исходной БД: *Файл* \ *Внешние данные* \ *Импорт*.
8. Удалить исходную БД после проверки правильности снятия защиты.

Для организации защиты рекомендуется следующее:

1. Спланировать работу пользователей, объединив их в одну или несколько рабочих групп.
2. В каждой рабочей группе выделить группы пользователей.
3. Каждого пользователя зарегистрировать в системе защиты, присвоив имя и необязательный пароль.
4. Активизировать окно входа в систему Access, иначе вход в нее будет происходить от имени администратора Admin.
5. Защитить требуемые базы данных с помощью Мастера защиты.
6. Удалить каждую исходную БД, позаботившись о создании ее копии или копии защищенной БД.
7. Каждой группе (не рекомендуется, но можно, и каждому пользователю) при необходимости приписать полномочия по доступу к объектам каждой защищенной БД.

В последующем состав групп и права в группах можно менять.

10.3. Шифрование БД

Средства шифрования в Access позволяют кодировать файл БД таким образом, что она становится недоступной для чтения из других программ, в которых известен формат БД Access.

Шифрация в Access применяется, чтобы изменить до неузнаваемости стандартный формат файла БД. При выполнении процедуры шифрации/дешифрации БД не нужно задавать ключ шифрации (он формируется Access автоматически. Поэтому шифровать незащищенную паролем базу данных большого смысла нет, так как дешифровать БД может любой пользователь на любом компьютере, где установлен Access. Более того, пользователь может открыть и использовать зашифрованную БД, как и обычную незашифрованную.

Для ***шифрации/дешифрации*** базы данных надо:

1. Запустить Access. Для выполнения операций надо обладать правами владельца БД.
2. Сервис \ Защита \ Шифровать/дешифровать.
3. Указать имя БД.
4. Указать имя, диск и папку для целевой (зашифрованной) базы данных и нажать ОК. Имя рекомендуется задавать отличное от исходного).

((Так как если зашифрованной БД присваивается имя, совпадающее с исходной БД, то исходная БД «затирается» зашифрованной. И при ошибке дешифрации получится ситуация, когда и к новой БД доступа нет м старой уже нет. Поэтому рекомендуется шифруемой БД задать новое имя, затем проверить правильность операции шифрации путем пробной работы с зашифрованной БД. Если все нормально, исходную БД удалить, а зашифрованную БД — переименовать в исходную)).

Примечание. Для шифровки БД, защищенной паролем, надо в ответ на запрос Access ввести его.

Для обычной работы с зашифрованной БД ее не обязательно специально расшифровывать. Система «понимает» и зашифрованную информацию.

((Следует иметь в виду, что с зашифрованной базой данных Access работает несколько медленнее, поскольку операции шифрации/дешифрации выполняются в реальном масштабе времени)).

11. Скрытие объектов баз данных

Механизм скрытия объектов применяется в случаях, когда пользователь работает с базой данных через стандартный интерфейс — окно БД, и желательно предохранить базу данных от случайного доступа к ее объектам.

Скрываемые от пользователя объекты не удаляются, а становятся временно невидимыми. Скрывать от пользователя можно произвольные объекты различных типов: таблицы, формы, запросы, отчеты, макросы и модули.

Для ***скрытия текущего объекта*** БД надо:

1 способ. В окне свойств (вызывается из контекстного меню) этого объекта установить флажок атрибутов *Скрытый*.

((Поскольку в любой момент времени текущим может быть один объект, поэтому скрыть несколько объектов одновременно нельзя, это делается последовательно)).

2 способ. Переименовать его таким образом, чтобы его имя начиналось с символов «Usys» (в любом регисте и любом сочетании регистров: Usys, usys, USYS и т. д.).

Чтобы скрытые объекты сделать ***не скрытыми*** надо:

1. Сделать их видимыми: *Сервис \ Параметры: Вид* и в области *Отображение* на экране установить флажок *Скрытые объекты*, а для скрытых 2 способом еще и флажок *Системные объекты*.
2. Вызвать окно свойств каждого из объектов и сбросить флажок атрибута *Скрытый*.

12. Обслуживание баз данных

Основные способы обслуживания БД в Access:

1. Копирование. Копирование баз данных применяется для защиты их от случайной потери данных.
2. Восстановление. Применяется при повреждениях БД, не позволяющих пользователю нормально работать с БД или даже открыть ее.

   Причины повреждения: компьютерные вирусы; наличие дефектов (физических или логических) на диске; выключение пиния компьютера до предварительного закрытия сеанса работы с Access.

   ((Повреждение БД Access в большинстве случаев определяется при попытках пользователя открыть, сжать, зашифровать или дешифровать БД. В некоторых ситуациях сразу не удается определить, что база данных повреждена.))

   Если база данных ведет себя непредсказуемо, то, скорее всего, она требует своего восстановления.

3. Сжатие (компрессия).

   Сжатие БД средствами Access отличается от сжатия с помощью архиваторов и состоит в освобождении места на диске от удаленных из базы иных записей.

   ((Необходимость такого сжатия обусловлена следующим. При внесении пользователем изменений в БД файл БД только увеличивается. Занимаемая удаляемыми объектами и записями таблиц память не освобождается, а отмечается как неиспользуемая. При очередном добавлении объектов и записей снова выделяется память под эти объекты и размер файла БД увеличивается.

   Чтобы БД не была перегружена неиспользуемыми областями, ее надо периодически сжимать.))

   Для сжатия и восстановления данных надо:

   - Закрыть БД.
   - Создать резервную копию базы данных.
   - *Сервис* \ *Служебные программы* \ *Сжать и Восстановить*.
   - В диалоговом окне выбрать нужную БД и нажать кнопку *Сжать*.

     Для сжатия или восстановления БД, защищенной паролем сначала ввести пароль.
   - Результат сжатия можно поместить в ту же БД либо указать другое имя.
   - Ненужные файлы БД удалить.