Государственное профессиональное образовательное
учреждение
«Кемеровский педагогический колледж»

КУРС ЛЕКЦИЙ
«Базы данных»

для специальности
44.02.06 «Профессиональное обучение»
(по отрасли: 09.02.03 Программирование в компьютерных системах)

Кемерово, 2019

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Курс лекций по разделу МДК.04.01 Организация технологического процесса (по отраслям) «Базы данных» составлен в соответствии с федеральным государственным образовательным стандартом по специальности 09.02.03 Программирование в компьютерных системах.

Раздел «Базы данных» рассчитан на 128 часов, в том числе 90 часов лекций, 38 часов лабораторно-практических занятий и 64 часа самостоятельная работа. Раздел «Базы данных» изучается в 5 семестре. По окончании изучения данного раздела, учебным планом предусмотрен в 5 семестре экзамен.

Курс лекций включает в себя: основы теории баз данных; основные понятия и определения; модели данных: иерархическая, сетевая и реляционная; дальнейшее развитие способов организации данных; постреляционные модели данных; атрибуты и ключи; нормализация отношений; реляционная алгебра; проектирование баз данных; основные принципы проектирования; описание баз данных; логическая и физическая структура баз данных; обеспечение непротиворечивости и целостности данных; средства проектирования структур баз данных; системы управления базами данных (СУБД); классификация и сравнительная характеристика СУБД; базовые понятия СУБД; примеры организации баз данных; принципы и методы манипулирования данными (в том числе хранение, добавление, редактирование и удаление данных, навигация по набору данных; сортировка, поиск и фильтрация (выборка) данных); построение запросов к СУБД.

Курс лекций опирается на знания и умения, полученные при изучении дисциплин «Информатика и ИКТ в профессиональной деятельности», «Математика», «Информационные технологии», «Операционные системы» и в дальнейшем тесно связан с профессиональными дисциплинами.

Цель данного курса: изучение основных понятий баз данных, изучение теории проектирования баз данных, базовых моделей данных, изучение СУБД MS Access и методов разработки БД на ее основе.

По окончании изучения курса лекций студент должен

знать:

- основные положения теории баз данных, хранилищ данных, баз знаний;
- основные принципы построения концептуальной, логической и физической модели данных;
- современные инструментальные средства разработки схемы базы данных;
- методы описания схем баз данных в современных СУБД;

уметь:

- создавать объекты баз данных в современных СУБД и управлять доступом к этим объектам;
- работать с современными Case-средствами проектирования баз данных;
- формировать и настраивать схему базы данных.

Результатом освоения является овладение обучающимися вида профессиональной деятельности:

- разрабатывать объекты базы данных.
- реализовывать базу данных в конкретной СУБД.

Перечень принятых сокращений

БД – базы данных;
ИС - информационная система
СУБД - системы управления базами данных;
РСУБД - реляционная система управления базами данных;
ПК – персональный компьютер;
ПО – программное обеспечение;

РАЗДЕЛ 1. ОСНОВНЫЕ ПОЛОЖЕНИЯ ТЕОРИИ БАЗ ДАННЫХ, ХРАНИЛИЩ ДАННЫХ, БАЗ ЗНАНИЙ

1. Основные понятия и определения баз данных

Каждая информационная система в зависимости от ее назначения имеет дело с частью реального мира, которую принято называть предметной областью системы. Предметная область может относиться к любому типу организаций: банк, учебное заведение, магазин, предприятие и т.д.

Предметной областью *называют совокупность реальных объектов (сущностей), которые представляют интерес для пользователя.*

Объект (сущность) *– предмет, процесс или явление, о котором собирается информация, необходимая для решения задачи.*

Объектом может быть человек, предмет, событие. Каждый объект характеризуется рядом основных свойств, которые принято называть атрибутами.

Атрибутом *называется поименованная характеристика объекта.*

Атрибут показывает, какая информация должна быть собрана об объекте. Например, объект – студент КемПК, атрибуты – номер зачётной книжки, фамилия, имя, отчество, домашний адрес.

Банк данных *– система специальным образом организованных данных, включающих базы данных, программные технические, языковые средства, которые предназначены для обеспечения централизованного и коллективного использования данных.*

Определение банка данных предполагает, что с функционально-организационнной точки зрения банк данных является сложной человеко-машинной системой, включающей в себя все подсистемы, необходимые для надёжного, эффективного и продолжительного во времени функционирования.

В литературе понятие «база данных (БД)» трактуется по-разному, поэтому можно о том, что «число определений БД

сравнимо с числом существующих систем управления базами данных (СУБД)».

БД *– совокупность данных, организованных по определённым правилам, которые предусматривают общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.*

БД *- поименованная совокупность взаимосвязанных данных, находящихся под управлением СУБД.*

БД *– именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.*

Большинство БД включают аппаратные и программные средства ЭВМ, которые используются для хранения и манипулирования данными. Такие БД, как правило, опираются на определённую файловую систему, обеспечивающую выполнение простейших операций с данными.

Данные из одной БД взаимосвязаны и предназначены для одного или нескольких типов приложений и хранятся так, чтобы быть независимыми от использующих их программ.

Приложения *– программы, с помощью которых пользователи работают с БД.*

С одной БД могут работать множество различных приложений. Например, если БД моделирует некоторое предприятие, то для работы с ней может быть создано приложение, которое обслуживает подсистему учёта кадров, другое приложение может быть посвящено работе подсистемы расчёта з/п сотрудников, третье приложение работает как система складского учёта, четвёртое приложение посвящено планированию производственного процесса.

Общее управление БД осуществляется специально предназначенной для этого системой управления БД, состоящей из языковых, алгоритмических и программных средств.

СУБД *– совокупность языковых и программных средств, предназначенных для создания, ведения и использования информации, хранящейся в БД.*

СУБД является составной частью автоматизированного банка данных и обеспечивает работу прикладных программных средств с БД.

Главная цель СУБД – предоставить пользователю возможность оперировать данными в близких ему терминах и понятиях, не связанных с конкретными способами хранения данных в компьютере.

СУБД имеет набор средств, которые обеспечивают определённые способы доступа к данным. Наиболее общими операциями, которые выполняются СУБД, являются операции поиска, исправления, добавления, и удаления данных. Операция поиска является главной.

Функции СУБД:

1. *Описание структуры БД.* Подобными средствами являются язык описания данных, язык манипулирования данными и язык создания запросов (SQL) .
2. *Создание, обновление и извлечение информации из БД.* Средством извлечения информации из БД является язык обработки данных.
3. *Защита данных.* Использование системы разрешается лишь пользователям, имеющим на это право.
4. *Целостность данных.* При выполнении пользователем операций над данными поддерживается согласованность хранящихся данных.
5. *Независимость данных.* При использовании данных изменение одних не приводит к изменению других.
6. *Восстановление БД после сбоев.* В случае аппаратных или программных сбоев система должна возвращаться к некоторому согласованному состоянию данных.

Классификация СУБД может быть произведена по различным признакам, среди которых выделяют:

1. *По форме представления информации:* фактографические, документальные, мультимедийные, которые соответствуют цифровой, символьной и др. формам представления информации. К ним относят картографические, видео-, аудио-, графические и другие БД.

2. *По типу используемой модели данных:* иерархические, сетевые, реляционные.

3. *По типологии хранения данных:* локальные и распределённые (удалённые) БД.

4. *По типологии доступа и характеру использования:* специализированные и интегрированные.

5. *По функциональному назначению* (характеру решаемых задач): операционные и справочно-информационные.

6. *По сфере возможного применения:* универсальные и специализированные (или проблемно-ориентированные) системы.

7. *По степени доступности:* общедоступные и с ограниченным доступом пользователей.

Классификация не является полной. Различные источники предоставляют разнообразную классификацию.

ВОПРОСЫ:

1. Дайте определение предметной области, объекта и атрибута?
2. Что называют банком данных?
3. Сформулируйте определение базы данных?
4. Дайте определение СУБД?
5. Перечислите и расшифруйте функции СУБД?

2. Модели данных. Типы отношений.

2.1 Представление данных в БД. Логическая и физическая независимость данных.

Из определения БД и приведенных ранее основных требований следует, что данные могут использоваться (представлять) по-разному. С одной стороны разные прикладные задачи требуют разных наборов данных, обеспечивающих полноту информации, с другой стороны – они должны быть различными для различных категорий пользователей. Также должны быть различными и способы описания самих данных, их природы, формы хранения, условия взаимной непротиворечивости.

Выделяют три уровня представления данных: концептуальный, внутренний и внешний (Рисунок 1).

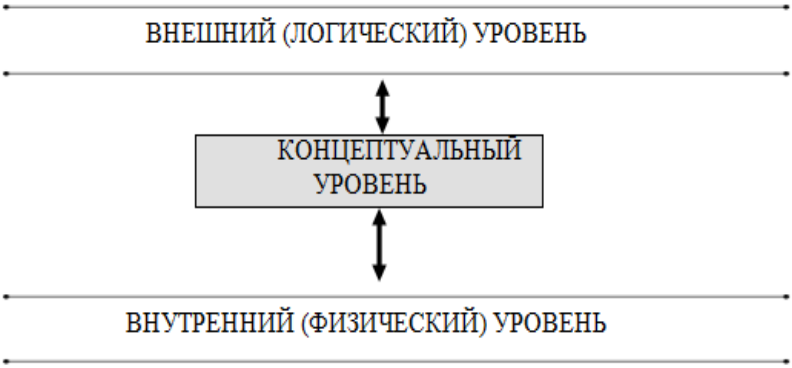


Рисунок 1 - Уровни представления БД

Эти уровни представлений введены исходя из различного рассмотрения БД.

1. ***Внешний уровень (логический уровень)*** – уровень представления данных конечного пользователя или прикладного программиста. Например, прикладному программисту требуются не все данные БД, а только некоторая их часть, используемая в его программе. Данный уровень обеспечивает именно эту форму представления данных. Внешний уровень также называют логическим уровнем, так как он не затрагивает физической организации (размещения) данных во внешней памяти. Этот уровень определяет точку зрения БД на отдельные приложения. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, именно эти сведения используются в подсистеме отдела кадров.

2. ***Концептуальный уровень*** – центральное управляющее звено, здесь БД представлена в наиболее общем виде. Представление на данном уровне представляет собой обобщённый взгляд на данные с позиции предметной области (разработчика приложений, пользователя или внешней информационной системы).

3. ***Внутренний уровень (физический уровень)*** – глобальное представление БД, определяет необходимые условия для организации хранения данных на внешних запоминающих устройствах. На этом уровне представления данные располагаются в файлах, с которыми взаимодействует СУБД.

Трёхуровневый подход позволяет обеспечить логическую (между 1 и 2) и физическую (между 2 и 3) независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных.

Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с одной базой данных.

Рассмотренная трехуровневая архитектура обеспечивает выполнение основных требований, предъявляемых к СУБД:

1. Адекватность отражения предметной области.
2. Возможность взаимодействия с БД различных пользователей при решении различных прикладных задач.
3. Обеспечение независимости программ и данных.
4. Надёжность функционирования БД и защита от несанкционированного доступа.

2.2 Понятие модели данных

Одно из важных мест взаимодействия пользователей с компьютером занимают языки запросов, используемые для извлечения информации из БД. Качество выполнения запросов пользователей зависит от структуры представления данных (модели данных), рассматриваемой предметной области.

В теории БД понятие модель данных является одним из фундаментальных. Модель – широкое понятие, включающее в себя множество способов представления изучаемой информации.

Модель данных – это совокупность функциональных характеристик объектов и особенностей представления информации.

Модель данных – это совокупность, трёх составляющих:

1) набора типов данных (являющихся блоками при построении БД);

2) набора операторов или правил вывода, предназначенных чтобы находить, выдавать или преобразовывать информацию, содержащуюся в любых частях структуры;

3) набор правил целостности, которые определяют множество непротиворечивых состояний базы данных.

В настоящее время предложено более 30 моделей данных. Некоторые из них: реляционные, сетевые, иерархические, модели данных «сущность-связь», бинарные модели, семантические сети, инфологические модели, патологические модели, физические.

2.3 Классические модели данных

Все БД могут быть разделены на три основных (классических) типа: иерархические, сетевые и реляционные. Назначение типа определяется тем, с помощью какой модели данных представлена информация.

Иерархическая модель данных. Её появление связано с тем, что в реальном мире многие связи соответствуют иерархии, когда один объект выступает как родительский, а с ним может быть связано множество подчинённых объектов (дочерних).

Основной структурой представления информации в иерархической модели данных является дерево. Дерево определяют как связный граф, не имеющий циклов. (Граф – это математическая конструкция, состоящая из вершин и рёбер) Все вершины разбиты на уровни. На самом высшем уровне находится только одна вершина, которая называется корнем дерева. Любой уровень можно достигнуть через корень дерева. Он соединяется рёбрами со всеми вершинами, находящимися на втором уровне, и только с ними. Вершины второго уровня соединяются с вершинами третьего уровня так, что каждая вершина третьего уровня соединяется только с одной вершиной второго уровня и т.д. Графическое изображение дерева имеет следующий вид (Рис. 2):

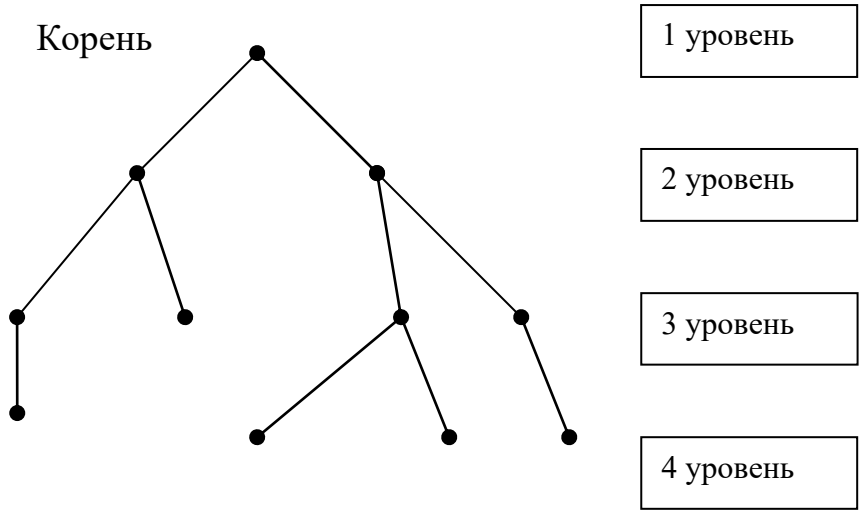


Рис. 2. Дерево

В дереве соотношение между верхними и нижними объектами имеет характер «один ко многим».

Положительные стороны использования иерархической модели данных:

- эффективное использование памяти компьютера,
- простота выполнения операций.

Сетевая модель данных. Её основой является сеть, т.е. произвольные графа, в вершинах, которых записана соответствующая информация, а рёбра соответствуют связям между ними. Каждое ребро соединяет две вершины. В сетевой структуре вершина может быть связана с любой другой. В сетевой структуре два уровня взаимосвязанных объектов: соотношение между ними – «многие ко многим».

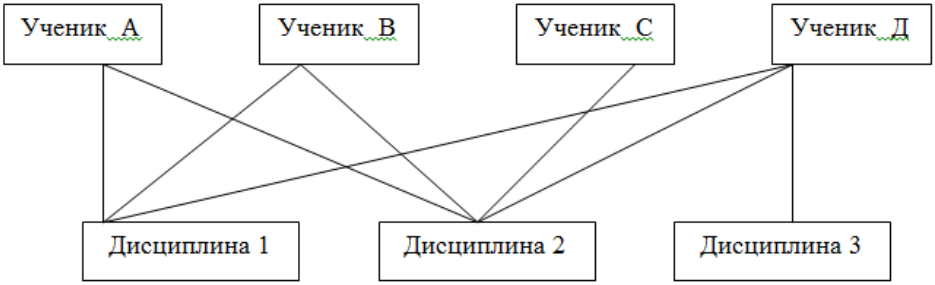


Рис. 3. Пример сетевой структуры

Примером сетевой модели может служить генеалогическое дерево человека.

Реляционная модель данных. Представляет собой совокупность данных, содержащихся в двухмерных таблицах, соединённых между собой отношениями. Любые данные можно преобразовать в простую таблицу. Такое представление является наиболее удобным и для пользователя, для и машины.

Название данной модели происходит от латинского слова relation, что в переводе означает отношение.

Каждая строка представляет собой совокупность строк и столбцов. Каждая строка описывает определённый объект. Каждый столбец содержит однородную информацию, т.е. элементы этого столбца имеют одинаковую природу (тип). Пример таблицы реляционной БД.

Ф И О студента	Дата рождения	Домашний адрес

В терминах БД строку называют **записью**, столбец – **полем**.

БД состоит из записей, а записи делятся на поля. Запись является наименьшей единицей обмена данными между оперативной и внешней памятью; поле – наименьшая единица обработки данных.

ВОПРОСЫ:

1. Каким образом представлены данные в БД?

2. В чём суть логической и физической независимости данных?
3. Что называют моделью данных?
4. Перечислите основные модели данных и дайте им характеристику?

3. Реляционная модель данных

Основные понятия реляционной базы данных.

Реляционная модель была предложена в 1969 году сотрудником фирмы IBM Е. Ф. Коддом, известным исследователем в области баз данных. Впервые основные концепции этой модели были опубликованы в 1970 году.

Набор средств для управления реляционными базами данных называется *реляционной системой управления базами данных* (РСУБД). Реляционная система управления базами данных может содержать утилиты, приложения, сервисы, библиотеки, средства создания приложений и другие компоненты.

Реляционная база данных представляет собой совокупность двумерных таблиц. Напомним, что любая таблица реляционной базы данных состоит из строк, называемых *записями,* и столбцов, называемых *полями.* Строки таблицы содержат сведения об объектах. Возьмем, например, записную книжку с заметками о друзьях и знакомых. О каждом в книжке есть информация - фамилия, имя, адрес, номер телефона, хобби. Для облегчения чтения расположим всю информацию в виде таблицы, то есть по строкам и столбцам. Каждый тип ин формации занесем в свой собственный столбец - отдельно имя человека, отдельно его адрес и т.д.

ФИО	Домашний адрес	Дата рождения	Телефон	Хобби
Андреев Олег Олегович	Ул. Зорге, 18-45	01.01.1980	5-46-28	Фотография
Иванов Иван Иванович	Уи. Тобольная, 12-45	23.03.1981		Рыбалка
Петров Петр Петрович	Ул. Красина, 45-1	08.08.1979	42-55-11	Филателия

Каждый столбец в таблице должен содержать только определенный тип информации (например, дату рождения, или хобби Вашего друга, или домашний адрес и т п.). Каждая *строка* таблицы содержит разнообразную (разного типа) информацию о человеке. Все данные, помещенные в одной строке, называют *записью,* каждый *элемент записи* - это *поле.* Таким образом, каждое поле содержит часть информации, находящейся на пересечении соответствующей строки и столбца. В таблице всевозможные значения одного типа в одном столбце называют *доменом.*

Поле является элементом записи и представляет собой ячейку таблицы. У каждого столбца есть свое неповторимое имя, описывающее тот вид информации, который содержится в нем. Это имя называют *«именем поля* базы данных»

Каждое поле имеет фиксированную длину, следовательно, и любая запись в таблице имеет фиксированную длину. Каждая запись характеризуется своим уникальным порядковым номером.

Данные в реляционной таблице должны удовлетворять следующим принципам:

1. Каждое значение поля должно быть атомарным, т.е. не расчленяемым на несколько значений;
2. Значения данных домена (в одном и том же столбце) должны принадлежать к одному и тому же типу данных, доступному для использования в данной СУБД;
3. Каждая запись в таблице уникальна, т.е. в таблице не существует двух записей с полностью совпадающим набором значений ее полей;
4. Каждое поле имеет уникальное имя;
5. Последовательность полей в таблице несущественна;
6. Последовательность записей в таблице несущественна.

Существенное отличие реляционной модели от обыкновенного последовательного файла заключается в том, что все столбцы в таблице с точки зрения входа предполагаются эквивалентными. Именно это свойство делает эту модель весьма мощной и делает невозможным отображение ее на память в виде последовательного массива данных.

Поскольку записи в таблице неупорядочены, то необходимо указать поле (или набор нескольких полей) для уникальной идентификации каждой записи.

Первичный ключ - это поле или набор полей, которые однозначно идентифицируют (определяют) запись таблицы

Обычно ключом является поле или совокупность полей фиксированной длины. Каждому значению *первичного* или *основного* ключа соответствует одна и только одна запись. Первичный ключ любой таблицы обязан содержать уникальные непустые значения для каждой записи. Если первичный ключ состоит из нескольких полей, он называется *составным первичным ключом* (primagy key).

Поле, указывающее на запись в другой таблице, связанную с данной записью, называется *внешним ключом* (foreign key).

Подобное взаимоотношение между таблицами называется *связью* (re1ationship). Связь между двумя таблицами устанавливается путем присвоения значений внешнего ключа одной таблицы значениям первичною ключа другой. Таблица, содержащая внешний ключ, называется *второстепенной* или, а таблица, содержащая первичный ключ, определяющий возможные значения внешнего ключа второстепенной таблицы, называется *главной*.

Типичная реляционная база данных состоит из нескольких связанных таблиц. Приведем пример фрагмента базы данных "Деканат".

Таблица **Студенты** состоит из следующих полей:

Номер зачётной книжки	Фамилия, Имя, Отчество	Группа	Дата рождения	Домашний адрес	Телефон

РК

Таблица **Успеваемость** состоит из следующих полей:

Код отметки	Номер зачетной	Код дисциплины	Семестр	Форма сдачи	Отметка	Дата сдачи	Код преподавателя
	РК	***FK***	***FK***				

Таблица **Дисциплины** состоит из следующих полей:

Код дисциплины	Название	Программа	Специальность
РК			

Таблица **Преподаватели** состоит из следующих полей:

Код преподавателя	ФИО	Код кафедры	Должность	Ученая степень	Ученое звание
РК					

Таблица **Кафедры** состоит из следующих полей

Код кафедры	Название	Код преподавателя (зав. кафедрой)
РК		***FK***

Номер зачетной книжки у каждого студента индивидуален. Таким образом поле "Номер зачетной книжки" в таблице "Студенты" однозначно определяет студента, следовательно, это поле является ключевым полем таблицы "Студенты". Если требуется узнать успеваемость студента, следует найти значение идентификатора студента в поле "Номер зачётной книжки" таблицы "Успеваемость" и в найденных строках прочесть значение поля "Отметка". Иными словами, нужно связать две таблицы "Студенты и "Успеваемость" по полю "Номер зачетной книжки". Внешним ключом таблицы "Успеваемость" является поле "Номер зачетной книжки".

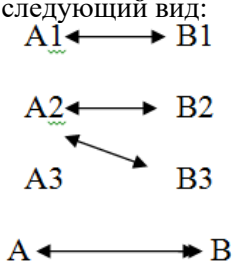
Типы связей между таблицами.

Все информационные объекты предметной области связаны между собой. Соответствия, отношения, возникающие

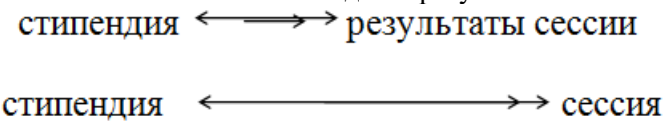
между объектами предметной области, называются *связями.* Связанные отношениями таблицы взаимодействуют по принципу *главная, подчиненная*. Возможны следующие отношения между таблицами:

1) Отношение «один – ко – многим»: одной записи из главной таблицы может соответствовать ноль, один или несколько записей подчинённой таблицы.

При связи «один – ко – многим» (обозначают 1:M) одному экземпляру информационного объекта A соответствует ноль, один или более экземпляров объекта B, но каждый экземпляр объекта B связан не более чем с 1 экземпляром объекта A. Графически данное соотношение имеет следующий вид:

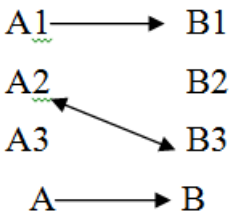


Пример. Установленный размер стипендии по результатам сессии может повторяться многократно для различных студентов, поэтому примером связи 1:M служит связь между информационными объектами стипендия и результаты сессии:



2) Отношение «один – к - одному»: одной записи из главной таблицы соответствует только одна запись из подчинённой таблицы.

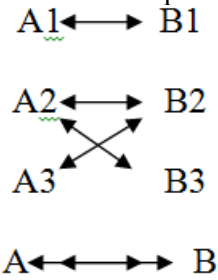
Связь «один – к – одному» (обозначают 1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует не более одного экземпляра информационного объекта B и наоборот.



Пример. Каждый студент за период обучения может выполнить только одну дипломную работу, поэтому примером связи 1:1 может служить связь между информационными объектами студент и дипломная работа:

студент ⟷ дипломная

3) Отношение «многие – ко - многим». Связь «многие – ко – многим» (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0,1 или более экземпляров объекта B и наоборот.



Пример. Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов, поэтому примером связи M:M может служить связь между информационными объектами студент и преподаватель:

студент ⟷ преподаватель

4) Отношение «многие – к - одному».

Группа связанных между собой таблиц называется *схемой базы данных.* Информацию о таблицах, их колонках (имена, тип данных, длина поля), первичных и внешних ключах, а также иных объектах базы данных, называют *метаданными.*

Ссылочная целостность

Одним из правил ссылочная целостность (referential integrity) является то, что первичный ключ любой таблицы должен содержать уникальные непустые значения для данной таблицы. Некоторые СУБД могут контролировать уникальность первичных ключей. Если СУБД контролирует уникальность первичных ключей, то при попытке присвоить первичному ключу значение, уже имеющееся в другой записи, СУБД сгенерирует диагностическое сообщение, обычно содержащее словосочетания primary key violation. Это сообщение в дальнейшем может быть передано в приложение, с помощью которого конечный пользователь манипулирует данными.

Если две таблицы связаны соотношением главная-подчиненная, внешний ключ подчинённой таблицы должен содержать только те значения, которые имеются среди значений первичного ключа главной задачи. Если корректность значений внешних ключей не контролируется СУБД, можно говорить о нарушении ссылочной целостности. В этом случае, если мы удалим из таблицы «Студенты» запись, имеющую хотя бы одну связанную с ней подчинённую запись в таблице «Успеваемость», то в таблице «Успеваемость» окажутся записи об успеваемости студентов, сданных неизвестно кем. Если же СУБД контролирует корректность значений внешних ключей, то при попытке присвоить внешнему ключу значение, отсутствующее среди значений первичных ключей главной таблицы, либо при удалении или модификации записей главной таблицы, приводящих к нарушению ссылочной целостности, СУБД сгенерирует сообщение, о котором говорилось выше.

ВОПРОСЫ:

1. Что такое реляционная база данных?
2. Что такое запись и поле?
3. Что такое первичный ключ?
4. Что такое внешний ключ?
5. Что называют связью?
6. Расскажите о видах связей между таблицами?
7. В чём суть ссылочной целостности?

4. Основы реляционной алгебры

Среди многих попыток представить обработку данных на формальном абстрактном уровне реляционная модель, предложенная Э. Ф. Коддом, стала по существу первой работоспособной *моделью данных,* поскольку помимо средств описания объектов имела эффективный инструментарий преобразований этих описаний — операции реляционной алгебры.

Реляционная алгебра — это теоретический язык операций, позволяющих создавать на основе одного или нескольких отношений другое отношение без изменения самих исходных отношений.

Таким образом, оба операнда и результат являются отношениями, поэтому результаты одной операции могут применяться в другой операции. Это позволяет создавать вложенные выражения реляционной алгебры (по аналогии с тем, как создаются вложенные арифметические выражения), но при любой глубине вложенности результатом является отношение. Такое свойство называется ***замкнутостью***. Оно подчеркивает то, что применение любого количества операций реляционной алгебры к отношениям не приводит к созданию иных объектов, кроме отношений, точно так же, как результатами арифметических операций с числами являются только числа.

Реляционная алгебра является языком последовательного использования отношений, в котором все кортежи, возможно, даже взятые из разных отношений, обрабатываются одной командой без организации циклов.

Существует несколько вариантов выбора операций, которые включаются в реляционную алгебру. Первоначально Кодд предложил восемь операций, но впоследствии к ним были добавлены и некоторые другие. Пять основных операций реляционной алгебры, а именно **выборка** (*selection*), **проекция** (*projection*), **декартово произведение** (*cartesian product*), **объединение** (*union*) и **разность множеств** (*set difference*), выполняют большинство действий по извлечению данных, которые могут представлять интерес. На основании пяти основных операций можно также вынести дополнительные операции, такие

как **операции соединения** (*join*), **пересечения** (*intersection*) и **деления** (*division*), которые могут быть выражены в терминах пяти основных операций.

Операции выборки и проекции являются **унарными**, поскольку они работают с одним отношением. Другие операции работают с парами отношений, и поэтому их называют **бинарными** операциями. В приведенных ниже определениях R и S — это два отношения, определенные на атрибутах

$A=(a_1, a_2, \dots a_N)$ и $B=(b_1, b_2, \dots b_M)$ соответственно.

1) Унарные операции:

- *Выборка (или ограничение)*

Операция выборки применяется к одному отношению R и определяет результирующее отношение, которое содержит только те кортежи (строки) из отношения R, которые удовлетворяют заданному условию (предикату).

- *Проекция*

Операция проекции применяется к одному отношению R и определяет новое отношение, содержащее вертикальное подмножество отношения R, создаваемое посредством извлечения значений указанных атрибутов и исключения из результата строк-дубликатов.

Пример 1. Пусть дано отношение с информацией о сотрудниках:

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Результат выборки **Зарплата < 3000** будет иметь вид:

Табельный номер	**Фамилия**	**Зарплата**

1	Иванов	1000
2	Петров	2000

Пример 2. Пусть дано отношение с информацией о поставщиках, включающих наименование и месторасположение:

Номер поставщика	**Наименование поставщика**	**Город поставщика**
1	Иванов	Уфа
2	Петров	Москва
3	Сидоров	Москва
4	Сидоров	Челябинск

Проекция **R [Город поставщика]** будет иметь вид:

Город поставщика
Уфа
Москва
Челябинск

2) Бинарные операции:

- *Объединение*

Объединение двух отношений R и S определяет новое отношение, которое включает все кортежи, содержащиеся только в R, только в S, одновременно в R и S, причем все дубликаты кортежей исключены. При этом отношения R и S должны быть совместимыми по объединению.

Если R и S включают, соответственно, I и J кортежей, то объединение этих отношений можно получить, собрав все кортежи в одно отношение, которое может содержать не более (I + J) кортежей. Объединение возможно, только если схемы двух

отношений совпадают, т.е. состоят из одинакового количества атрибутов, причем каждая пара соответствующих атрибутов имеет одинаковый домен. Отметим, что в определении совместимости по объединению не указано, что атрибуты должны иметь одинаковые имена. В некоторых случаях для получения двух совместимых по объединению отношений может быть использована операция проекции.

Пример 3. Пусть даны два отношения R и S с информацией о сотрудниках:

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 1 Отношение R

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

Таблица 2 Отношение S

Объединение отношений R и S будет иметь вид:

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

2	Пушников	2500
4	Сидоров	3000

Таблица 3 Отношение R UNION S

- *Разность*

Разность двух отношений R и S состоит из кортежей, которые имеются в отношении R, но отсутствуют отношении S. Причем отношения R и S должны быть совместимыми по объединению.

Пример 4. Для тех же отношений R и S , что и в предыдущем примере вычитание имеет вид:

Табельный номер	**Фамилия**	**Зарплата**
2	Петров	2000
3	Сидоров	3000

Таблица 4 Отношение A MINUS B

- *Пересечение*

Операция пересечения определяет отношение, которое содержит кортежи, присутствующие как в отношении R, так и в отношении S. Отношения R и S должны быть совместимыми по объединению.

Пример 5. Для тех же отношений R и S, пересечение имеет вид:

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000

Таблица 5 Отношение A INTERSECT B

- *Декартово произведение*

Операция декартово произведение определяет новое отношение, которое является результатом конкатенации (т.е. сцепления) каждого кортежа из отношения R с каждым кортежем из отношения S.

Операция декартово произведение применяется для умножения двух отношений. Умножением двух отношений называется создание другого отношения, состоящего из всех возможных пар кортежей обоих отношений. Следовательно, если одно отношение имеет I кортежей и N атрибутов, а другое — I кортежей и M атрибутов, то их декартово произведение будет содержать (I x J) кортежей и (Л* + M) атрибутов. Исходные отношения могут содержать атрибуты с одинаковыми именами. В таком случае имена атрибутов будут содержать названия отношений в виде префиксов для обеспечения уникальности имен атрибутов в отношении, полученном как результат выполнения операции декартова произведения.

Пример 6. Пусть даны два отношения R и S с информацией о поставщиках и деталях:

Номер поставщика	**Наименование поставщика**
1	Иванов
2	Петров
3	Сидоров

Таблица 6 Отношение R (Поставщики)

Номер детали	**Наименование детали**
1	Болт
2	Гайка
3	Винт

Таблица 7 Отношение S (Детали)

Декартово произведение отношений R и S будет иметь вид:

Номер	**Наименование**	**Номер**	**Наименование**

поставщика	поставщика	детали	детали
1	Иванов	1	Болт
1	Иванов	2	Гайка
1	Иванов	3	Винт
2	Петров	1	Болт
2	Петров	2	Гайка
2	Петров	3	Винт
3	Сидоров	1	Болт
3	Сидоров	2	Гайка
3	Сидоров	3	Винт

Таблица 8 Отношение R TIMES S

Результат выполнения любой операции над таблицами также является таблицей, поэтому результат одной операции может использоваться в качестве исходных данных для другой. Другими словами, можно записывать вложенные реляционные выражения, т. е. выражения, в которых операторы сами представлены реляционными выражениями, причем произвольной сложности. Эта особенность называется свойством *реляционной замкнутости.*

ВОПРОСЫ:

8. Перечислите и охарактеризуйте унарные операции реляционной алгебры? Приведите примеры.
9. Перечислите и охарактеризуйте бинарные операции реляционной алгебры? Приведите примеры.

5. Основные понятия удаленных баз данных.

При размещении базы данных на ПК, который не находится в сети, БД всегда используется в монопольном режиме. Даже если БД используют несколько пользователей, они могут

работать с БД только последовательно. Однако работа на изолированном ПК с небольшой БД в настоящий момент становится уже не характерной для большинства приложений.

БД отражает информационную модель реальной предметной области, она растет по объему следовательно резко увеличивается количество задач, решаемых с помощью этой БД и в соответствии с этим увеличивается количество приложений, работающих с единой БД. ПК объединяются в локальные сети и необходимость распределения приложений, работающих с единой БД по сети, является несомненной.

Параллельный доступ к одной БД нескольких пользователей, в том случае, если БД расположена на одной машине, соответствует режиму распределенного доступа к центральной БД. Такие системы называются **системами распределенной обработки данных.**

Если же БД расположена на нескольких ПК, распределенных в сети, и к ней возможен параллельный доступ нескольких пользователей, то мы имеем дело с параллельным доступом к распределенным БД. Такие **системы называются системами распределенных (удаленных) баз данных.**

Режимы работы с базой данных.

Терминология удаленных баз данных.

Пользователь БД – это программа или человек, обращающиеся к БД на языке манипулирования данными.

Запрос – это процесс обращения пользователя к БД с целью ввода, получения или изменения информации в БД.

Транзакция – это последовательность операций модификации данных в БД, переводящая БД из одного непротиворечивого состояния в другое непротиворечивое состояние.

Логическая структура БД – это определение БД на физически независимом уровне, ближе всего соответствующем концептуальной модели БД.

Топология БД – это схема распределения физических БД по сети.

Локальная автономность означает принадлежность локальному владельцу информации локальной БД и связанных с ней определенных данных.

Удаленный запрос – это запрос, который выполняется с использованием модемной связи.

Возможность реализации удаленной транзакции – это обработка одной транзакции, состоящей из множества SQL-запросов, на одном удаленном узле.

Поддержка распределенной транзакции допускает обработку транзакции, состоящей из нескольких SQL-запросов, которые выполняются на нескольких узлах сети (удаленных или локальных), но каждый запрос в этом случае обрабатывается только на одном узле, т.е. запросы не являются распределенными.

Распределенный запрос – запрос, при обработке которого используются данные из БД, расположенные в разных узлах сети.

ВОПРОСЫ:

1. Дайте определения следующих понятий:
- топология БД, или структура распределенной БД;
- локальная автономность;
- удаленный запрос;
- поддержка распределенной транзакции.

6. Архитектуры баз данных.

Понятие базы данных изначально предполагало возможность решения многих задач несколькими пользователями. В связи с этим, важнейшей характеристикой современных СУБД является наличие многопользовательской технологии работы.

Разная реализация таких технологий в разное время была связана как с основными свойствами вычислительной техники, так и с развитием программного обеспечения.

Централизованная архитектура

При использовании этой технологии база данных, СУБД и прикладная программа (приложение) располагаются на одном компьютере (рисунок 1). Для такого способа организации не требуется поддержки сети и все сводится к автономной работе.

Рисунок 1 - Централизованная архитектура

Многопользовательская технология работы обеспечивалась либо режимом мультипрограммирования, либо режимом разделения времени.

Архитектура "файл-сервер"

Увеличение сложности задач, появление персональных компьютеров и локальных вычислительных сетей явились предпосылками появления новой архитектуры *файл-сервер*. Эта архитектура баз данных с сетевым доступом предполагает назначение одного из компьютеров сети в качестве выделенного сервера, на котором будут храниться файлы базы данных. В соответствии с запросами пользователей файлы с *файл-сервера* передаются на рабочие станции пользователей, где и осуществляется основная часть обработки данных. Центральный сервер выполняет в основном только роль хранилища файлов, не участвуя в обработке самих данных (рисунок 2).

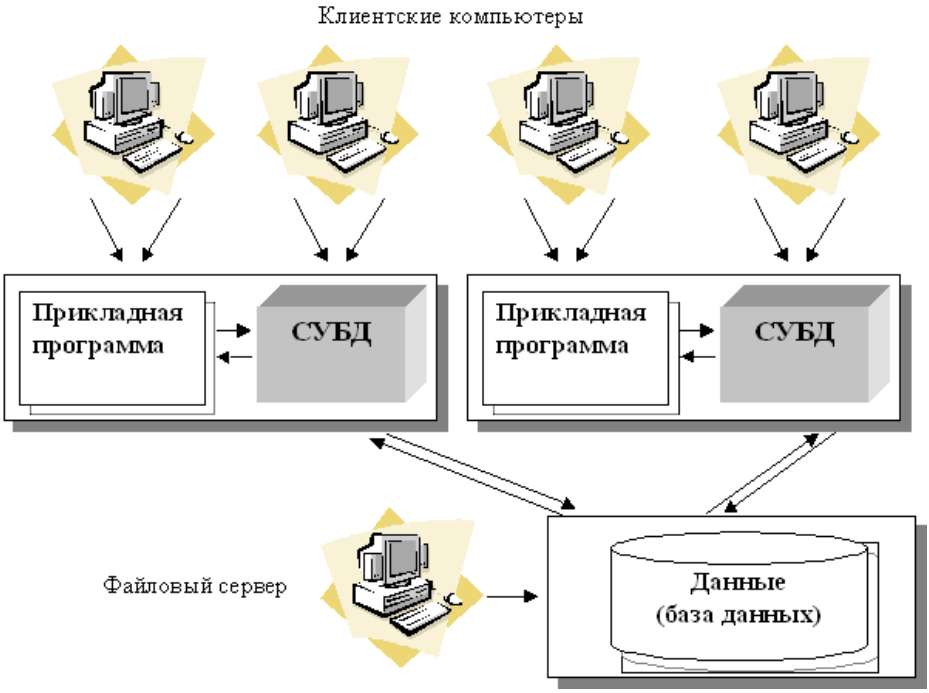


Рисунок 2 - Архитектура "файл-сервер"

Технология "клиент – сервер"

Использование технологии "*клиент – сервер*" предполагает наличие некоторого количества компьютеров, объединенных в сеть, один из которых выполняет особые управляющие функции (является сервером сети).

Так, архитектура "*клиент – сервер*" разделяет функции приложения пользователя (называемого клиентом) и сервера. Приложение-клиент формирует запрос к серверу, на котором расположена БД, на структурном языке запросов SQL (Structured Query Language), являющемся промышленным стандартом в мире реляционных БД. Удаленный сервер принимает запрос и переадресует его SQL-серверу БД.

SQL-сервер – специальная программа, управляющая удаленной базой данных. SQL-сервер обеспечивает интерпретацию запроса, его выполнение в базе данных, формирование результата выполнения запроса и выдачу его приложению-клиенту. При этом ресурсы клиентского компьютера не участвуют в физическом выполнении запроса; клиентский компьютер лишь отсылает запрос к серверной БД и получает результат, после чего интерпретирует его необходимым образом и представляет пользователю. Так как клиентскому приложению посылается результат выполнения запроса, по сети "путешествуют" только те данные, которые необходимы клиенту. В итоге снижается нагрузка на сеть. Поскольку выполнение запроса происходит там же, где хранятся данные (на сервере), нет необходимости в пересылке больших пакетов данных. Кроме того, SQL-сервер, если это возможно, оптимизирует полученный запрос таким образом, чтобы он был выполнен в минимальное время с наименьшими накладными расходами.

Архитектура системы представлена на рисунке 3.

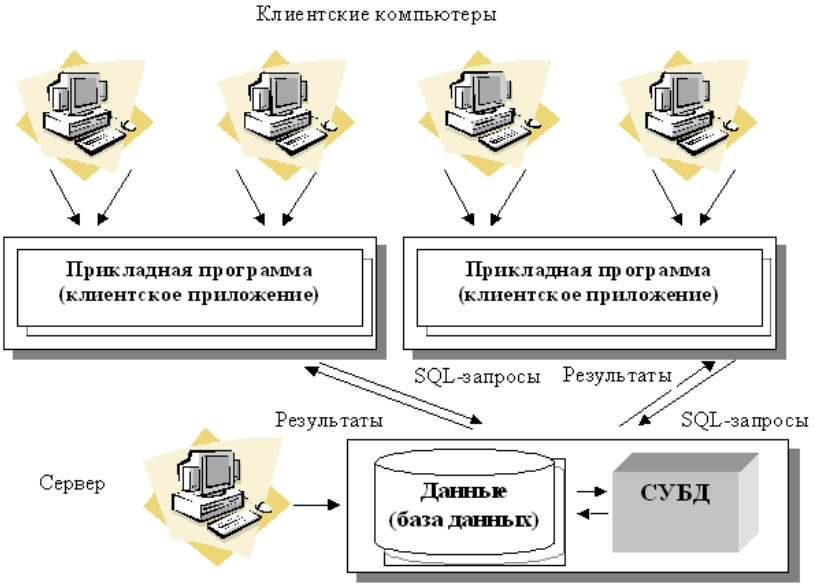


Рисунок 3 - Архитектура "клиент – сервер"

Трехзвенная (многозвенная) архитектура "клиент – сервер"

Трехзвенная (в некоторых случаях *многозвенная*) **архитектура** (N-tier или multi-tier). представляет собой дальнейшее совершенствование технологии " *клиент – сервер* ". Рассмотрев архитектуру " *клиент – сервер* ", можно заключить, что она является 2-звенной: первое звено – клиентское приложение, второе звено – сервер БД + сама БД. В *трехзвенной архитектуре* вся бизнес-логика (деловая логика), ранее входившая в клиентские приложения, выделяется в отдельное звено, называемое сервером приложений. При этом клиентским приложениям остается лишь пользовательский интерфейс.

Схематически такую *архитектуру* можно представить, как показано на рисунке 4.

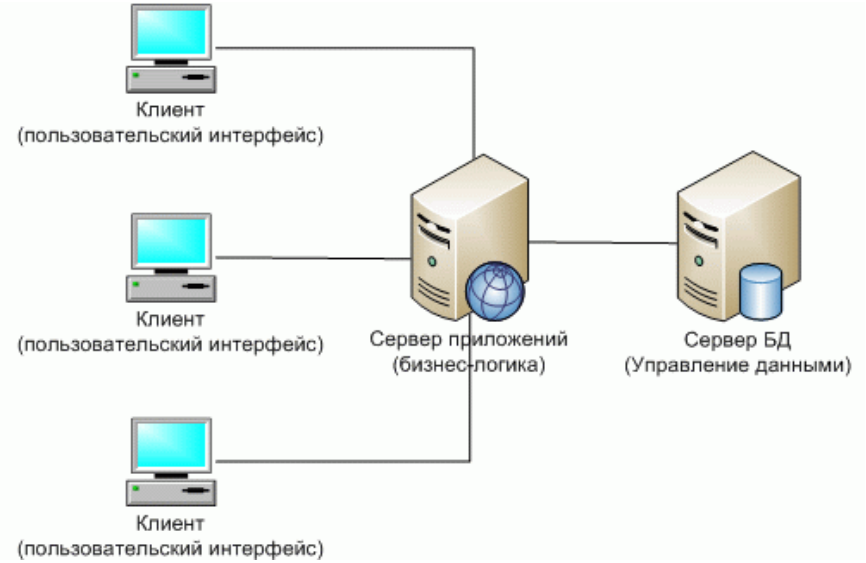


Рисунок 4 - Представление многоуровневой архитектуры "клиент-сервер"

ВОПРОСЫ:

1. Назовите достоинства и недостатки существующих многопользовательских технологий с базами данных.

РАЗДЕЛ 2. ОСНОВНЫЕ ПРИНЦИПЫ ПОСТРОЕНИЯ КОНЦЕПТУАЛЬНОЙ, ЛОГИЧЕСКОЙ И ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ

1. Типы информационных моделей. Концептуальные модели данных. Логические модели данных. Физические модели данных.

При создании баз данных рассматривают два вида информационных моделей: информационная модель предприятия и информационная модель данных.

Информационная модель предприятия строится на втором этапе проектирования базы данных. Здесь определяются структурные подразделения фирмы, которые используют информацию из базы данных, и направление движения потоков информации между структурными подразделениями фирмы.

Информационная модель данных (рисунок 5) имеет более сложную структуру. Здесь отображаются:

- источники возникновения информации,
- структурные подразделения фирмы, которые создают или используют информацию,
- переходы от одного типа модели к другому,
- подразделения - потребители информации.

Подразделение 1,... Подразделение N — структурные подразделения фирмы. В левой части рисунка эти подразделения выступают как источники концептуальных требований, то есть предоставляют сведения для создания базы данных. В верхней части рисунка эти же подразделения выступают как потребители информации и источники данных для базы данных.

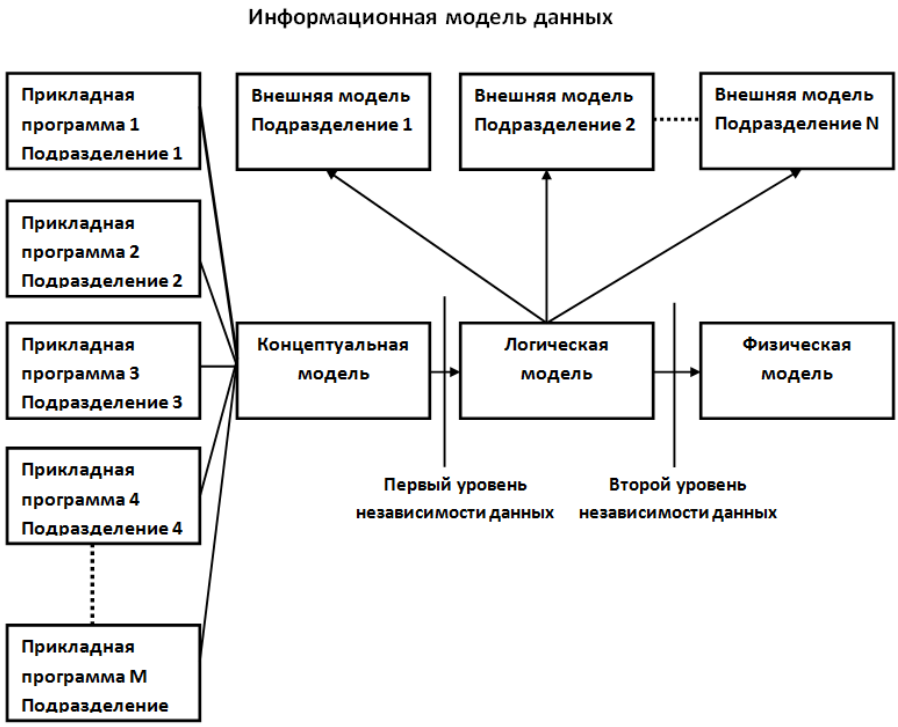


Рисунок 5 - Информационная модель данных

Концептуальная модель данных — это совокупность концептуальных требований, выдвинутых сотрудниками структурных подразделений фирмы. В результате отображения концептуальной модели на СУБД будет получена *логическая модель данных.*

В процессе отображения концептуальной модели подбирается такая СУБД, которая в полной мере может удовлетворить требования заказчика. Если по каким-либо причинам выполнить требования заказчика не удается, то разработчик должен предоставить заказчику убедительные аргументы и убедить заказчика снизить концептуальные требования. Процесс согласования концептуальных требований трудоемкая и длительная процедура. После построения логической

модели необходимо составить письменный протокол, в котором перечислить все концептуальные требования и операции по обработке информации в базе данных.

При переходе от концептуальной к логической модели необходимо обеспечить следующее условие: внешние модели никак не связаны с типом физической памяти, где хранятся данные базы данных, и методами обработки этих данных. Это условие называется *первым уровнем независимости данных*. Далее общая логическая модель данных разделяется на некоторые составные части, которые называют внешними моделями.

Внешняя модель — это та часть общей логической модели данных, с которой работает конкретное структурное подразделение фирмы. Границы разделения внешних моделей не четкие, то есть одни и те же данные могут получать различные структурные подразделения фирмы, но пополнять и редактировать данные может только одно конкретное подразделение. Форма отображения одних и тех же данных в разных подразделениях может быть различной. В зависимости от поставленных целей *логическая модель данных* может быть либо *иерархической*, либо *сетевой*, либо *реляционной*.

Отображение логической модели на конкретные технические средства называется *физической моделью данных*. Физическая модель строится на пятом этапе проектирования базы данных. При построении физической модели определяются технические характеристики персонального компьютера: объем оперативной памяти, необходимый объем памяти на жестком диске и т. д. Кроме того, на этом этапе определяют количество и типы индексов, методы доступа к данным. **При переходе от логической модели к физической модели необходимо обеспечить выполнение условия: концептуальная модель допускает некоторое расширение требований к базе данных без переделки самой базы данных. Это условие называется *вторым уровнем независимости данных*.**

Второй уровень независимости данных достигается, как правило, за счет хорошей техники и дисциплины программирования (например, константа задается только один раз, а ее значение передается всем подпрограммам через параметры).

ВОПРОСЫ:

1. Какие уровни описания информационной системы (типы моделей) предусматривает информационная модель данных?
2. Дайте характеристики каждому типу информационной модели данных.

2. Этапы проектирования базы данных

Процесс конструирования базы данных (ее проектирования и реализации) состоит из последовательности преобразований модели данных одного уровня в модель данных другого уровня.

Последовательности преобразований модели данных:

– систематизация объектов реального мира;

– создание информационных структур, описывающих систему объектов реального мира;

– создание структуры данных, которая используется для представления информационных структур в базе данных и прикладных программах;

– представление структуры памяти, используемой для хранения структур данных.

Процесс проектирования базы данных включает три этапа.

Первый этап - анализ предметной области или ***этап концептуального проектирования.***

На этапе концептуального проектирования осуществляется сбор, анализ и редактирование требований к данным.

Первым этапом проектирования БД любого типа является анализ предметной области, который заканчивается построением информационной структуры (концептуальной схемы). На данном этапе анализируются запросы пользователей, выбираются информационные объекты и их характеристики, которые предопределяют содержание проектируемой БД. На основе проведенного анализа структурируется предметная область. Анализ предметной области не зависит от программной и технической сред, в которых будет реализовываться БД.

Анализ предметной области целесообразно разбить на три фазы:

1) анализ требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение модели предметной области и проектирование схемы БД.

Рассмотрим каждую фазу данного этапа проектирования подробно.

Анализ концептуальных требований и информационных потребностей.

Во время анализа концептуальных требований и информационных потребностей необходимо выполнить:

- анализ требований пользователей к базе данных (концептуальных требований);
- выявление имеющихся задач по обработке информации, которая должна быть представлена в базе данных (анализ приложений),
- выявление перспективных задач (перспективных приложений);
- документирование результатов анализа

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требования к вводу, обновлению и корректировке информации Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Выявление информационных объектов и связей между ними.

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов.

При выборе информационных объектов следует ответить на следующие вопросы:

1. На какие классы можно разбить данные, подлежащие хранению в БД?

2. Какое имя можно присвоить каждому классу данных?

3. Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?

4. Какие имена можно присвоить выбранным наборам характеристик?

В ходе выявления связей между информационными объектами следует ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?

2. Какое имя можно присвоить каждому типу связей?

3. Каковы возможные типы связей, которые могут быть использованы впоследствии?

4. Имеют ли смысл какие-нибудь комбинации типов связей?

Далее проектировщик пытается задать ограничения на объекты и их характеристики. Под ограничением целостности обычно понимают логические ограничения, накладываемые на данные. При выявлении условий ограничения целостности проектировщик пытается ответить на следующие вопросы:

1. Какова область значений для числовых характеристик?

2. Каковы функциональные зависимости между характеристиками одного информационного объекта?

3. Какой тип отображения соответствует каждому типу связей?

Построение концептуальной модели предметной области.

Заключительная фаза анализа предметной области состоит в проектировании ее информационной структуры или концептуальной модели.

Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в, рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных. Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы "подняться вверх" над ПО и увидеть ее отдельные элементы. При этом, уровень детализации зависит от выбранной модели.

Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений. Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель «сущность-связь». Остановимся на наиболее известной модели данного типа, названной по фамилии автора, - модели П. Чена, или ER-модели. Основными конструкциями данной модели являются сущности и связи.

Под ***сущностью*** понимают основное содержание объекта предметной области , о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление. ***Экземпляр*** сущности - конкретный объект.

Например: сущность (объект) - студент, экземпляр сущности - Иванов А. В.; сущность (объект) - колледж, экземпляр сущности - КемПК.

Сущность принято определять ***атрибутами*** - поименованными характеристиками.

Например: сущность - студент, атрибуты - ФИО, год рождения, адрес, номер группы и т.д.

Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначений атрибута - идентифицировать сущность.

Второй этап - этап логического проектирования, т.е. моделирование построенной информационной системы и проектирование её отдельных составляющих в форме, соответствующей реальной базе данных. В процессе логического проектирования требования к данным преобразуются в структуры используемой СУБД. На этом этапе достаточно ответственным является выбор СУБД. Это обусловлено тем что, с одной стороны, число СУБД достаточно велико, а с другой - проектировщику необходимо оценить СУБД по множеству характеристик. Однако основным критерием отбора остается оценка того, насколько эффективно

внутренняя модель данных, поддерживаемая системой, способна описать построенную концептуальную схему.

Третий этап - этап физического проектирования (тесно связан с этапом реализации) - решаются вопросы, связанные с производительностью системы, определяются структуры хранения данных и методы доступа.

Процесс проектирования БД не может быть сделан автоматическим, так как для решения многих проблем участие человека является обязательным.

ВОПРОСЫ:

1. Перечислите этапы проектирования базы данных?
2. Раскройте суть каждого этапа проектирования базы данных?
3. На каком этапе проектирования осуществляется выбор СУБД?

3. Нормализация баз данных

Задача разработчика БД состоит в структуризации данных таким образом, чтобы устранить ненужное дублирование и обеспечить быстрый путь поиска необходимой информации

При проектировании БД могут появиться нежелательные свойства, такие как избыточность, аномалии обновления, аномалии включения, аномалии удаления и др. Для уменьшения нежелательных характеристик БД к схемам отношений применяют процедуры нормализации.

Нормализация - это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Нормализация – это процесс разделения информации на структурные единицы, т.е. таблицы.

Нормализация БД должна быть выполнена с учётом следующего правила: таблицы, которые содержат повторяющуюся информацию, для устранения дублирования значений должны быть разделены на отдельные таблицы, что приводит к сокращению размеров БД.

Если определенным образом ограничить наличие зависимостей записей в схеме данных, то получим нормальные формы отношения: первую нормальную форму (1НФ), вторую нормальную форму (2НФ), третью нормальную форму (ЗНФ), нормальную форму Бойса-Кодда (НФБК).

Так же можно сказать, что процесс нормализации представляет собой приведение таблиц к требуемому уровню нормальности: первый, второй и третий. Каждый уровень нормальности соответствует определённой нормальной форме.

Теория нормализации основана на концепции нормальных форм. Говорят, что таблица находится в данной нормальной форме, если она удовлетворяет определенному набору требований. Теоретически существует пять нормальных форм, но на практике обычно используются только первые три. Первые две нормальные формы являются промежуточными шагами для приведения базы данных к третьей нормальной форме.

Первая нормальная форма

Реляционная таблица находится в *первой нормальной форме,* если все значения ее полей атомарны, т.е. неделимы и все записи уникальны и значения элементов в домене не являются ни списками, ни множествами.

Схема БД находится в 1НФ, если каждая схема отношения находится в 1НФ.

Пример. Имеем отношение: *R рождение* (ФИО, дата рождения). Переведем отношение *R рождение* в 1НФ и получим:

R *рождение* (фамилия, имя, отчество, день рождения, месяц, год рождения)

Вторая нормальная форма

Реляционная таблица находится во *второй нормальной форме* (2НФ), если она находится в первой нормальной форме и ее *неключевые* поля полностью зависят от *всего* первичного ключа.

Чтобы перейти от первой нормальной формы ко второй, нужно выполнить следующую последовательность действий:

1. Определить, на какие части можно разбить первичный ключ, так чтобы некоторые из не ключевых полей зависели от одной из этих частей (эти части не обязаны состоять из одной колонки).

2. Создать новую таблицу для каждой такой части ключа и группы, зависящих от нее полей и переместить их в эту таблицу. Часть бывшего первичного ключа станет при этом первичным ключом новой таблицы.

3. Удалить из исходной таблицы поля, перемещенные в другие таблицы, кроме тех, которые станут внешними ключами.

Схема БД находится в 2НФ, если схема каждого отношения БД находится в 2НФ.

Пример. Рассмотрим отношение R *учеба* (факультет, группа, дисциплина, вид занятий, преподаватель, квалификация преподавателя)

Данное отношение не находится во 2НФ, так как неключевые атрибуты "факультет" и "квалификация преподавателя" функционально неполно зависят от ключа, например, неключевой атрибут "квалификация преподавателя" зависит только от части ключа - от "преподавателя" Приведем отношение R *учеба* ко 2НФ путем его декомпозиции с помощью проекции исходного отношения:

R1 (группа, дисциплина, вид занятий, преподаватель)

R2 (дисциплина, вид занятий, квалификация преподавателя)

R3 (группа, факультет)

R4 (преподаватель, квалификация преподавателя)

Каждое из отношений Rl, R2, R3, R4 находится во 2НФ. Исходное отношение R учеба может быть восстановлено путем эквисоединения полученных отношений R1, R2, R3, R4.

Третья нормальная форма

Приведение отношений к ЗНФ требует дальнейшего сокращения возможных функциональных зависимостей между атрибутами. Переход к ЗНФ происходит через проекцию исходного отношения. Переход может быть не единственным. Можно говорить об оптимальной ЗНФ, если число отношений в ЗНФ будет минимально.

Третья нормальная форма не допускает функциональных зависимостей между неосновными атрибутами отношения, т.е.

транзитивная зависимость неосновных атрибутов от ключа исключается.

Реляционная таблица находится в *третьей нормальной форме,* если она находится во второй нормальной форме, и ее *некпючевые* поля полностью зависят *только* от первичного ключа, т.е. не взаимозависимы.

Чтобы перейти от второй нормальной формы к третьей, нужно выполнить следующую последовательность действий:

1. Определить все поля, от которых зависят другие поля.
2. Создать новую таблицу для каждого такого поля или группы полей и группы зависящих от него полей и переместить их в эту таблицу. Поле или группа полей, от которого зависят перемещенные поля, станет при этом первичным ключом новой таблицы.
3. Удалить из исходной таблицы поля, перемещенные в другие таблицы, кроме тех, которые станут внешними ключами.

ВОПРОСЫ

1. Что такое нормализация?
2. Что означает, когда таблица находится, в какой либо, нормальной форме?
3. В чём суть первой нормальной формы?
4. В чём суть второй нормальной формы?
5. В чём суть третьей нормальной формы?

4. Принципы и средства проектирования баз данных.

К современным базам данных, а, следовательно, и к СУБД, на которых они строятся, предъявляются следующие основные требования:

- Высокое быстродействие (малое время отклика на запрос). Время отклика - промежуток времени от момента запроса к БД до фактического получения данных;
- Простота обновления данных;
- Независимость данных - возможность изменения логической и физической структуры БД без изменения представлений пользователей;

- Совместное использование данных многими пользователями.
- Безопасность данных - защита данных от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения;
- Стандартизация построения и эксплуатации БД (фактически СУБД);
- Адекватность отображения данных соответствующей предметной области;
- Простой интерфейс пользователя.

Важнейшими являются первые два противоречивых требования: повышение быстродействия требует упрощения структуры БД, что, в свою очередь, затрудняет процедуру обновления данных, увеличивает их избыточность.

Безопасность данных включает их целостность и защиту. *Целостность данных* - устойчивость хранимых данных к разрушению и уничтожению, связанных с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей. Она предполагает:

- отсутствие неточно введенных данных или двух одинаковых записей об одном и том же факте;
- защиту от ошибок при обновлении БД;
- невозможность удаления (или каскадное удаление) связанных данных разных таблиц;
- не искажение данных при работе в многопользовательском режиме и в распределенных базах данных;
- сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается триггерами целостности - специальными приложениями-программами, работающими при определенных условиях.

Защита данных от несанкционированного доступа предполагает ограничение доступа к конфиденциальным данным и может достигаться:

- введением системы паролей;
- получением разрешений от администратора базы данных (АБД);

- запретом от администратора БД на доступ к данным;
- формирование видов - таблиц, производных от исходных и предназначенных конкретным пользователям.

Стандартизация обеспечивает преемственность поколений СУБД, упрощает взаимодействие БД одного поколения СУБД с одинаковыми и различными моделями данных. При этом может быть осуществлен как локальный, так и удаленный доступ к данным (технология клиент/сервер или сетевой вариант).

Проектирование баз данных - процесс решения класса задач, связанных с созданием баз данных.

Основные задачи проектирования баз данных:

- Обеспечение хранения в БД всей необходимой информации.
- Обеспечение возможности получения данных по всем необходимым запросам.
- Сокращение избыточности и дублирования данных.
- Обеспечение целостности данных (правильности их содержания): исключение противоречий в содержании данных, исключение их потери и т.д.

Классификация СУБД

Классифицировать СУБД можно по следующим признакам:

– по используемой модели данных,

– по способу организации БД (централизованная или распределенная);

– по реализуемым режимам работы (однопользовательский, многопользовательский и т.д.);

– по способам физической организации данных.

Требования к СУБД

Выбор СУБД является одним из важных этапов при разработке приложений баз данных. Выбранный программный продукт должен удовлетворять как текущим, так и будущим потребностям предприятия, при этом следует учитывать финансовые затраты на приобретение необходимого оборудования, самой системы, разработку необходимого программного обеспечения на ее основе, а также обучение персонала. Кроме того, необходимо убедиться, что новая СУБД способна принести предприятию реальные выгоды.

Очевидно, наиболее простой подход при выборе СУБД основан на оценке того, в какой мере существующие системы удовлетворяют основным требованиям создаваемого проекта информационной системы. Более сложным и дорогостоящим вариантом является создание испытательного проекта на основе нескольких СУБД и последующий выбор наиболее подходящего. Но и в этом случае используются определенные критерии отбора.

Перечень требований к СУБД может изменяться в зависимости от поставленных целей. Тем не менее, можно выделить несколько групп критериев:

– реализуемые режимы работы с БД и максимальное число пользователей одновременно обращающихся к базе;

– модель данных (предусмотренные типы данных, средства поиска, реализация языка запросов, средства поддержания целостности базы данных);

– особенности архитектуры и функциональные возможности (масштабируемость, которая определяет, сможет ли данная СУБД соответствовать росту информационной системы, распределенность, сетевые возможности);

– контроль работы системы (возможность управления использованием памяти, возможность самоконфигурирования, самодиагностики производительности);

– особенности разработки приложений (средства проектирования, поддержка большого количества национальных языков, возможности разработки Web-приложений, поддерживаемые языки программирования);

– производительность, т.е. отношение количества запросов, обрабатываемых за некий промежуток времени, к стоимости всей системы, возможности параллельной обработки данных, возможности оптимизирования запросов);

– надежность (сохранность информации при сбоях, обеспечение защиты данных от несанкционированного доступа);

– требования к рабочей среде (минимальные требования к оборудованию, максимальный размер адресуемой памяти, операционные системы, под управлением которых способна работать СУБД);

– требуемый уровень квалификации персонала;

– смешанные критерии (качество и полнота документации, стоимость, стабильность производителя, распространенность СУБД).

Общая характеристика и классификация CASE-средств

Проектирование информационной системы - это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Применение структурной методологии проектирования при неавтоматизированной (ручной) разработке затруднено.

Это способствовало появлению программно-технологических средств, реализующих CASE-технологию (Computer Aided Software Engineering) создания и сопровождения ИС. Под термином *CASE-средства* понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного программного обеспечения (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО.

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку

синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

Современный рынок программных средств насчитывает около 300 различных CASE-средств. Это как относительно дешевые системы для персональных компьютеров с ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла программного обеспечения.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям.

Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ и включает следующие основные типы:

– средства анализа, предназначенные для построения и анализа моделей предметной области;

– средства анализа и проектирования, использующиеся для создания проектных спецификаций. Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;

– средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных для наиболее распространенных СУБД;

– средства разработки приложений;

– средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций;

– средства планирования и управления проектом;

– средства конфигурационного управления;

– средства тестирования;

– средства документирования.

Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает в себя:

– отдельные локальные средства, решающие небольшие автономные задачи (tools),

– частично интегрированные средства, охватывающие большинство этапов жизненного цикла ИС (toolkit)

– полностью интегрированные средства, поддерживающие весь жизненный цикл ИС и связанные общим репозиторием.

Помимо этого, CASE-средства можно классифицировать по следующим признакам:

– применяемым методологиям и моделям систем и БД;
– степени интегрированности с СУБД;
– доступным платформам.

ВОПРОСЫ:

1. Дайте определение CASE-средствам и CASE-технологии.
2. Назовите признаки классификации CASE-средств.

РАЗДЕЛ 3. МЕТОДЫ ОПИСАНИЯ СХЕМ БАЗ ДАННЫХ В СОВРЕМЕННЫХ СУБД

1. Понятие объекта баз данных. Назначение объектов баз данных.

Объектами баз данных называют их структурные составляющие, такие как таблицы, отчеты, триггеры, ограничения и т.п. Они выполняют различные функции по хранению и обработке информации.

Кроме таблиц база данных может содержать и другие типы объектов. Привести полную классификацию возможных объектов баз данных затруднительно, поскольку каждая система управления базами данных может реализовать свои типы объектов. Однако основные типы объектов мы можем

рассмотреть на примере СУБД Microsoft Access, которая относится к системам, ориентированным на пользователя.

Объекты представлены в окне базы данных Access, все операции по работе с объектами собственно базы данных и приложений начинаются в этом окне.

Объекты СУБД

Таблицы создаются пользователем для хранения данных об одной сущности – одном информационном объекте модели данных предметной области. Таблица состоит из полей (столбцов) и записей (строк). Каждое поле содержит одну характеристику информационного объекта предметной области. В записи собраны сведения об одном экземпляре информационного объекта.

База данных Access может включать до 32768 объектов (включая формы, отчеты и т.д.) одновременно может открываться до 2048 таблиц. Таблицы можно импортировать из баз данных dBase, FoxPro, Paradox и других приложений, из базы данных архитектуры клиент-сервер, таких как Microsoft SQL Server, или из электронных таблиц, таких как Excel и Lotus 1-2-3. База данных Access позволяет работать с таблицами перечисленных источников путем организации связи с ними.

Таблицы базы данных могут иметь различное назначение (например, таблицы постоянной информации, таблицы переменной информации). Таблицы постоянной информации (условно постоянной) должны содержать данные, не меняющиеся в течение длительного времени (например, списки сотрудников организации, названия технологических операций, применяемых при изготовлении продукции, и т.п.). Таблицы переменной информации — это таблицы, информация об объектах в которых постоянно дополняется или изменяется пользователем.

Запросы. Запросы на выборку служат для выборки нужных данных из одной или нескольких связанных таблиц. Результатом выполнения запроса является таблица, которая может быть использована наряду с другими таблицами базы данных при обработке данных. В запросе можно указать, какие поля выбранных таблиц нужно выбрать, как на их основе сформировать записи и выбрать нужные. Запрос может формироваться с помощью QBE-запросов (Query By Example,

Запрос по образцу) или посредством инструкции SQL. Запросы действия позволяют обновлять, удалять или добавлять данные в таблицы, а также создавать новые таблицы на основе уже существующих.

Схемы данных, определяют с помощью каких полей, таблицы связываются между собой, как будет выполняться объединение данных этих таблиц, нужно ли проверять связную целостность при добавлении и удалении записей, изменении ключей таблиц. Схемы данных на панели объектов в окне базы данных отображаются только в проектах Access, работающих с базами данных сервера.

Формы являются основным средством создания диалогового интерфейса приложения пользователя. Форма может создаваться для ввода и просмотра взаимосвязанных данных базы на экране в удобном виде, соответствующем привычному для пользователя документу. Кнопочные формы могут использоваться для создания панелей управления в приложении. В формы могут вставляться рисунки, диаграммы, звуковые фрагменты, видео. Форма может включать подчиненные формы. В форму могут включаться процедуры обработки событий, которые позволяют управлять процессом ввода, просмотра и корректировки данных. Такие процедуры хранятся в модуле формы.

Отчеты предназначены для форматирования выходных документов любых форматов, содержащих результаты решения задач пользователя, и вывода их на печать. Использование графических объектов позволяет дополнять данные отчета иллюстрациями.

Страницы доступа к данным являются диалоговыми Web – страницами, которые поддерживают динамическую связь с базой данных и позволяют просматривать, редактировать и вводить данные в базу, работая в окне браузера.

Макросы позволяют автоматизировать некоторые действия в приложении пользователя. Макрос является программой, состоящей из последовательности макрокоманд, которая выполняется при наступлении некоторого события в объекте приложения или его элементе управления. Создание макросов осуществляется в диалоговом режиме путем выбора

нужных макрокоманд и задании параметров, используемых ими при выполнении.

Модули содержат процедуры на языке Visual Basic for Applications. Могут создаваться процедуры-подпрограммы, процедуры-функции, которые разрабатываются пользователем для реализации нестандартных функций в приложении пользователя, и процедуры для обработки событий. Использование процедур позволяет создать законченное приложение, которое имеет собственный графический интерфейс, позволяющий запросить выполнение всех функций приложения.

ВОПРОСЫ:

1. Назовите основные объекты систем управления реляционными базами данных.
2. Опишите назначение таблиц с постоянной информацией, с переменной информацией.

2. Системы управления базами данных (СУБД) и манипулирование данными.

Система управления базами данных (СУБД) - это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования *БД* многими пользователями.

СУБД представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная база данных. В связи с этим полезно поговорить о системе программно-технических, организационных и "человеческих" составляющих. Программные средства включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование БД, трансляторы.

Базовыми внутренними языками программирования являются языки четвертого поколения. В качестве базовых языков могут использоваться C, C++, Pascal, Object Pascal. Язык C++ позволяет строить программы на языке Visual Basic с широким спектром возможностей, более близком и понятном даже пользователю-непрофессионалу, и на непроцедурном

(декларативном) языке структурированных запросов SQL. Следует отметить, что исторически для системы управления базой данных сложились три языка:

- *язык описания данных (ЯОД),* называемый также языком описания схем, для построения структуры ("шапки") таблиц БД;
- *язык манипулирования данными (ЯМД)* - для заполнения БД данными и операций обновления (запись, удаление, модификация);
- *язык запросов* - язык поиска наборов величин в файле в соответствии с заданной совокупностью критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД (язык преобразования критериев в систему команд).

В настоящее время функции всех трех языков выполняет язык SQL, относящийся к классу языков, базирующихся на исчислении кортежей (кортеж чаще всего является единицей информации), языки СУБД FoxPro, Visual Basic for Application (СУБД Access) и т.д.

Вместе с тем сохранились и языки запросов, например язык запросов по примеру Query By Example (QBE) класса исчисления доменов. Отметим, что эти языки в качестве «информационной единицы» БД используют отдельную запись. С помощью языков БД создаются приложения, базы данных и интерфейс пользователя, включающий экранные формы, меню, отчеты. При создании БД на базе СУБД FoxPro эти элементы (объекты) фиксируются в отдельных файлах, которые, в свою очередь, сосредоточиваются в одном файле, называемом *проектом*. После отработки БД проект преобразуется в приложение. В СУБД Access все созданные объекты размещаются в одном файле.

Функциональные возможности СУБД

Современные СУБД обеспечивают:

- набор средств для поддержки таблиц и соотношений между связанными таблицами;
- развитый пользовательский интерфейс, позволяющий вводить и модифицировать информацию, выполнять поиск;

− средства программирования для разработки собственных приложений.

По назначению СУБД делятся на:

− специализированные (например, для хранения геофизической информации) - эти системы оптимизированы для решения конкретных задач;

− СУБД общего назначения, предназначенные для решения широкого круга задач обработки данных, среди них;

− СУБД для работы на персональных компьютерах и в локальных сетях в режиме файл-сервер (dBASE, Paradox, FoxPro, Access);

− СУБД, функционирующие в режиме клиент-сервер, в которых сервер базы данных используется не только для хранения информации, но и для обработки запросов к базе данных - на рабочую станцию возвращается только результат выполнения запроса (это уменьшает поток данных в сети, кроме того, обработка данных на сервере обычно осуществляется гораздо быстрее, чем на рабочей станции - в связи с более мощным компьютером в качестве сервера и более совершенными средствами обработки таких СУБД - это Oracl, Informix и другие).

По степени сложности СУБД можно классифицировать на системы

− для обработки небольших объемов информации (обычно, встроенные в интегрированные пакеты MS Works, FrameWork и другие);

− ориентированные на конечного пользователя (Access, Paradox);

− ориентированные на разработку приложений (dBase, FoxPro, СУБД типа клиент - сервер) - они требуют умения программировать на конкретном языке и используются при создании сложных систем (пользователю этой системы владеть языками не нужно). Встроенные специализированные языки программирования для разработки приложений имеются сейчас практически во всех СУБД.

Производительность СУБД оценивается:

− временем выполнения запросов;

− скоростью поиска информации;

- временем выполнения операций импортирования данных из других форматов;
- скоростью выполнения таких операций как обновления, вставка, удаление данных;
- максимальным числом параллельных обращений к данным в многопользовательском режиме;
- временем генерации отчёта.

На *производительность СУБД* оказывают влияния 2 фактора:

- правильное проектирование
- построения БД.

СУБД, которые следят за соблюдением целостности данных, несут дополнительную нагрузку, которую не испытывают другие программы;

Целостность данных подразумевает наличие средств, позволяющих удостовериться, что *информация* в *БД* всегда остаётся корректной и полной.

Операции, обеспечивающие *безопасность*:

- шифрование прикладных программ;
- шифрование данных;
- защита паролем;
- ограничение уровня доступа.

Хороший уровень безопасности в *СУБД* dBase IV, Access

Для сохранения информации используется двойной подход. Некоторые *операции* сохранения происходят в обход операционной системы

Целостность должна обеспечиваться независимо от того, каким образом данные заносятся в *память*, не конкретных действий пользователей, пробоев сети и т.п. Он предусматривает назначение паролей для индивидуальных пользователей или групп пользователей и присвоение различных прав доступа отдельно таблицам, запросам, отчётам на уровне пользователя или группы.

ВОПРОСЫ:

1. Как классифицируются СУБД.
2. Перечислите языки управления БД, дайте их характеристики.

3. Чем отличаются системы общего назначения от специализированных систем?

4. С помощью, каких показателей оценивается производительность СУБД?

3. Методы описания и построения схем баз данных в современных СУБД.

Схема базы данных включает в себя описания содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных.

Постоянные данные в среде базы данных включают в себя схему и базу данных. Система управления базами данных(СУБД) использует определения данных в схеме для обеспечения доступа и управления доступом к данным в базе данных.

Схема данных (от англ. *Database schema*) — её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности.

Схемы в общем случае хранятся в словаре данных. Хотя схема определена на языке базы данных в виде текста, термин часто используется для обозначения графического представления структуры базы данных.

Основными объектами графического представления схемы являются таблицы и связи, определяемые внешними ключами.

Есть и другое понятие схемы в теории баз данных.

Схема (SCHEMA) является одним из основных объектов базы данных Oracle Database. Близкое понятие (RIS Schema) существует в RIS-интерфейсе доступа к базам данных. SCHEMA также появилась и в Microsoft SQL Server 2005 и формально определяется как набор объектов в базе данных.

В Oracle схема привязывается только к одному пользователю (USER) и является логическим набором объектов базы данных. Схема создаётся при создании пользователем первого объекта, и все последующие объекты, созданные этим пользователем, становятся частью этой схемы.

Схема может включать другие объекты, принадлежащие этому пользователю:

- таблицы,
- последовательности,
- хранимые программы,
- кластеры,
- связи баз данных,
- триггеры,
- библиотеки внешних процедур,
- индексы,
- пакеты,
- хранимые функции и процедуры,
- синонимы,
- представления,
- снимки,
- объектные таблицы,
- объектные типы,
- объектные представления.

Существуют и подобъекты схемы, такие как:

- столбцы: таблиц и представлений,
- секции таблиц,
- ограничения целостности,
- триггеры,
- пакетные процедуры и функции и другие элементы, хранимые в пакетах (курсоры, типы и т. п).

Существуют объекты, независимые от схемы:

- каталоги,
- профили,
- роли,
- сегменты,
- табличные области,
- пользователи.

Уровни схемы базы данных

- Концептуальная схема — карта концепций и их связей
- Логическая схема — карта сущностей и их атрибутов и связей
- Физическая схема — частичная реализация логической схемы

- Схема объекта — объект БД Oracle

В СУБД Access процесс создания реляционной базы данных включает создание *схемы данных*. Схема данных наглядно отображает логическую структуру базы данных: таблицы и связи между ними, а также обеспечивает использование установленных в ней связей при обработке данных.

Для нормализованной базы данных, основанной на одно-многозначных и одно-однозначных отношениях между таблицами, в схеме данных для связей таких таблиц по первичному ключу или уникальному индексу главной таблицы могут устанавливаться параметры обеспечения *связной целостности*.

При поддержании целостности взаимосвязанных данных не допускается наличия записи в подчиненной таблице, если в главной таблице отсутствует связанная с ней запись. Соответственно при первоначальной загрузке базы данных, а также корректировке, добавлении и удалении записей система допускает выполнение операции только в том случае, если она не приводит к нарушению целостности.

Связи, определенные в схеме данных, автоматически используются для объединения таблиц при разработке многотабличных форм, запросов, отчетов, существенно упрощая процесс их конструирования.

В схеме данных связи могут устанавливаться для любой пары таблиц, имеющих одинаковое поле, позволяющее объединять эти таблицы.

Схема данных (Relationships) определяет, с помощью каких полей таблицы связываются между собой, как будет выполняться объединение данных этих таблиц, нужно ли проверять связную целостность при добавлении и удалении записей, изменении ключей таблиц.

Схемы данных (Рисунок 6) в области навигации в окне базы данных отображаются только в проектах Access, работающих с базами данных сервера. Для отображения схемы данных в базах данных Access используется команда *Схема данных* (Relationships), размещенная на вкладке ленты *Работа с базами данных (*Database Tools) в группе *Отношения* (Relationships).

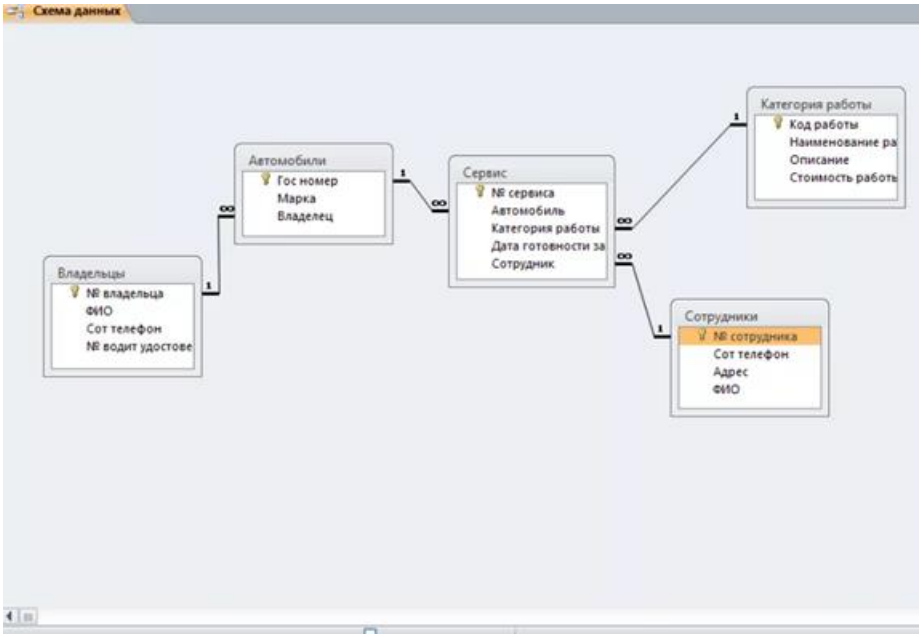


Рисунок 6 - Схемы данных

Сформулируем основные правила установления связей между таблицами.

1. Выбрать из двух связываемых таблиц главную и подчиненную.

2. В каждой таблице выбрать ключевое поле. Ключевое поле главной таблицы называют первичным ключом. Ключевое поле подчиненной таблицы называют внешним ключом.

3. Связываемые поля таблиц должны иметь один тип данных.

4. Между таблицами устанавливаются следующие типы связей: «один к одному»; «один ко многим»; «многие ко многим»:

- связь «один к одному» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана только с одной строкой подчиненной таблицы;
- связь «один ко многим» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана с несколькими строками подчиненной таблицы; при этом

любая строка подчиненной таблицы связана только с одной строкой главной таблицы;

– связь «многие ко многим» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана с несколькими строками подчиненной таблицы и в то же время одна строка подчиненной таблицы связана с несколькими строками главной таблицы.

При изменении значения первичного ключа в главной таблице возможны следующие варианты поведения зависимой таблицы.

Каскадирование (Cascading). При изменении данных первичного ключа в главной таблице происходит изменение соответствующих данных внешнего ключа в зависимой таблице. Все имеющиеся связи сохраняются.

Ограничение (Restrict). При попытке изменить значение первичного ключа, с которым связаны строки в зависимой таблице, изменения отвергаются. Допускается изменение лишь тех значений первичного ключа, для которых не установлена связь с зависимой таблицей.

Установление (Relation). При изменении данных первичного ключа внешний ключ устанавливается в неопределенное значение (NULL). Информация о принадлежности строк зависимой таблицы теряется. Если изменить несколько значений первичного ключа, то в зависимой таблице образуется несколько групп строк, которые ранее были связаны с измененными ключами. После этого невозможно определить, какая строка с каким первичным ключом была связана.

ВОПРОСЫ:

1. Для чего предназначена схема базы данных?
2. Какие объекты может включать схема базы данных?
3. Опишите основные правила установления связей между таблицами.
4. Перечислите варианты поведения зависимой таблицы.

РАЗДЕЛ 4. ОРГАНИЗАЦИЯ БАЗ ДАННЫХ

1. Структуры данных СУБД. Организация таблиц, индексов

Понятие структуры данных используется на всех уровнях представления предметной области и реализуется как:

- **структура информации –** схематичная форма представления сложных композиционных объектов и связей реальной предметной области, выделяемых как актуально необходимые для решения прикладных задач;
- **структура данных -** атрибутивная форма представления свойств и связей предметной области, ориентированная на выражение описания данных средствами формальных языков;
- **структура записи** – целесообразная (учитывающая особенности физической среды) реализация способов хранения данных и организации доступа к ним как на уровне отдельных записей, так и их.

Структура является общепринятым и удобным инструментом, одинаково эффективно используемым как на уровне сознания человека при работе с абстрактными понятиями, так и на уровне логики машинных алгоритмов.

Выделение трех видов структур, относящихся к представлению объектов предметной области, имеет, в некотором смысле, принципиальный характер.

Структура информации – это неотъемлемое свойство информации о некоторой совокупности объектов предметной области, имеет в контексте практической задачи (решаемой субъектом), в общем случае без учета того, будут ли для ее решения использованы средства программирования и вычислительные машины. Структурирование информации осуществляется системным аналитиком и сводится к выделению операционных объектов и определении, их характеристических свойств и взаимосвязей.

Структура данных – это определение информационных массивов (т.е. состава и взаимосвязей данных на логическом уровне, соответствующих характеру информации и видам соответствующих преобразований). При определении структур

данных необходимо установить не только состав массива, но и определить оптимальную, их взаимосвязь.

Структура записи – это определение структуры физической памяти: выделение, освобождение и защита областей физического носителя, способы адресации пересылки.

Рассмотрим разновидности и топологию «компьютерных» логических структур данных с точки зрения особенности их организации.

Физическому понятию **структура** соответствует **запись данных**.

Запись – это упорядоченная в соответствии с характером взаимосвязей совокупность полей (элементов) данных, размещаемых в памяти в соответствии с их типом.

Поле представляет собой минимальную адресуемую (идентифицируемую) часть памяти – единицу данных, на которую можно ссылаться при обращении к данным.

Таким образом, структура данных – это способ отражения значений в памяти: размер области и порядок ее выделения который и определит характер процедуры адресации/выборки.

Классификация структур данных должна проводиться с двух точек зрения:

1. по характеру взаимосвязи элементов структуры (с точки зрения порядка их размещения/выборки), виды структур можно разделить на **линейные** и **нелинейные.**

2. по характеру информации, представляемой структурой (с точки зрения однородности и «элементарности») типов данных, отражающих понятийную структуру предметной области):

- однородные структуры, где все элементы находятся на одном понятийном уровне и имеют один тип данных;
- неоднородные (композиционные), где элементы относятся к нескольким понятийным уровням или имеют разную природу.

Линейные структуры

К линейным структурам относятся: **массивы, последовательности, таблицы.**

Порядок следования (и, соответственно выборки) элементов таких структур имеет линейный характер и соответствует порядку расположения элементов в памяти: один за другим без каких-либо

промежутков. Адрес элемента соответствует его положению и определяется индексом – порядковым номером элемента в последовательности размещения.

Особенностью линейной структуры является то, что при последовательной организации (размещении) одна допускает возможность прямого доступа к произвольному элементу, поскольку условие однородности однотипности предполагает, что все элементы занимают расположение строго последовательно области одинакового физического адреса элемента по значению его индекса.

Массив представляет собой совокупность однотипных элементов, причем число элементов массива известно до его размещения, что позволяет строить гибкие многомерные системы адресации.

Последовательность также как и массив это совокупность однотипных элементов, однако, число элементов до размещения неизвестно, хотя каждая конкретная последовательность имеет конечную длину до начала обработки. Необходимо считать длину последовательности бесконечной.

Таблица это последовательность обычно представляется строками, совокупностью различных элементов, иначе это множество записей каждая из которых представляет набор поименованных полей. С точки зрения размещения элементов таблица может быть представлена как одномерный массив с однородными композиционными элементами, каждый из которых представляет собой совокупность разнотипных элементов.

Таблицы БД состоят из полей – столбцов, записей – строк. Каждое поле таблицы имеет свойства и тип. Имена полей соответствуют атрибутам объектов. Каждая таблица должна иметь хотя бы одно поле.

Структура таблицы БД включает:

Описание полей, ключ, индексы, ограничения значений полей, ограничения ссылочной целостности между таблицами, пароли.

Описание полей – имя поля, свойства, тип хранимой информации.

Ключ (первичным ключом) таблицы БД представляет собой комбинацию полей, однозначно определяющих запись в таблице.

Ключевым, называется поле, на котором строится ключ. Ключ может быть *простым* (состоит из одного поля) и *составным* (состоит из нескольких полей). При поиске нужной записи выполняется не последовательный просмотр всей таблицы, а не посредственный доступ к записи на основании упорядоченных значений ключа. Ключ должен быть уникальным.

Основным предназначением индексов является ускорение процесса поиска и сортировки записей в таблице, а так же организация проверки повторяющихся значений. *Индексы*, как и ключи, строятся на полях таблицы, однако они могут допускать повторение значений составляющих их полей. В этом основное отличие ключей от индексов. *Индексированными* называют поля, на которых построен индекс. Как и ключ, простой индекс состоит из одного поля, а сложный – из нескольких.

Процесс создания индексов называют *индексированием.*

Использование индекса позволяет:

1. увеличить скорость поиска.
2. производить сортировку записей в таблице.
3. устанавливать связи между таблицами.
4. использовать ограничения ссылочной целостности.

Для решения двух последних задач индекс используется совместно с ключом второй таблицы. Индекс и ключ, представляют собой своеобразное оглавление таблицы БД, которое просматривается перед обращением к её записям.

ВОПРОСЫ

1. Для чего предназначены таблицы в реляционных БД?
2. Что входит в структуру таблиц?
3. Какие поля называют ключевыми? Виды ключей?
4. Какие поля называют индексированными? Виды индексов?

2. Технология разработки таблиц баз данных.

Таблицы – это основные объекты любой БД, в которых хранится вся информация, имеющиеся в базе, а также структура

базы (поля, их типы и свойства). Все другие объекты (формы, запросы, отчёты) зависят от данных таблиц.

Создание таблиц с помощью мастера производится путём выбора типовой таблицы и необходимых полей из типовой таблицы или нескольких типовых таблиц. Выбранные имена полей можно редактировать. После ввода имени таблицы выбирается ключевые поля, позволяющие осуществлять связи между таблицами в БД.

При создании таблиц в режиме Конструктора выводится пустая структура таблицы, в которую необходимо ввести имена полей, указать типы данных в полях и задать размеры полей. В нижней части бланка структуры таблицы задаются свойства полей таблицы, позволяющие изменить способы хранения и отображения данных.

Рассмотрим открытее, редактирование и модификацию таблиц на примере работы с таблицами СУБД Access.

Таблица СУБД Access может быть представлена в двух режимах:

- *в режиме конструктора* (кнопка **Конструктор**), предназначенном для создания и изменения структуры таблицы;
- *в режиме просмотра таблицы* (кнопка **Открыть**), предназначенном для просмотра, редактирования и ввода данных. Иногда данный режим называют режимом таблицы.

Режим просмотра таблицы - позволяет добавлять, изменять или анализировать данные. При работе с данными в таблице возможно:

- просмотреть данные из таблицы;
- найти интересующую Вас информацию или заменить устаревшие сведения на более новые;
- отсортировать (то есть расположить в определённом порядке) имеющуюся информацию в таблице;
- вводить новые записи в таблицу;
- удалять ненужные записи из таблицы;
- копировать и перемещать данные;
- убрать с экрана ненужные для работы в данный период времени столбцы (скрыть столбцы) или расположить их в удобном для Вас порядке (то есть изменить внешний вид таблицы);

- работать с ограниченным Вами (по определенным признакам) набором данных, то есть применить фильтр.

Структура окна таблицы базы данных ничем не отличается от структуры других окон в среде Windows.

Режим конструктора таблицы - позволяет:

- изменять имена полей;
- изменять типы данных;
- задавать и изменять свойства полей;
- добавлять в таблицу новые поля;
- удалять имеющиеся поля.

Итак, работа в режиме конструктора сводится к изменению (определению) свойств полей таблицы, а не к изменению самих данных (как это происходит в режиме просмотра таблицы). Для того, чтобы открыть таблицу в режиме Конструктора необходимо выделить соответствующую таблицу и нажать кнопку Конструктор!

Конструктор - это режим определения свойств объектов базы данных таблиц, запросов, форм, отчетов, макросов, модулей.

Конструктор имеет свое окно. Окно конструктора таблицы условно можно разделить на две части. В верхней части окна расположен бланк, который напоминает обычную таблицу из трёх колонок (столбцов). Первый столбец бланка предназначен для отображения *имен полей,* а второй — *типов данных поля,* третий - не обязателен для заполнения - предназначен для текста пояснения, отображаемого в строке состояний. Таким образом, каждая строка бланка конструктора содержит информацию о конкретном поле таблицы. В нижней части окна отображаются *свойства текущего* поля.

Имена полей должны быть уникальными и отображать содержимое, не должны содержать знаков препинания, скобок и начинаться с пробелов. Количество символов в имени поля не должно превышать 64.

Типы данных

Microsoft Access поддерживает следующие *типы данных:*

Текстовый. Данные этого типа представляют собой символьную строку, то есть последовательность символов фиксированной длины. Размер текстового поля может находиться в пределах от 1 до 255 символов. При определении размера

необходимо учитывать размер данных (то есть количество символов) в этом поле. Слишком малый размер поля неприятен тем, что данные в нём могут не поместиться, а слишком большой - нерациональное использование памяти компьютера. В процессе работы Вы можете изменить размер поля в режиме конструктора, но при этом помните, что при уменьшении размера поля возможно усечение данных, а, значит, потеря части информации.

Числовой. Данные этого типа предназначаются для характеристики объектов БД, которые могут участвовать в математических расчётах. В строке *"Размер поля"* бланка свойств числового поля возможен выбор из следующего списка:

1) *Байт* - целые числа в пределах от 0 до 255. Размер - 1 байт.

2) *Целое* - целые числа в пределах от -32 768 до 32 767. Размер - 2 байта.

3) *Длинное целое* (размер - 4 байта) - целые числа в пределах от -2 147 483 648 до 2 147 483 647.

4) *С плавающей точкой* (размер - 4 байта) - действительные числа в пределах от $-3{,}402823*10^{38}$ до $3{,}402823*10^{38}$ (после запятой 7 знаков).

5) *С плавающей точкой* (размер - 8 байт) - действительные числа в пределах от $-1{,}797693*10^{308}$ до $1{,}797693*10^{308}$ (после запятой 15 знаков).

Логический. Данные логического типа могут принимать *только два* значения: "истина", которое в Access обозначается "Да", или "ложь", имеющее обозначение "Нет". Access также предлагает следующие *варианты* значений логического типа: Истина/Ложь, включено/выключено. Пользователь по своему усмотрению определяет вариант обозначения данных в поле логического типа. Поля данного типа не могут быть ключевыми, но могут быть индексированными.

Денежный. Данные этого типа аналогичны данным числового типа и отличаются от них только характеристиками вводимых чисел. Целая часть, вводимого числа может содержать до 15 десятичных разрядов, после запятой 4. По умолчанию денежный формат представляет собой число в диапазоне от -922337203685447.5808 до 922337203685447.5808 с двумя знаками

после запятой с разделением разрядов и обозначением р. в конце. Пользователь вводит только число, после нажатия клавиши Enter Access подставляет р. или знак доллара (в зависимости от установленных ранее свойств этого поля). Размер - 4 байта. После запятой возможно 4 знака.

Счётчик. Счетчик - это всегда число. Поля с этим типом выполняют только одну функцию - автоматическую идентификацию записей. Различают два способа задания счетчика: *последовательный* и *случайный.* Поле-счетчик нумерует записи в порядке их ввода автоматически, как только заполнено хоть одно поле записи. Таким образом, пользователю не нужно заполнять данное поле. Последовательный формат счетчика нумерует записи последовательно. 1, 2, 3 и т.д., случайный выбирает коды случайным образом (например, 1193517479 и т.д.) как число типа Длинное целое. Значения этого поля не обновляются. Таблица СУБД Access может содержать 2 млрд. записей.

Дата/время. Этот тип предназначен для работы с датами или временем. Данные этого типа содержат день, месяц и год (тип дата) или часы и минуты (тип время). Допустимы все даты от 01/01/0100 до 31/12/2999. Даты могут быть заданы в различных форматах. Это связано с тем, что в раз личных странах используется разная форма записи дат, различающаяся по рядком следования дней, месяца и года и видом разделителя между этими величинами. Желаемый формат можно определить при помощи свойств поля в бланке конструктора таблицы. В поле данных этого типа можно вводить даты с 100 по 9999.

МЕМО-формат. Этот тип данных используется для описания полей базы данных с текстовой информацией произвольной длины, то есть в тех случаях, когда заранее невозможно определиться с количеством символов. Например, в поле МЕМО можно хранить характеристику человека. В поле МЕМО может находиться до 65535 символов (64 Кб). Особенностью поля МЕМО в том, что реально данные этого поля хранятся в другом месте, а в поле хранится указатель на то, где расположен текст. Поля данного типы не могут быть ключевыми или индексированными.

OLE (Object Linking and Embedding). В ячейки поля данного типа вводятся ссылки на приложения, разрабатываемые

для Windows. Такими объектами могут быть, например, графические файлы (растровые: .bmp, dib, tif и векторные: .wmf), лист таблицы Microsoft Excel (.xls, .dib), документ Microsoft Word, звукозапись (.wav), видеозапись (.avi) или другие типы данных, которые могут быть созданы компонентами ActiveX. Поле объекта OLE не может быть ключевым и его нельзя индексировать. Тип объекта OLE не указывается в свойствах поля. Объём хранимых данных такого типа в ячейках поля ограничен только дисковым пространством компьютера.

Гиперссылка. Значением поля является ссылка на Word Wide Web. Содержит адреса Web-страниц, которые могут быть Web-страницами Internet или локально храниться на персональном компьютере или сети. Может содержать до 2048 символов.

Мастер подстановки. Представляет собой команду для запуска Мастера, который позволяет создать список для выбора соответствующего значения записи из фиксированного списка или таблицы.

Целостность данных

Когда структура таблиц БД созданы, для обеспечения целостности данных необходимо установить связи между таблицами. Целостность данных гарантирует защиту информации от случайных изменений в связанных таблиц.

Целостность данных - это набор правил, которые защищают Вашу информацию от случайных изменений или удалений за счет механизма поддержки корректности связи между связанными таблицами.

Если на связь между таблицами наложены условия ссылочной целостности, то Access не позволит 1) добавлять в связанную таблицу записи, для которых нет соответствующих записей в главной таблице, 2) изменять записи в главной таблице таким образом, что после этого в связанной таблице появятся записи, не имеющие главных записей; 3) удалять записи главной таблицы, для которых имеются подчиненные записи в связанной таблице.

Чтобы условие целостности могло существовать, поле главной таблицы должно быть обязательно ключевым и оба поля должны иметь одинаковый тип. Существует только одно

исключение: если в главной таблице ключевое поле - Счетчик, то в связанной подчиненной таблице тип поля должен быть Длинное целое.

В процессе создания связей между таблицами необходимо включить параметр *Обеспечение целостности данных*, при котором не допускается произвольное удаление или изменение записей в главной таблице.

Если установить параметры связи между таблицами *Каскадное обновление связанных полей и Каскадное удаление связанных полей*, то при любых изменениях данных в главной таблице произойдёт автоматическое изменение связанных данных в подчинённой таблице.

После создания структуры таблиц и установки связей между ними можно перейти к заполнению таблиц данными. Ввод данных производится двумя способами: непосредственно в ячейки таблицы и через формы.

ВОПРОСЫ

1. В каких режимах может быть представлена таблица СУБД Access?
2. Опишите окно конструктора таблицы.
3. Перечислите и охарактеризуйте типы данных?
4. В чём суть ссылочной целостности?

3. Основные свойства полей. Маска ввода. Форматы полей.

Любое поле обладает свойствами. От свойств поля зависит, какие типы данных можно вносить в поле, а какие нет, а также то, что можно делать с данными, содержащимися в поле. Заметим, что список свойств меняется в зависимости от типа поля. Например, поле текстового типа не имеет установок для количества знаков после запятой, а поле типа МЕМО не имеет установок для размеров поля.

Для того, чтобы просмотреть или изменить свойства конкретного поля таблицы, необходимо выбрать это поле. В

нижней части окна бланка конструктора будут отображены *свойства текущего* поля. Каждая строка этого бланка свойств выполняет определённую функцию.

Поля таблиц БД не просто определяют структуру базы данных – они ещё определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Ниже перечислены основные свойства полей таблиц БД на примере СУБД MS Access.

Свойства полей:

1. *Имя поля* – определяет, как следует обращаться к данным этого поля при автоматических операциях с БД (по умолчанию имена полей используются в качестве заголовка столбцов таблиц);
2. *Тип поля* - определяет тип данных, которые могут содержаться в данном поле;
3. *Размер поля* – определяет предельную длину (в символах) данных, которые могут размещаться в данном поле;
4. *Формат поля* - определяет способ форматирования данных в ячейках, принадлежащих полю;
5. *Маска ввода* - определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных);
6. *Подпись* - определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство Имя поля);
7. *Значение по умолчанию* – то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных);
8. *Условие на значение* – ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты);
9. *Сообщение об ошибке* – текстовое сообщение, которое выдается при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство Условие на значение);
10. *Обязательное поле* – свойство, определяющее обязательность заполнения данного поля при наполнении базы;

11. *Пустые строки* – свойство, разрешающее ввод пустых строковых данных (от свойства Обязательное поле отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым);

12. *Индексированное поле* – если поле обладает этим свойством, то все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, индексированное поле можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

Маска ввода

Маска ввода — это *шаблон,* который позволяет вводить в поле значения, имеющие одинаковый формат, и постоянные символы.

Используется для облегчения ввода форматированных данных. Access позволяет задать маску ввода для полей любого типа кроме полей: МЕМО, счетчик, гиперссылка, логический, OLE-объекты. Можно использовать маску ввода для выполнения простых операций (преобразование всех вводимых символов к верхнему регистру) или более сложных (добавление скобок и дефиса в номера телефона). Маска ввода создаётся с помощью специальных символов.

Рассмотрим *набор специальных символов,* которые жестко задают маску ввода в строке свойств "Маска ввода" текстового поля.

0 - Цифра (знаки + и - не разрешены). Ввод *обязателен;*

9 - Цифра или пробел (знаки (+) и (-) не разрешены). Ввод *не обязателен;*

- Цифра или пробел (незаполненные позиции выводятся как пробелы в режиме редактирования, но удаляются при сохранении данных; знаки + и - не разрешены). Ввод *не обязателен;*

L - Любая буква. Ввод *обязателен',*

? - Буква. Ввод *не обязателен;*

А - Буква или цифра. Ввод *обязателен:*

а - Буква или цифра. Ввод *не обязателен;*

& - Любой символ или пробел. Ввод *обязателен;*

С - Любой символ или пробел. Ввод *не обязателен;*

. , : ; - / - разделители, которые сохраняют свой вид в строке данных;

< - преобразование символов в нижний регистр;

> - преобразование символов в верхний регистр;

! - маска должна заполняться справа налево; этот символ следует использовать, если в левой части маски находятся позиции, заполнять которые не обязательно. Маски ввода обычно заполняются слева направо. Символ восклицательного знака можно помещать в произвольную позицию в маске ввода/

\ - ввод следующего за обратной косой чертой символа как символьной константы.

Для создания маски ввода можно пользоваться мастером.

Форматы полей

Свойство "Формат поля", заданное в режиме конструктора таблицы, используется для отображения данных в режиме просмотра таблицы. Эта же настройка применяется при создании связанных с этим полем новых элементов управления в форме или отчете. Выбор стандартного формата осуществляется по кнопке выбора в поле ввода свойства "Формат поля". Для различных типов данных существуют определённые форматы полей, но все-таки их можно разделить на две группы: стандартные и специализированные.

Стандартные форматы.

Для числовых, денежных полей и *поля* "Счетчик": Основной, Денежный, Фиксированный, С разделителями разрядов, Процентный, Экспоненциальный.

Для полей даты/времени: Полный формат, Длинный формат, Средний формат, Краткий формат.

Допустимо как использование встроенных, так и специальных форматов, созданных при помощи символов форматирования. Рассмотрим набор специальных символов формата, которые жестко задают вид и размер вводимых значений поля в строке свойств *"Формат поля":*

Ниже перечислены *специальные символы,* используемые при определении специальных форматов для *любого* типа данных.

Кодовые символы	Значение
(Пробел)	Выводит пробел как символьную константу.
"ABC"	Все символы внутри кавычек считаются символьными константами
1	Выравнивает символы по левому, а не по правому краю.
*	Заполняет доступное пустое пространство следующим за ним символом.
\	Выводит следующий символ как символьную константу. Для этой же цели можно использовать кавычки.
[цвет]	Задает цвет, название которого указано в скобках. Допустимые имена цветов: Черный (Black), Синий (Blue), Зеленый (Green), Бирюзовый (Cyan), Красный (Red), Лиловый (Magenta), Желтый (Yellow), Белый (White).

Замечание: не разрешается смешивать в одном формате специальные символы, предназначенные для определения числовых форматов, форматов даты/времени и текстовых форматов

Перечислим *специальные символы,* используемые при определении специальных форматов для *текстового, числового, денежного, MEMO* полей.

Тип	Кодовые символы

Текстовый или МЕМО	@- должен быть текстовый символ или пробел; & - должен быть текстовый символ; < - преобразование символов в нижний регистр; > - преобразование символов в верхний регистр.
Числовой или денежный	. - в качестве десятичного разделителя; , - как разделитель групп разрядов; 0- вывод цифры или нуля, если разряд незначащий; # - вывод только цифры; $- вывод знака доллара; %- вывод числа в процентном формате; E или e - вывод числа в экспоненциальной форме.

Специальные форматы, задаваемые пользователем, для числовых, денежных, поля "Счетчик". Специальные числовые форматы могут включать в себя от одного до четырех разделов, отделенных друг от друга точкой с запятой (;). Каждый формат содержит спецификацию для различных типов числовых данных.

Первый раздел содержит описание формата положительных чисел.

Второй раздел содержит описание формата отрицательных чисел

Третий раздел содержит описание формата нулевых чисел

Четвертый раздел — описание формата пустых (Null) значений.

Например, возможно использование следующего специального денежного формата: # ##0.00 р.;-# ##0.00 р. [Красный]; 0.00 р.; "Нет данных". Этот формат определяется следующими компонентами, разделяемыми точкой с запятой, которые определяют вывод положительных, отрицательных значений и значений Null. Например, 1234.56 отобразится как 1 234.56 р.; -1234.56 отобразится как -1 234.56р. красного цвета;

0 отобразится как 0.00; значение Null отобразится как Нет данных.

Если описано несколько разделов, но формат каждого из них не указан, в результате либо не будет никакого форматирования, либо будет использовано форматирование первого из них.

Специальные форматы даты и *времени* определяются с помощью следующих символов:

Символ	Описание
•	Разделитель компонентов времени. Символ разделителя выбирается в окне Язык и стандарты панели управления Windows.
/	Разделитель компонентов даты.
c	Задает встроенный "Полный формат даты".
d	Номер дня месяца, состоящий из 1 или 2 цифр (1-
dd	Номер дня месяца, состоящий из 2 цифр (01-31).
ddd	Сокращенное название дня недели (Пн-Вс).
dddd	Полное название дня недели (понедельник-
ddddd	Задает встроенный "Краткий формат даты".
ddddd	Задает встроенный "Длинный формат даты".
\v	Номер дня недели (1-7).
ww	Номер недели в году (1-53).
m	Номер месяца, состоящий из 1 или 2 цифр (1-12).
mm	Номер месяца, состоящий из 2 цифр (01-12).
mmm	Первые три буквы названия месяца (янв-дек).
mmmm	Полное название месяца (Январь-Декабрь),.
q	Номер квартала в году (1-4).
У	Номер дня в году (1-366).
УУ	Последние две цифры номера года (01-99).
УУУУ	Полный номер года (0100-9999).
h	Число часов, состоящее из 1 или 2 цифр (0-23).
hh	Число часов, состоящее из 2 цифр (00-23).

n	Число минут, состоящее из 1 или 2 цифр (0-59).
nn	Число минут, состоящее из 2 цифр (00-59).
s	Число секунд, состоящее из 1 или 2 цифр (0-59). •
ss	Число секунд, состоящее из 2 цифр (00-59).
ttttt	Задает встроенный "Длинный формат времени".
AM/PM	12-часовой формат времени с добавлением прописных букв "AM" или "PM".
am/pm	12-часовой формат времени с добавлением строчных букв "am" или "pm".
A/P	12-часовой формат времени с добавлением прописных букв "A" или "P".
a/p	12-часовой формат времени с добавлением строчных букв "a" или "p".
AMPM	12-часовой формат времени; используется индикатор "утро/день", выбранный в окне Язык и стандарты панели управления Windows.

ВОПРОСЫ

1. Перечислите свойства полей таблиц базы данных.
2. Что такое маска ввода?
3. Расскажите об основных форматах полей?

4. Сортировка, поиск и фильтрация данных

Когда вы открываете таблицу в режиме просмотра, СУБД Access выводит строки в последовательности, определяемой первичным ключом. Если первичный ключ не был определен, вы увидите строки в пос7ледовательности, в которой они были введены в таблицу. Если вы хотите изменить их порядок, Access предоставит вам необходимые средства.

СУБД Access позволяет сортировать (располагать в определённом порядке) поля различных типов данных по возрастанию или по убыванию значений, а также имеется возможность сортировать два поля в одной таблице одновременно. Пользователь может сортировать записи в любом поле, - это значит, что порядок записей будет устанавливаться в соответствии со значением величин в этом поле.

Сортировка записей (по полю) — это изменение порядка следования записей в соответствии со значениями данных в этом поле.

Сортировка (упорядочение) данных по возрастанию означает, что значения в поле отсортированного *текста* располагаются по алфавиту (от А до Я), отсортированные *числовые* значения ищут от меньшего к *большему,* а отсортированные поля *дат/времени* располагаются по увеличению даты и. времени от годов до н.э. к годам н.э.

Сортировка (упорядочение) данных по убыванию означает обратное.

Сортировка выполняется при помощи команд меню или пиктограмм на панели инструментов.

Поиск данных

При работе с информацией, как правило, возникает необходимость в поиске или замене данных. Стандартное диалоговое окно *"Найти" или "Заменить"* предоставляет несколько вариантов поиска и замены. Можно организовать поиск с учётом регистров символов или формата полей. Например, если Вы зададите образец поиска "Ира" с учетом регистров символов, Access не обнаружит "ира" и "ИРА". Формат полей важен при поиске, например, по дате.

Также можно задать направление поиска: вниз (от текущей до конца), вверх (от текущей до начала), все.

Существует возможность влиять на область поиска: либо в текущем поле, либо во всей таблице. В окне "Найти" или "Заменить" можно выбрать один их следующих типов совпадения с образцом:

1) *С любой части поля* - совпадения с образцом ищутся в любой части содержимого поля. Например, для образца "Ира" будут найдены "Кира", и "Ираида", и "Ира".

2) *Поля целиком* - будут обнаружены поля, содержание которых полностью совпадает с образцом.

3) С *начала поля* - будут обнаружены поля, начало содержимого которых полностью совпадает с образцом. Например, для образца "Ира" будут найдены "Ираида", но не найдены "Кира".

В некоторых случаях не возможно точно сформулировать требование поиска, например, фамилия студента то ли Зозулин, то ли Зазулин, а может - Зизулин. В Microsoft Access предусмотрены средства для поиска в строках символов при помощи шаблонов поиска. Рассмотрим основные шаблоны для поиска в тестовых полях:

Символ шаблона	Шаблон используется	Шаблон поиска	Результаты поиска
*	*произвольного* количества *алфавитно-цифровых символов* в произвольном месте значения.	*ров *Мар*	Ров; Петров; Комаров; коров; ул. 7 костров. Ул. 8 Марта, 45-23; ул. Маршака, 180а -1; Маркин.
?	любого *одиночного алфавитно-цифрового символа*	м?р Т??я	мур; мир; мор; мэр; Толя: Таня; Тоня; То1я, ! Тося, Тася.
#	любой *одиночной цифры* в произвольном месте значения	23#45 52-5#-4#	123545, 23145, 23745, 23945 52-55-46, 52-59-41
[]	*любого из перечисленных* в скобках символов	З[оан]лин	Зозулин, Зазулин, Зизулин.
!	любого (одного) символа, кроме указанных в скобках	ко[!рс]а	коза; кожа; кома НО не найдет кора, коса.
-	любого одного символа из указанного	ко[к-м]а	кола; кома

Замечание. Если потребуется найти символы # или ? или * (совпадающие с шаблонами), то их следует заключить в квадратные скобки. Например, чтобы найти: "Где ты?", следует задать: Где ты[?]

Фильтрация данных

Фильтр — это набор условий для отбора записей или их сортировки.

Microsoft Access поддерживает три разновидности фильтров:

1. *Фильтр по выделенному.* Критерий отбора записей устанавливается путем выделения всего значения поля таблицы или его части. Недостаток такой разновидности фильтров — отбор можно производить по значению только одного поля. Используется чтобы вывести только записи с определенным значением одного или нескольких полей.

2. *Обычный фильтр.* Фильтр по выделенному используется, если нужно отобрать записи, удовлетворяющие всем условиям, т. е. условия объединяются по И. Если нужны записи, удовлетворяющие совокупности условий, объединённых и по **И,** и по **ИЛИ,** нужно применять обычный фильтр.

Критерии отбора указываются в форме, при этом можно задавать критерии отбора по каждому из полей таблицы.

3. *Расширенный фильтр.* В окне расширенного фильтра можно указывать как критерии отбора для различных полей, так и порядок их сортировки.

Фильтр действует в рамках конкретного объекта: таблицы, формы или запроса.

ВОПРОСЫ

1.Что означает сортировать записи в любом поле?
2. Как организовать поиск данных?
3. Что такое фильтр? Назовите виды фильтров.

5. **Формы**

Понятие, назначение и виды форм.

Форма содержит те же поля, что и запись таблицы, поэтому многие предпочитают использовать формы для ввода и просмотра данных, так как при этом легче сосредоточиться на конкретной информации. Форма позволяет отобразить информацию из одной записи наглядно как бы на отдельной карточке. Такое представление информации облегчает её восприятие. Форма похожа

на обычный бланк с полями, которые пользователь должен заполнить. Access связывает форму с таблицей и хранит в таблице введенные пользователем в форме данные.

Существует возможность использовать одну форму для ввода данных одновременно в несколько связанных таблиц.

Форма - это объект базы данных, который предназначен для ввода или просмотра данных на экране. Другими словами, *форма* - средство отображения данных на экране и управления ими.

Формы являются основным средством организации интерфейса пользователя.

Формы можно использовать:

1) для ввода и редактирования данных записи. Это самый распространенный способ использования форм. Упрощается изменение, добавление и удаление данных.

2) Только для ввода данных. Форма открывается только на заполнение данных.

3) Для вывода сообщений.

4) Для управления приложением, т.е. автоматизировать вывод определённых данных или выполнение некоторой последовательности действий. В форме можно разместить: меню; командные кнопки, например, вызывающие макрос, функцию, открывающие запрос, другую форму или отчет.

5) Для вывода данных на печать.

6) для отображения данных в виде диаграммы.

Формы бывают следующих видов:

1. Автоформа – создаются при помощи мастера: в столбец, ленточная, табличная и т.д.

2. Составная форма - содержит данные из двух различных таблиц,
между которыми установлена связь.

3. Подчиненная форма – это форма, находящаяся внутри другой формы.

4. Диаграмма.

5. Модальная форма – формы для вывода сообщений или запроса информации от пользователя.

6. Всплывающие формы – формы, предназначенные для отображения информации, которую постоянно необходимо

отображать на переднем плане даже если форма не находится в фокусе.

Формы создаются при помощи мастера и конструктора. Любую форму можно модифицировать в режиме конструктора.

Области и свойства форм.

Для повышения функциональности формы её рабочая область может быть разбита на несколько областей. В режиме конструктора форма состоит из следующих областей:

1. *Область заголовка* формы. В области заголовка формы расположены название формы и ряд других элементов, содержимое которых не изменяется при переходе от одной записи к другой.

2. *Область данных.* В области данных выводятся все данных из базовой таблицы или запроса.

3. *Верхний колонтитул* формы. Используется для отображения заголовка формы, заголовков столбцов или других сведений, которые требуется напечатать сверху на каждой странице формы. Выводится только при печати формы.

4. *Нижний колонтитул* формы. Используется для отображения дат, номеров страниц или других сведений, которые требуется напечатать снизу на каждой странице формы. Выводится только при печати формы.

5. *Область примечаний* формы. Здесь располагаются элементы, содержимое которых постоянно.

Первый шаг в проектировании формы – установить её общие свойства. Это можно сделать как во время разработки – в окне свойств, так и в период выполнения через программный код. Многие свойства формы воздействуют на её физическое представление. Самый лучший способ изучения свойств формы – экспериментировать. Изменяя значения свойств формы в окне свойств и затем запуская её в режиме формы, можно почувствовать эффект изменений.

Форма, любая область формы, каждый элемент управления обладают свойствами. Для того, чтобы открыть окно свойств (просмотреть или изменить эти свойства), необходимо выбрать нужный элемент и щелкнуть правой кнопкой мыши. В контекстно-зависимом меню выбрать пункт "Свойство". Перечислим основные

свойства формы:

Свойство	Описание
Подпись	Выводится в строке заголовка формы.
Источник записей	Указывается имя таблицы, запроса, подчиненной формы.
Режим по умолчанию	Определяет тип режима формы при открытии (простая, ленточная, табличная форма).
Допустимые режимы	Определяет, можно ли пользователю переключаться между режимами (форма, таблица и т.д.).
Разрешить изменения	Запрещает или разрешает редактировать данные в форме.
Разрешить удаление	Запрещает или разрешает удалять записи в форме.
Разрешить добавление	Определяет возможность добавления новых записей.
Ввод данных	Определяет, будет ли выводить форма сохраненные записи (при значении до имеющейся записи ранее сохраненные данные не выводятся, при открытии формы отображается бланк новой записи).
Блокировка записей	Определяет блокировку данных при работе нескольких пользователей.
Полосы прокрутки	Определяет, будут ли выводиться на экран полосы прокрутки.
Поле номера записи	Определяет наличие или отсутствие в форме кнопок перехода и поля с номером текущей записи.

Горизонтальные линии	Это свойство определяет будут ли видны линии между областями формы.
Автоматический размер	Размер окна формы позволяет отобразить содержимое всех полей записи.
Выравнивание по центру	При открытии форма размещается по центру экрана.
Всплывающее окно	Форма отображается на переднем плане. *
Тип границы	Определяет тип границы формы.
Кнопка оконного меню	Определяет доступно ли оконное меню.
Кнопка закрытия	Определяет наличие кнопки закрытия в правом верхнем углу.
Ширина	Показывает значение ширины формы.
Переход по TAB	Указывает порядок перехода от одного поля к другому, осуществляемый по нажатию кнопки TAB. Начинается с 0.
Рисунок	Содержит имя файла растрового изображения для фона всей формы. В качестве фонового рисунка можно использовать любой графический файл в формате .wmf, .emf, .dib, .bmp.
Строка меню	Используется для определения строки меню.
Наличие модуля	Наличие или отсутствие модуля класса для формы определяется значением свойства «Наличие модуля». Значение «Нет» говорит о том, что в форму нет необходимости добавлять какой-либо модуль (код VBA). Если установлено значение «Да», то форма

	имеет модуль формы.

Типы элементов управления.

Интерфейс приложения может состоять из одной из одной или нескольких форм с размещёнными на них элементами управления.

Элемент управления - графический объект, который предназначен для отображения данных (например, надпись или поле ввода), выполнения операций (например, кнопка открытия формы) или является оформлением формы (линии, прямоугольники).

Элементы управления могут быть *связанными, свободными* или *вычисляемыми.*

1. Связанный элемент управления присоединен к полю базовой таблицы или запроса. Такие элементы управления используются для отображения, ввода или обновления значений из полей базы данных.

2. Вычисляемый элемент. Для вычисляемого элемента управления в качестве источника данных используется выражение. В выражении могут быть использованы данные из поля базовой таблицы или запроса для формы или отчета, а также данные другого элемента управления формы или отчета.

3. Для свободного элемента управления источника данных не существует. Свободные элементы управления используются для вывода на экран данных, линий, прямоугольников и рисунков.

В Microsoft Access существуют следующие типы элементов управления (Рисунок 7):

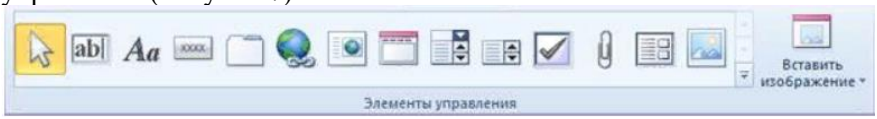


Рисунок 7 - Элементы управления

1. *Надпись*. Надписи предназначены для отображения в форме или отчете описательных текстов, таких как заголовки, подписи или краткие инструкции. В надписях не выводятся значения полей или выражений; они всегда являются свободными и

не меняются при переходе от записи к записи. Надпись может быть присоединена к другому элементу управления (такую надпись называют подписью). Например, поле ввода создается с присоединенной надписью, которая содержит подпись этого поля.

2. *Поле ввода.* Поле ввода - это поле, в котором задается или отображается значение поля. При создании поля ввода вместе с ним создается присоединенная надпись. Поле можно использовать для вывода данных на экран в форме или отчете, для вывода значения выражения.

3. *Группа переключателей* (Рисунок 8):

- *Переключатели.* Представляет собой набор надписей, помеченных кружочками. Знаком © изображается действующий (выбранный) режим. При выборе другого режима переключателя (надписи) точка переместится в позицию рядом с выбранным названием. Переключатели не допускают множественный выбор.

- *Флажки.* Если рядом с соответствующей надписью размещен пустой (П) или перечеркнутый квадратик (0 или D*3), то эта надпись будет соответствовать полю типа переключатель. Знак 0 или DD означает надпись выбрана. В отличие от переключателя флажки допускают множественный выбор.

- *Выключатели.* Изображается в виде кнопки. Изображение "кнопка нажата (вжата)" означает режим выбран (включен).

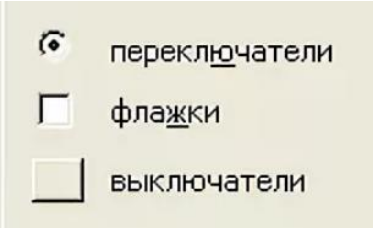


Рисунок 8 - Группа переключателей

4. Списки:

- *список.* Содержит фиксированный набор значений или значения из заданного поля одной из таблиц. Позволяет не вводить данные, а выбирать из списка (Рисунок 9).

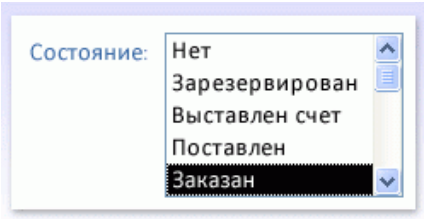


Рисунок 9 - Список

- *поле со списком.* Применяется, так же как и список, но занимает меньше места в форме, поскольку список раскрывается только после щелчка на раскрывающей кнопке (Рисунок 10).

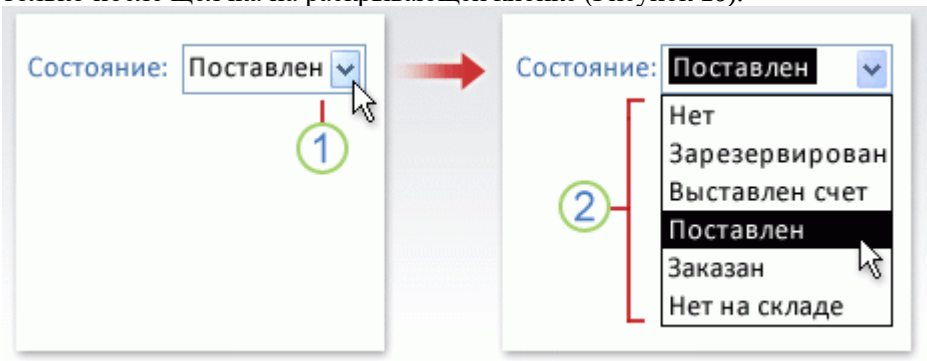


Рисунок 10 - Поле со списком

5. *Рамки объектов.* Служит для размещения внешнего объекта: иллюстрации, фотографии и т.д.

- *Присоединенная рамка объекта.* С рамкой связано определенное поле, например, поле OLE, таблицы. В ней отображается содержимое этого поля.

- *Свободная рамка объекта.* Рамка, не связана ни с каким полем таблицы. Объект, находящийся в ней, исполняет роль иллюстрации и служит для оформления формы.

6. *Командные кнопки.* С каждой из них можно связывать выполнение команды. Кнопку можно использовать для открытия другой формы или отчета, или для запуска макроса и т.д.

7. *Набор вкладок.* Позволяют поместить большой объем информации на ограниченной площади (несколько страничек).

8. *Подчиненная форма/отчет.* Подчиненная форма - это форма, находящаяся внутри другой формы. Первичная форма называется главной (родительской) формой, а форма внутри формы называется подчиненной (дочерней) формой. Подчиненная форма удобна для вывода данных из таблиц или запросов, связанных с отношением "один-ко-многим". Главная форма и подчиненная форма в этом типе форм связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме. Например, когда главная форма отображает данные о конкретном студенте, подчиненная форма отображает только работы этого студента. При входе в подчиненную форму для ввода новых записей текущая запись в главной форме сохраняется.

Это гарантирует, что: 1) записи из таблицы на стороне "многие" будут иметь связанную запись в таблице на стороне "один".; 2) автоматически сохраняется каждая запись, добавляемая в подчиненную форму. Подчиненная форма может быть выведена в режиме таблицы, или как простая, или как ленточная форма. Главная форма может быть выведена только как простая форма. Главная форма может содержать любое число подчиненных форм, если каждая подчиненная форма помещается в главную форму. Имеется также возможность создавать подчиненные формы двух уровней вложенности. Это означает, что можно иметь подчиненную форму внутри главной формы, а другую подчиненную форму внутри этой подчиненной формы. Например, можно иметь главную форму, в которой выводятся данные о клиентах, подчиненную форму, выводящую данные о заказах и другую подчиненную форму, которая отображает то, что заказано.

9. *Линия, прямоугольник и дополнительные элементы ActiveX*. Линию или прямоугольник можно использовать для разделения и группировки элементов управления с тем, чтобы они лучше воспринимались пользователем.

10.*Разрыв страницы.*

В Microsoft Access создать элемент управления и задать свойства элемента можно в режиме конструктора формы или режиме конструктора отчета. Кнопки для создания элементов управления содержатся на панели элементов.

ВОПРОСЫ

1. С помощью чего создается интерфейс пользователя?
2. Что такое форма?
3. Для чего используются формы?
4. Расскажите об разделах формы?
5. Перечислите основные виды формы и расскажите об их предназначении?
6. Что такое элемент управления?

6. Отчеты

Понятие и назначение отчетов.

Отчёты являются компонентами интерфейса приложения, во многом схожи с формами. Они позволяют предварительно просмотреть и распечатать информацию, полученную из объектов данных. При проектировании отчётов используются многие технологии, применяемые для форм, включая процедуры связывания с источником записей и работу с элементами управления.

Отчет - это объект базы данных, который предназначен для печати данных. Другими словами, отчет - средство отображения данных при выводе на печать.

Отчёты представляют наилучшее средство создания печатных форм документов. Отчёты обладают следующими преимуществами:

1. Предоставляют возможность для группировки и вычисления промежуточных и общих итогов для больших наборов данных.

2. Отчёты, могут бать использованы для получения красиво оформленных счетов, заказов на покупку, почтовых наклеек, материалов для презентаций и других документов, которые могут понадобиться для ведения бизнеса.

Отчет, так же как и форма, состоит из нескольких областей (Рисунок 11):

1. *Область заголовка отчета.* В области заголовка отчета

расположено название отчета и ряд других элементов, содержимое которых не изменяется

2. *Область данных*. В области данных выводятся все данных из базовой таблицы или запроса.

3. *Области верхнего и нижнего колонтитулов*. В них можно расположить подзаголовки, колонцифры {номера страниц).

4. *Область примечаний отчета*. Здесь располагается дополнительная информация, содержимое которой постоянно.

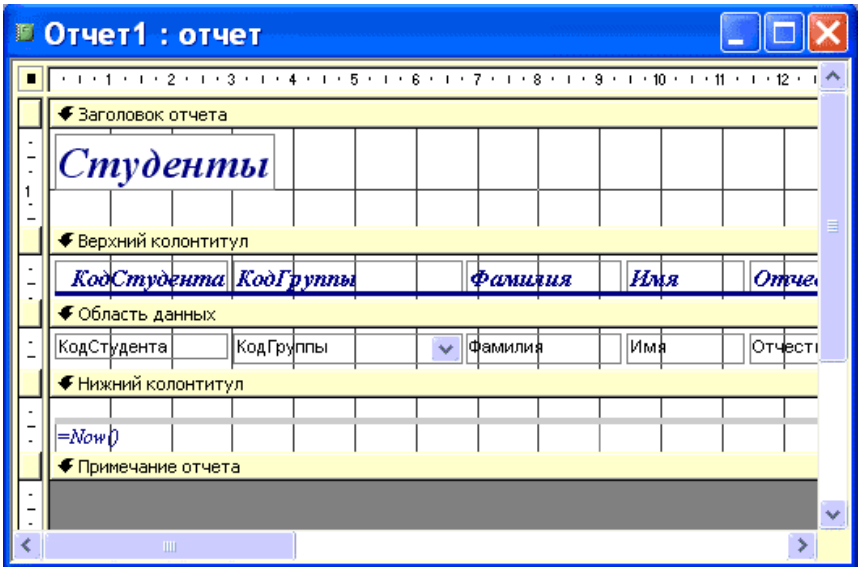


Рисунок 11 – Области отчета

Любая область отчёта, каждый элемент управления обладают свойствами, которые вызываются при помощи контекстного меню.

Отчеты во многом похожи на формы. В Access можно создать несколько видов отчетов, причем создание отчета не отличается от создания формы. Итак, отчеты бывают следующих видов:

1. Автоотчёты:

- «в столбец» На экран выводится в один столбец все данные из всех записей таблицы.

- ленточный отчет. Каждая запись занимает отдельную строку - ленту.

- табличный отчет.

2. Групповые/итоговые отчеты.

3. Почтовая наклейка.

4. Слияние с Microsoft Word.

5. Диаграммы.

Способы создания отчетов.

Отчёты создаются с помощью мастера и конструктора. В режиме конструктора отчёт создается при помощи элементов управления. Если отчёт создаётся с помощью мастера, то умолчанию задаются стандартные характеристики отчёта, определяемые используемым шаблоном.

Элементы управления отчётов не привязаны, к каким либо событиям в отличие от элементов управления форм.

Так же, как и в форме, вы можете вставлять в любой раздел отчёта рисунки или диаграммы. В отчёт также можно внедрять подчинённые отчёты или подчинённые формы.

В отчётах можно производить сортировку, группировку данных, определять итоговые значения и изменять внешний вид.

Группировку записей можно производить по:

1. Текстовым значениям.
2. Значениям даты и времени.
3. Значениям счётчика, типов данных денежный или числовой.

Группировка и сортировка записей отчёта производится при помощи свойств или пиктограмм на панели инструментов. В отчёте можно создавать до 10 уровней группировки.

Итоговые значения подсчитывают при помощи выражений, которые создаются построителем.

Точно так же, как в основные формы можно внедрять подчинённые, в основной отчёт можно внедрять подчиненные отчёты. Иногда удобнее подсчитать в отчёте итоговые значения и включать результаты вычислений в другой отчёт, содержащий детальную информацию.

Подчинённый отчёт – это отчёт, вставляемый в другой отчёт.

Подчинённый отчёт должен быть связан с главным отчётом. Это связь устанавливается автоматически, если отчёты создаются на основе связанных таблиц. Если главный и подчинённый отчёты не связаны, то связь между ними устанавливается при помощи свойств.

ВОПРОСЫ

1. Что такое отчеты и каково их предназначение?
2. Какие отчёты и формы называют подчинёнными?
3. С помощью чего создаются отчёты и формы?

7. Запросы. Технология разработки запросов.

Запрос (query) – это средство выбора необходимой информации из базы данных.

SQL-запросы – это запросы, которые составляются (программистами) из последовательности SQL-инструкций. Эти инструкции задают, что надо сделать с входным набором данных для генерации выходного набора. Все запросы Access строит на основе SQL-запросов, чтобы посмотреть их, необходимо в активном окне проектирования запроса выполнить команду Вид/SQL.

Существует несколько типов запросов: на выборку, на обновление, на добавление, на удаление, перекрестный запрос, создание таблиц. Наиболее распространенным является запрос на выборку. Он создается только для связанных таблиц.

Создание запроса на выборку с помощью Мастера

При создании запроса необходимо определить:

- Поля в базе данных, по которым будет идти поиск информации
- Предмет поиска в базе данных
- Перечень полей в результате выполнения запроса

В окне база данных выбрать вкладку Запросы и дважды щелкнуть на пиктограмме Создание запрос с помощью мастера, появится окно Создание простых запросов.

В окне мастера выбрать необходимую таблицу (таблицу - источник) из опции Таблицы и запросы и выбрать поля данных (в области Доступные поля выделить поле и нажать кнопку >, поле перенесется в область Выбранные поля). Если запрос формируется на основе нескольких таблиц, необходимо повторить действия для каждой таблицы – источника.

Затем в окне Мастера надо выбрать подробный или итоговый отчет и щелкнуть на кнопке Далее. После этого необходимо задать имя запроса и выбрать один из вариантов дальнейшего действия: Открыть запрос для просмотра данных или Изменить макет запроса и нажать кнопку Готово. В результате чего получите готовый запрос.

Создание запроса на выборку с помощью Конструктора

С помощью конструктора можно создать следующие виды запросов:

- Простой
- По условию
- Параметрические
- Итоговые
- С вычисляемыми полями

В окне база данных необходимо выбрать вкладку Запросы и дважды щелкнуть на пиктограмме Создание запроса в режиме конструктора. В окне Добавление таблицы следует выбрать таблицу – источник или несколько таблиц из представленного списка таблиц, на основе которых будет проводиться выбор данных, и щелкнуть на кнопке Добавить.

Окно Конструктора (Рисунок 12) состоит из двух частей – верхней и нижней. В верхней части окна размещается схема данных запроса, которая содержит список таблиц – источников и отражает связь между ними.

В нижней части окна находится Бланк построения запроса QBE (Query by Example), в котором каждая строка выполняет определенную функцию:

- Поле – указывает имена полей, которые участвуют в запросе
- Имя таблицы – имя таблицы, с которой выбрано это поле
- Сортировка – указывает тип сортировки

- Вывод на экран – устанавливает флажок просмотра поля на экране
- Условия отбора - задаются критерии поиска
- Или – задаются дополнительные критерии отбора

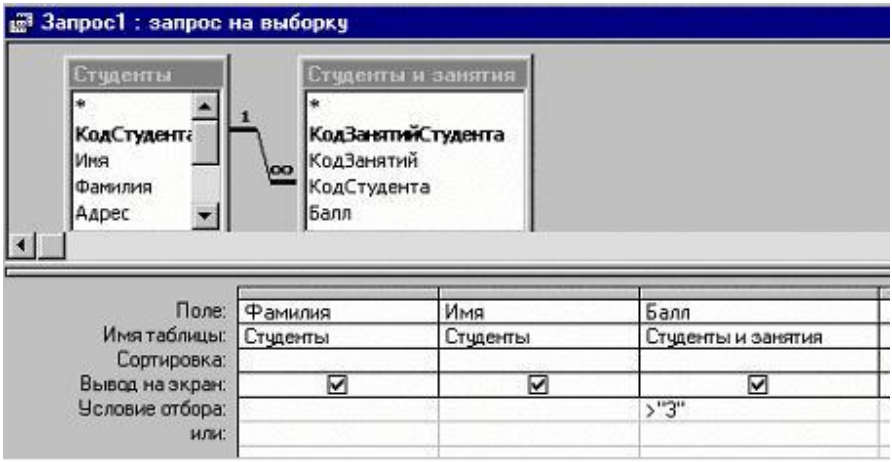


Рисунок 12 - Окно Конструктора запроса

В окне «Запрос: запрос на выборку» с помощью инструментов формируем запрос:

- Выбрать таблицу – источник, из которой производится выборка записей.
- Переместить имена полей с источника в Бланк запроса.
- Задать принцип сортировки.
- В строке вывод на экран автоматически устанавливается флажок просмотра найденной информации в поле.
- В строке "Условия отбора" и в строке "Или" необходимо ввести условия ограниченного поиска – критерии поиска.

После завершения формирования запроса закрыть окно Запрос на выборку, ввести имя созданного запроса, и щелкнуть ОК и вернуться в окно базы данных.

Чтобы открыть запрос из окна базы данных, необходимо выделить имя запроса и щелкнуть кнопку Открыть, на экране появится окно запрос на выборку с требуемым именем.

Чтобы внести изменения в запрос его необходимо выбрать щелчком мыши в окне базы данных, выполнить щелчок по кнопке Конструктор, внести изменения. Сохранить запрос, повторить его выполнение.

ВОПРОСЫ

1. Назовите назначение и виды запросов, разрабатываемых в СУБД ACCESS.
2. В чем состоит отличие постоянного запроса от параметрического?
3. Каково назначение перекрестного запроса?
4. Назовите типы запросов по выполняемым действиям.
5. Назовите правила ввода условий отбора данных в текстовые поля.
6. В чем состоит различие между условиями отбора данных, связанных отношениями AND и OR?

8. Макросы

Понятие и назначение макросов. Управление объектами баз данных с помощью макросов.

Макрос – набор из одной или более команд (макрокоманд), выполняющих определённые операции.

Макросы применяются для автоматизации часто используемых простых задач, таких как открытее и закрытее форм, вывод на экран или печать отчётов при нажатии пользователем кнопки. Действия, связывающие различные объекты БД выполняются очень легко, поскольку пользователь не должен запоминать правила синтаксиса – все аргументы, требуемые каждой макрокомандой, предоставляются в панели аргументов окна макроса. Для реализации более сложных задач следует использовать программы, разработанные при помощи Visual Basic.

Макрос может состоять из одной макрокоманды и из последовательности макрокоманд.

При наличии большого числа макросов, объединение

родственных макросов в группы может упростить управление базой данных.

Группой макросов называют набор макросов, сохранённых под общим именем и объеденных по смыслу.

Все макрокоманды выполняются каждый раз при запуске макроса в порядке их следования.

Макросы применяются для автоматизации часто повторяющихся действий. Макрокоманды в СУБД Access по назначению можно разделить на следующие классы:

1. Открытие и закрытее таблиц, запросов, форм, отчётов;
2. Печать данных;
3. Выполнение запросов;
4. Проверка истинности условий и управления выполнением макрокоманд;
5. Установка значений;
6. Поиск данных;
7. Построение пользовательского меню и выполнение команд меню;
8. Управление выводом информации на экран;
9. Сообщение пользователю о выполняемых действиях;
10. Переименование, копирование, удаление, импорт и экспорт объектов;
11. Запуск других приложений Windows.

Создание макросов

Макросы создаются при помощи конструктора (Рисунок 13). Окно конструктора можно условно разделить на две части:

- Верхняя часть представляет собой таблицу, в которой отражается имя макроса, макрокоманды, условие и описание.

- В нижней части отражаются аргументы. Аргументы предоставляют дополнительную информацию о выполнении макрокоманды, например, какой объект или данные нужно использовать.

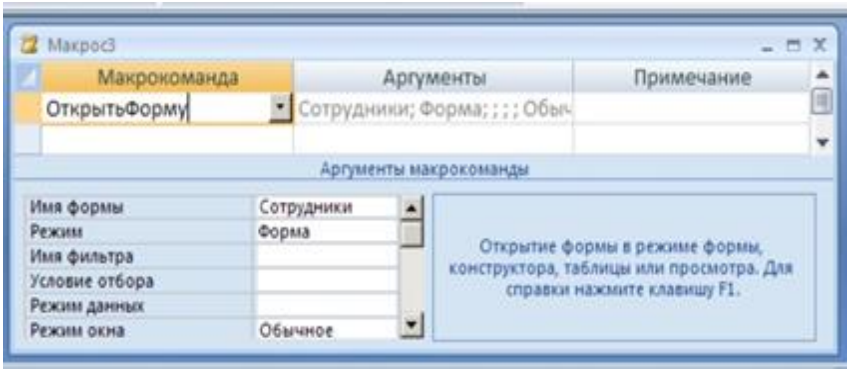


Рисунок 13 – Конструктор макроса

В некоторых случаях требуется выполнить макрокоманду или серию макрокоманд только при выполнении некоторых условий. Например, если в макросе проверяется соответствие данных в форме условиям на значение, то для одних значений может потребоваться вывести одно сообщение, а для других значений другое сообщение. В подобных случаях условия позволяют определить порядок передачи управления между макрокомандами в макросе.

Условие задаётся с помощью логического выражения. Условия в макросах создаются при помощи построителя выражений в конструктора макроса. В зависимости от значения логического выражения, управление передаётся разным макрокомандам. При запуске макроса проверяется значение каждого условного выражения. Если условие истинно, выполняется макрокоманда, содержащаяся в данной строке.

Выполнение макросов

Макрос может состоять из серии макрокоманд. Эти макрокоманды выполняются каждый раз при запуске макроса в порядке их записи.

Выполнение макрокоманд макроса начинается с первой в порядке следования и продолжается до конца списка макрокоманд макроса. Если макрос входит в группу макросов, то выполнение его макрокоманд продолжается до начала следующего макроса.

Выполнение макроса может начинаться по команде пользователя, при вызове из другого макроса или процедуры

обработки события, а так же в ответ на событие в форме, отчёте или элементе управления. Например, можно привязать макрос к кнопке в форме, в результате чего макрос будет запускаться при нажатии кнопки. Допускается также создание специальной команды меню или кнопки панели инструментов, запускающей макрос, определение сочетания клавиш, нажатие которых запускает макрос, а также автоматический запуск макроса при открытии БД.

Макросы часто связывают с событиями базы данных. Microsoft Access реагирует на события различных типов, возникающие в формах, отчетах или элементах управления. Например, на нажатия кнопок мыши, изменение данных, а также на открытие или закрытие формы или отчета.

События в Access можно разделить по следующим категориям:

1. События, связанные с клавиатурой и мышью.
2. События данных: до обновления, после обновления, удаление, изменение и т.п.
3. События управления объектами: применение фильтра и т.п.
4. События печати: форматирование печатной формы, страница – просмотр печатной формы, печать и т.п.
5. Другие.

Access позволяет привязывать макрокоманды к клавишам клавиатуры. Для этого необходимо создать макрос или группу макросов и сохранить под именем AutoKeys. При этом в имени макроса следует указать сочетание клавиш, при нажатии на которые будет выполняться макрос.

Если макрокоманда или набор макрокоманд связывается с сочетанием клавиш, которые уже используются в Microsoft Access (Ctrl+C – команда копировать), то новое значение макрокоманд на это сочетание клавиш переопределит стандартное назначение команд Microsoft Access.

Допускается выполнение макроса при открытии БД. Например, открытее главной формы. Для создания такого макроса необходимо определить набор макрокоманд, которые требуется выполнить при открытии БД и сохранить макрос под именем AutoExec.

ВОПРОСЫ

1. Что такое макрос, и для каких задач управления базами данных его разрабатывают?
2. Перечислите классы макрокоманд.
3. Опишите окно конструктора макросов.
4. Перечислите категории событий в Microsoft Access.

9. Разработка меню пользователя.

Главная кнопочная форма (Рисунок 14) создается с целью навигации по базе данных. Эта форма может использоваться в качестве главного меню БД. Элементами главной кнопочной формы являются объекты форм и отчётов.

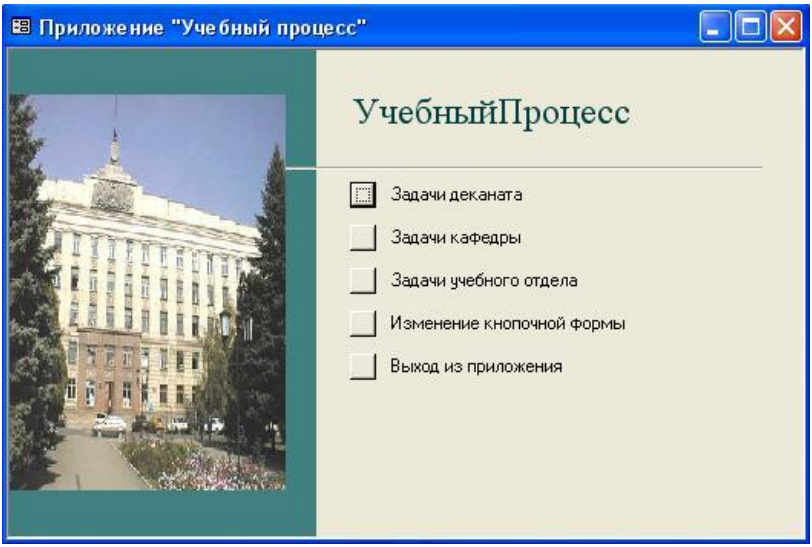


Рисунок 14 – Пример главной кнопочной формы

Запросы и таблицы не являются элементами главной кнопочной формы. Поэтому для создания кнопок Запросы или Таблицы на кнопочной форме можно использовать макросы. Сначала в окне базы данных создают макросы «Открыть Запрос»

или «Открыть Таблицу» с уникальными именами, а затем в кнопочной форме создают кнопки для вызова этих макросов.

Для одной базы данных можно создать несколько кнопочных форм. Кнопки следует группировать на страницах кнопочной формы таким образом, чтобы пользователю было понятно, в каких кнопочных формах можно выполнять определенные команды (запросы, отчеты, ввода и редактирования данных). Необходимо отметить, что на подчиненных кнопочных формах должны быть помещены кнопки возврата в главную кнопочную форму.

Технология создания кнопочных форм следующая:

- создать страницу главной кнопочной формы (ГКФ);
- создать необходимое количество страниц подчиненных кнопочных форм (например, формы для ввода данных, для отчетов, для запросов и т.д.);
- создать элементы главной кнопочной формы;
- создать элементы для кнопочных форм отчетов и форм ввода или изменения данных;
- создать макросы для запросов или для таблиц с уникальными именами;
- создать элементы для кнопочных форм запросов или таблиц.

Настройка параметров запуска

Существует два способа задать действия, выполняемые приложением при запуске, и ряд параметров, влияющих на работу приложения: установка параметров запуска в специальном диалоговом окне и создание макроса "AutoExec". Создание и назначение макроса "AutoExec" уже рассматривалось.

В данной лекции будем рассматривать установку параметров запуска. Для этого выберите команду **Сервис, Параметры запуска**. На экране появляется диалоговое окно *Параметры запуска* (Рисунок 15).

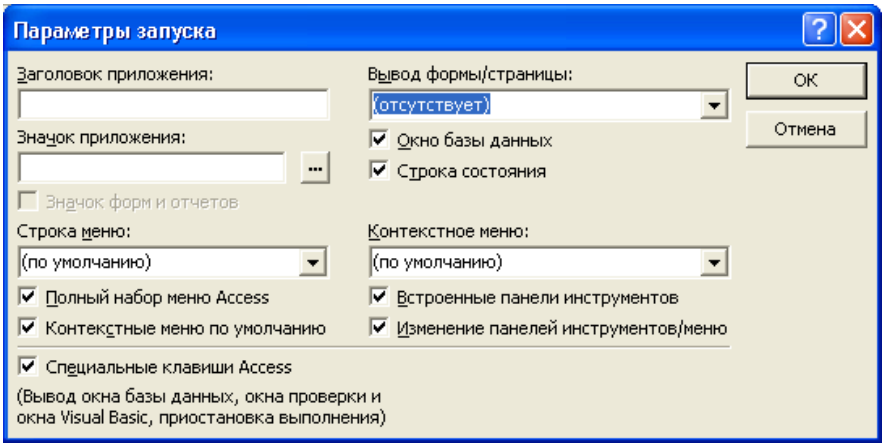


Рисунок 15 - Диалоговое окно *Параметры запуска*

В диалоговом окне *Параметры запуска* можно задать:

- значок и заголовок вашего приложения, которые будут отображаться в главном окне приложения и на панели задач вместо стандартного значка и заголовка "Microsoft Access". Значок должен быть задан в файле с расширением ico, а выбрать данный файл можно с помощью кнопки Построителя. Этот же значок может быть использован во всех формах и отчетах, для чего необходимо установить флажок *Значок форм и отчетов*. Эти параметры вступают в силу сразу после закрытия диалогового окна *Параметры запуска*;
- меню, которое будет появляться при запуске вместо стандартного меню Access и определять основные функции приложения;
- форму или страницу, которая будет появляться на экране при открытии базы данных;
- специальное контекстное меню, которое будет заменять встроенные контекстные меню во всех окнах приложения, кроме тех, с которыми связаны другие контекстные меню.

Флажки *Окно базы данных* и *Строка состояния* позволяют скрывать при запуске окно базы данных и строку состояния.

Следующая группа флажков позволяет запретить пользователям вносить изменения в разработанное приложение.

Пока вы разрабатываете приложение, все эти флажки установлены, но когда вы будете передавать его пользователям, целесообразно сбросить флажки *Полный набор меню* **Access**, *Встроенные панели инструментов* и *Изменение панелей инструментов/меню*.

ВОПРОСЫ

1. Назначение главной кнопочной формы.
2. Опишите технологию создания кнопочных форм
3. Перечислите, что можно задать в диалоговом окне *Параметры запуска.*

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Основы проектирования баз данных : Учебное пособие / О. Л. Голицына, Т. Л. Партыка, И. И. Попов. - 2-е изд., перераб. и доп.. - М. : Форум, 2012. - 416 с

2. Основы проектирования баз данных : Учебное пособие для студ. учреждений сред. проф. образования / Г. Н. Федорова. - 2-е изд., стер.- М.: Издательский центр "Академия", 2016. - 224 с

Дополнительная литература:

1. Информатика, автоматизированные информационные технологии и системы: учебник / В.А. Гвоздева . - М. : ИД "Форум": ИНФРА-М, 2014. - 544 с.

Интернет-ресурсы:

1 Интернет Университета информационных технологий: http://www.intuit.ru