Базы ДАННЫХ

Издательство ТГТУ

Министерство образования и науки Российской Федерации

ГОУ ВПО «Тамбовский государственный технический университет»

БАЗЫ ДАННЫХ

Лекции к курсу для студентов 2, 3 курсов
специальностей 030501 «Юриспруденция», 010502 «Прикладная
информатика в юриспруденции» всех форм обучения

Тамбов
Издательство ТГТУ
2007

УДК 004.65
ББК 973-018.3я73-5
С956

Р е ц е н з е н т:

Доктор технических наук, профессор
П.С. Беляев

С о с т а в и т е л и:
Э.В. Сысоев,
Е.В. Бурцева

С956 Базы данных : лекции к курсу / сост. : Э.В. Сысоев, Е.В. Бурцева. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. – 48 с. – 70 экз.

Изложены основные положения теории баз данных и этапы их создания с помощью системы управления Microsoft Access.
Предназначены для студентов вузов, аспирантов, преподавателей высших учебных заведений, имеющих дело с базами данных.

УДК 004.65
ББК 973-018.3я73-5

Учебное издание

БАЗЫ ДАННЫХ

Лекции к курсу

Составители:
СЫСОЕВ Эдуард Вячеславович,
БУРЦЕВА Елена Васильевна

Редактор О.М. Ярцева
Компьютерное макетирование Е.В. Кораблевой

Подписано в печать 26.11.07
Формат 60 × 84/16. 2,79 усл. печ. л. Тираж 70 экз. Заказ № 757

Издательско-полиграфический центр
Тамбовского государственного технического университета
392000, Тамбов, Советская, 106, к. 14

ВВЕДЕНИЕ

В современном мире юридическая деятельность, как правило, связана с большим объемом информации. В свою очередь, существует необходимость хранения столь значительных информационных массивов и быстрого поиска в них необходимых сведений. Существенно облегчить эти задачи позволяют базы данных – специальным образом организованная информация, в которой необходимо осуществить поиск нужных материалов. Для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации созданы специальные программы – системы управления базами данных.

Важность организации информации в виде баз данных непрерывно возрастает. Таким образом, освоение современных баз данных и систем управления базами данных является одним из важнейших этапов в подготовке квалифицированных специалистов юридического профиля.

Для подготовки таких специалистов разработаны материалы лекций к курсу. Представленные материалы посвящены, прежде всего, подробному рассмотрению современного состояния и перспектив развития баз данных. В работе изложены основные понятия, классификация данных, состав и порядок работ по проектированию баз данных. Особое внимание уделено построению информационно-логической модели предметной области.

В лекциях описывается также немаловажный язык реляционных баз данных SQL. Кроме того, излагаются функциональные возможности современных систем управления базами данных. В заключении приводится принципиальная схема работы с системой управления базами данных Microsoft Access.

1. БАЗЫ ДАННЫХ. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

1.1. ОБЩИЕ ПОЛОЖЕНИЯ

В этом году нам предстоит познакомиться с основными принципами организации и обработки больших массивов данных об объектах и явлениях реального мира. Такие массивы данных вместе с программно-аппаратными средствами для их обработки называют информационными системами (ИС).

Цель любой информационной системы – обработка данных об объектах реального мира. В широком смысле слова ***база данных*** – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области. Под предметной областью принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, предприятие, вуз и т.д.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование – это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Пример. На рис. 1 представлен пример неструктурированных данных, содержащих сведения о студентах (номер личного дела, фамилию, имя, отчество и год рождения). Легко убедиться, что сложно организовать поиск необходимых данных, хранящихся в неструктурированном виде, а упорядочить подобную информацию практически не представляется реальным.

Чтобы автоматизировать поиск и систематизировать эти данные, необходимо выработать определенные соглашения о способах представления данных, т.е. дату рождения нужно записывать одинаково для каждого студента, она должна иметь одинаковую длину и определенное место среди остальной информации. Эти же замечания справедливы и для остальных данных (номер личного дела, фамилия, имя, отчество).

Личное дело № 16493, Сергеев Петр Михайлович, дата рождения 1 января 1976; личное дело № 16593, Петрова Анна Владимировна, дата рождения 15 марта 1975; личное дело № 16693, Анохин Андрей Борисович, дата рождения 14 апреля 1976.

Рис. 1. Пример неструктурированных данных

Пример. После проведения несложной структуризации с информацией, указанной в примере выше, она будет выглядеть так:

№ личного дела	Фамилия	Имя	Отчество	Дата рождения
16493	Сергеев	Петр	Михайлович	01.01.76
16593	Петрова	Анна	Владимировна	15.03.75
16693	Анохин	Андрей	Борисович	14.04.76

Пользователями базы данных могут быть различные прикладные программы, программные комплексы, а также специалисты предметной области, выступающие в роли потребителей или источников данных, называемые конечными пользователями.

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария – системы управления базами данных.

База данных (БД) – это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система управления базами данных (СУБД) – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

1.2. КЛАССИФИКАЦИЯ БАЗ ДАННЫХ

По технологии обработки данных БД подразделяются на *централизованные* и *распределенные*.

Централизованная база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе. Такой способ использования баз данных часто применяют в локальных сетях ПК.

Распределенная база данных состоит из нескольких, возможно, пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на *базы данных с локальным доступом* и *базы данных с удаленным* (*сетевым*) *доступом*.

Системы централизованных баз данных с сетевым доступом предполагают различные архитектуры подобных систем:

- файл-сервер;
- клиент-сервер.

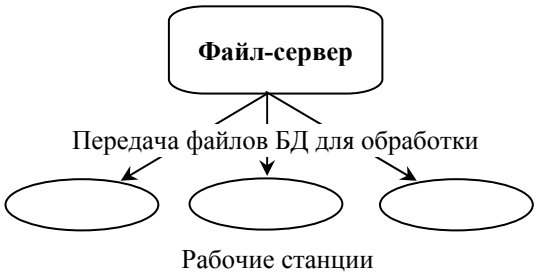

Рис. 2. Схема обработки информации БД по принципу файл-сервер

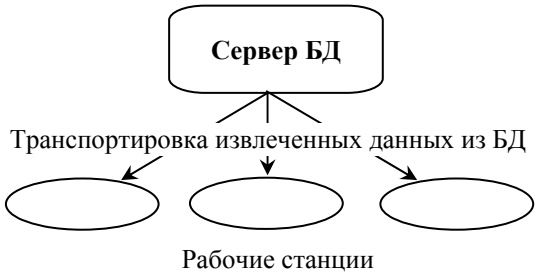

Рис. 3. Схема обработки информации в БД по принципу клиент-сервер

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также на рабочих станциях локальные БД, которые используются ими монопольно. Концепция файл-сервер условно отображена на рис. 2.

Клиент-сервер. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запросов SQL. Концепция клиент-сервер условно изображена на рис. 3.

1.3. СТРУКТУРНЫЕ ЭЛЕМЕНТЫ БАЗЫ ДАННЫХ

Понятие базы данных тесно связано с такими понятиями структурных элементов, как поле, запись, файл (рис. 4).

Поле – элементарная единица логической организации данных, которая соответствует неделимой единице информации – реквизиту. Для описания поля используются следующие характеристики:

имя, например, Фамилия, Имя, Отчество, Дата рождения;

тип, например, символьный, числовой, календарный;

длина, например, 15 байт, причем будет определяться максимально возможным количеством символов;

точность для числовых данных, например, два десятичных знака для отображения дробной части числа.

Запись – совокупность логически связанных полей. Экземпляр записи – отдельная реализация записи, содержащая конкретные значения ее полей. Файл (таблица) – совокупность экземпляров записей одной структуры.

Описание логической структуры записи файла содержит последовательность расположения полей записи и их основные характеристики, как это показано на рис. 5.

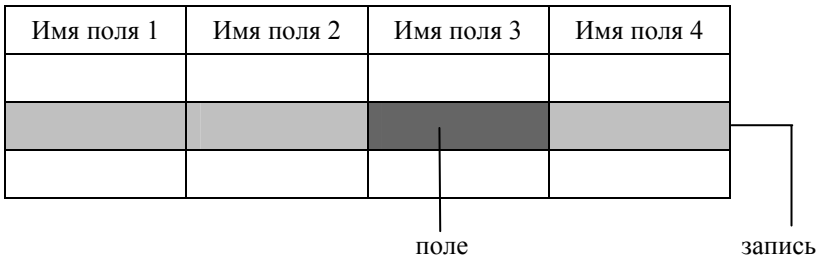

Рис. 4. Основные структурные элементы БД

Имя файла					
Поле		Признак ключа	Формат поля		
Имя (обозначение)	Полное наименование		Тип	Длина	Точность (для чисел)
имя 1					
имя *N*					

Рис. 5. Описание логической структуры записи файла

Имя файла: СТУДЕНТ					
Поле		Признак	Формат поля		
Обозначение	Наименование	ключа	Тип	Длина	Точность
Номер	№ личного дела	*	Симв.	5	
Фамилия	Фамилия студента		Симв.	15	
Имя	Имя студента		Симв.	10	
Отчество	Отчество студента		Симв.	IS	
Дата	Дата рождения		Дата	8	

Рис. 6. Описание логической структуры записи файла СТУДЕНТ

В структуре записи файла указываются поля, значения которых являются ключами: первичными (ПК), которые идентифицируют экземпляр записи, и вторичными (ВК), которые выполняют роль поисковых или группировочных признаков (по значению вторичного ключа можно найти несколько записей).

Пример. На рис. 6 приведен пример описания логической структуры записи файла (таблицы) СТУДЕНТ. Структура записи файла СТУДЕНТ линейная, она содержит записи фиксированной длины. Повторяющиеся группы значений полей в записи отсутствуют. Обращение к значению поля производится по его номеру.

1.4. ВИДЫ МОДЕЛЕЙ ДАННЫХ

Общие положения. Ядром любой базы данных является модель данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной обла-сти и взаимосвязи между ними.

Модель данных – совокупность структур данных и операций их обработки.

СУБД основывается на использовании иерархической, сетевой или реляционной модели, на комбинации этих моделей или на некотором их подмножестве.

Рассмотрим три основных типа моделей данных: *иерархическую*, *сетевую* и *реляционную*.

Иерархическая модель данных. Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево), вид которого представлен на рис. 7.

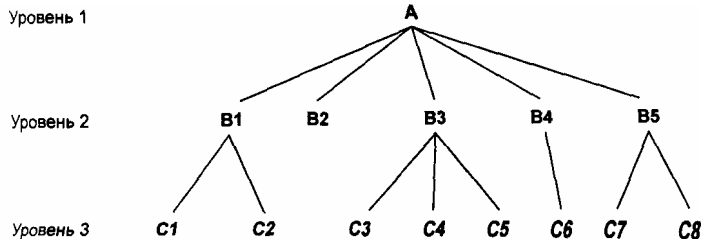

Рис. 7. Графическое изображение иерархической структуры БД

Рис. 8. Пример иерархической структуры БД

К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь. *Узел* – это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне. Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях. Количество деревьев в базе данных определяется числом корневых записей.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рис. 7, для записи С4 путь проходит через записи А и В3.

Пример, представленный на рис. 8, иллюстрирует использование иерархической модели базы данных.

Для рассматриваемого примера иерархическая структура правомерна, так как каждый студент учится в определенной (только одной) группе, которая относится к определенному (только одному) институту.

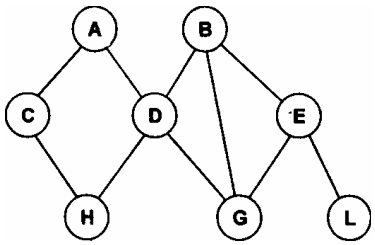

Рис. 9. Графическое изображение сетевой структуры

Сетевая модель данных. В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

На рис. 9 изображена сетевая структура базы данных в виде графа.

Примером сложной сетевой структуры может служить структура базы данных, содержащей сведения о студентах, участвующих в научно-исследовательских работах (НИРС). Возможно участие одного студента в нескольких НИРС, а также участие нескольких студентов в разработке одной НИРС. Графическое изображение описанной в примере сетевой структуры, состоящей только из двух типов записей, показано на рис. 10. Единственное отношение представляет собой сложную связь между записями в обоих направлениях.

Реляционная модель данных. Понятие реляционный (англ. *relation* – отношение) связано с разработками известного американского специалиста в области систем баз данных Е. Кодда.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Рис. 10. Пример сетевой структуры БД

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;

- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Пример. Реляционной таблицей можно представить информацию о студентах, обучающихся в вузе.

№ личного дела	Фамилия	Имя	Отчество	Дата рождения	Группа
16493	Сергеев	Петр	Михайлович	01.01.76	111
16593	Петрова	Анна	Владимировна	15.03.75	112
16693	Анохин	Андрей	Борисович	14.04.76	113

Отношения представлены в виде таблиц, строки которых соответствуют кортежам или записям, а столбцы – атрибутам отношений, доменам, полям.

Поле, каждое значение которого однозначно определяет соответствующую запись, называется простым ключом (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет составной ключ. В представленной выше таблице ключевым полем является «№ личного дела».

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей); в противном случае нужно ввести в структуру первой таблицы внешний ключ – ключ второй таблицы.

Пример. На рис. 11 показан пример реляционной модели, построенной на основе отношений: СТУДЕНТ, СЕССИЯ, СТИПЕНДИЯ.

СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа);

СЕССИЯ (Номер, Оценка1, Оценка2, Оценка3, Оценка4, Результат);

СТИПЕНДИЯ (Результат, Процент).

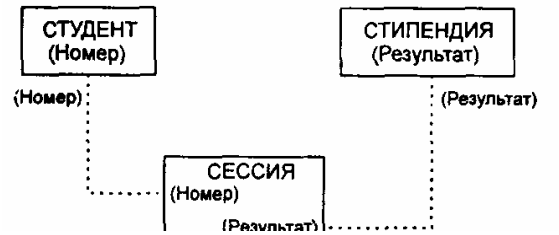

Рис. 11. Пример реляционной модели

Таблицы СТУДЕНТ И СЕССИЯ имеют совпадающие ключи (Номер), что дает возможность легко организовать связь между ними. Таблица СЕССИЯ имеет первичный ключ Номер и содержит внешний ключ Результат, который обеспечивает ее связь с таблицей СТИПЕНДИЯ.

2. РЕЛЯЦИОННЫЙ ПОДХОД К ПОСТРОЕНИЮ ИНФОЛОГИЧЕСКОЙ МОДЕЛИ

2.1. ПОНЯТИЕ ИНФОРМАЦИОННОГО ОБЪЕКТА

Информационный объект – это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных реквизитов (информационных элементов).

Информационный объект определенного реквизитного состава и структуры образует класс (тип), которому присваивается уникальное имя (символьное обозначение), например Студент, Сессия, Стипендия.

Информационный объект имеет множество реализации – экземпляров, каждый из которых представлен совокупностью конкретных значений реквизитов и идентифицируется значением ключа (простого – один реквизит или составного – несколько реквизитов). Остальные реквизиты информационного объекта являются описательными. При этом одни и те же реквизиты в одних информационных объектах могут быть ключевыми, а в других – описательными. Информационный объект может иметь несколько ключей.

Пример структуры и экземпляров информационного объекта *Студент*.

Структура	Номер	Фамилия	Имя	Отчество	Дата	Группа
Экземпляры информационного объекта Студент	16493	Сергеев	Петр	Михайлович	01.01.76	111
	16593	Петрова	Анна	Владимировна	15.03.75	112
	16693	Анохин	Андрей	Борисович	14.04.76	111

В информационном объекте *Студент* ключом является реквизит *Номер* (№ личного дела), к описательным реквизитам относятся: *Фамилия* (Фамилия студента), *Имя* (Имя студента), *Отчество* (Отчество студента), *Дата* (Дата рождения), *Группа* (№ группы). Если отсутствует реквизит *Номер*, то для однозначного определения характеристик конкретного студента необходимо использование составного ключа из трех реквизитов: *Фамилия* + *Имя* + *Отчество*.

2.2. НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ

Одни и те же данные могут группироваться в таблицы (отношения) различными способами, т.е. возможна организация различных наборов отношений взаимосвязанных информационных объектов. Группировка атрибутов в отношениях должна быть рациональной, т.е. минимизирующей дублирование данных и упрощающей процедуры их обработки и обновления.

Определенный набор отношений обладает лучшими свойствами при включении, модификации, удалении данных, чем все остальные возможные наборы отношений, если он отвечает требованиям нормализации отношений.

Нормализация – это разбиение таблицы на две или более, обладающие 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 отношений – формальный аппарат ограничений на формирование отношений (таблиц), который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение (ввод, корректировку) базы данных.

Коддом Е. (основателем реляционной модели) выделены три нормальные формы отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей (самой совершенной) нормальной форме, но существуют *нормальная форма Бойса-Кодда* (*НФБК*), *пятая нормальная форма* (*5НФ*). *Четвертая нормальная форма* (*4НФ*) является частным случаем 5НФ.

Каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется *нормализованной*. Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы, даже не допускаются в реляционной БД.

Всякая нормализованная таблица автоматически считается таблицей в *первой нормальной форме* (сокращенно *1НФ*).

Отношение называется **нормализованным** или **приведенным к первой нормальной форме**, если все его атрибуты простые (далее не делимы).

Таблица находится в *первой нормальной форме* (*1НФ*) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто.

Таким образом, строго говоря, «нормализованная» и «находящаяся в 1НФ» означают одно и то же. Однако на практике термин «нормализованная» часто используется в более узком смысле – «полностью нормализованная», который означает, что в проекте не нарушаются никакие принципы нормализации.

Например, отношение Студент (Номер, Фамилия, Имя, Отчество, Дата, Группа) находится в первой нормальной форме.

Таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

Описательные реквизиты информационного объекта логически связаны с общим для них ключом, эта связь носит характер функциональной зависимости реквизитов.

Функциональная зависимость реквизитов – зависимость, при которой в экземпляре информационного объекта определенному значению ключевого реквизита соответствует только одно значение описательного реквизита. (Такое определение функциональной зависимости позволяет при анализе всех взаимосвязей реквизитов предметной области выделить самостоятельные информационные объекты.)

Функциональная зависимость. Поле *B* таблицы функционально зависит от поля *A* той же таблицы в том и только том случае, когда в любой заданный момент времени для каждого из различных значений поля *A* обязательно существует только одно из различных значений поля *B*. (Отметим, что здесь допускается, что поля *A* и *B* могут быть составными.)

Пример графического изображения функциональных зависимостей реквизитов *Студент* показан на рис. 12, на котором ключевой реквизит указан *.

В случае составного ключа вводится понятие функционально полной зависимости.

Функционально полная зависимость неключевых атрибутов заключается в том, что каждый неключевой атрибут функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Полная функциональная зависимость. Поле *B* находится в полной функциональной зависимости от составного поля *A*, если он нально зависит от *A* и не зависит функционально от любого подмножества пс

Многозначная зависимость. Поле *A* мног ределяет поле *B* той же таблицы, если для каждого значения поля *A* существ о определенное множество соответствующих значений *B*.

Номер*
Фамилия
Имя
Отчество
Дата
Группа

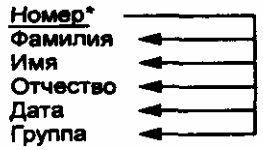

Рис. 12. Графическое изображение функциональной зависимости реквизитов

Таблица находится во *второй нормальной форме* (*2НФ*), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом. (Отношение будет находиться во второй нормальной форме, если оно находится в первой нормальной форме, и каждый неключевой атрибут функционально полно зависит от составного ключа.)

Пример. Отношение *Студент* (*Номер*, *Фамилия*, *Имя*, *Отчество*, *Дата*, *Группа*) находится в первой и во второй нормальной форме одновременно, так как описательные реквизиты однозначно определены и функционально зависят от ключа *Номер*. Отношение *Успеваемость* (*Номер*, *Фамилия*, *Имя*, *Отчество*, *Дисциплина*, *Оценка*) находится в первой нормальной форме и имеет составной ключ *Номер + Дисциплина*. Это отношение не находится во второй нормальной форме, так как атрибуты *Фамилия*, *Имя*, *Отчество* не находятся в полной функциональной зависимости с составным ключом отношения.

Понятие третьей нормальной формы основывается на понятии нетранзитивной зависимости.

Транзитивная зависимость наблюдается в том случае, если один из двух описательных реквизитов зависит от ключа, а другой описательный реквизит зависит от первого описательного реквизита.

Таблица находится в *третьей нормальной форме* (*3НФ*), если она удовлетворяет определению 2НФ и ни одно из ее неключевых полей не зависит функционально от любого другого неключевого поля. (Отношение будет находиться в третьей нормальной форме, если оно находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.)

Пример. Если в состав описательных реквизитов информационного объекта *Студент* включить фамилию старосты группы (*Староста*), которая определяется только номером группы, то одна и та же фамилия старосты будет многократно повторяться в разных экземплярах данного информационного объекта. В этом случае наблюдаются затруднения в корректировке фамилии старосты в случае назначения нового старосты, а также неоправданный расход памяти для хранения дублированной информации.

Для устранения транзитивной зависимости описательных реквизитов необходимо провести «расщепление» исходного информационного объекта. В результате расщепления часть реквизитов удаляется из исходного информационного объекта и включается в состав других (возможно, вновь созданных) информационных объектов.

«Расщепление» информационного объекта, содержащего транзитивную зависимость описательных реквизитов, показано на рис. 13. Как видно из рис. 13, исходный информационный объект *Студент группы* представляется

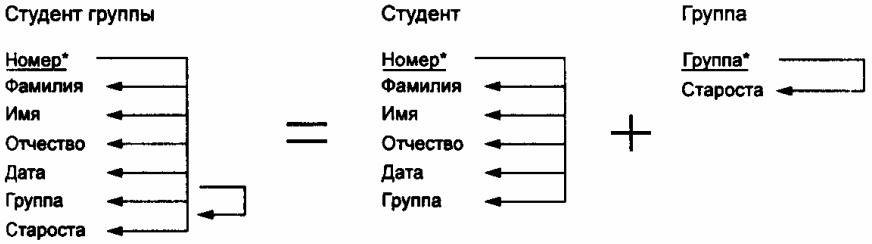

Рис. 13. Пример «расщепления» структуры информационного объекта

в виде совокупности правильно структурированных информационных объектов (*Студент* и *Группа*), реквизитный состав которых тождественен исходному объекту. Отношение *Студент* (*Номер*, *Фамилия*, *Имя*, *Отчество*, *Дата*, *Группа*) находится одновременно в первой, второй и третьей нормальной форме.

Таблица находится в *нормальной форме Бойса-Кодда* (*НФБК*), если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от *возможного* ключа.

Полной декомпозицией таблицы называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым таблицы.

Таблица находится в *пятой нормальной форме* (*5НФ*) тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный ключ. Таблица, не имеющая ни одной полной декомпозиции, также находится в 5НФ.

2.3. ТИПЫ СВЯЗЕЙ

Все информационные объекты предметной области связаны между собой. Различаются несколько типов связей, для которых введены следующие обозначения:

- один к одному (1 : 1);
- один ко многим (1 : M);
- многие ко многим (M : M).

Пример. Дана совокупность информационных объектов, отражающих учебный процесс в вузе:

СТУДЕНТ (*Номер*, *Фамилия*, *Имя*, *Отчество*, *Пол*, *Дата рождения*, *Группа*) *СЕССИЯ* (*Номер*, *Оценка1*, *Оценка2*, *Оценка3*, *Оценка4*, *Результат*) *СТИПЕНДИЯ* (*Результат*, *Процент*) *ПРЕПОДАВАТЕЛЬ* (*Код преподавателя*, *Фамилия*, *Имя*, *Отчество*).

Связь один к одному (1 : 1) предполагает, что в каждый момент времени одному экземпляру информационного объекта *A* соответствует не более одного экземпляра информационного объекта *B* и наоборот (рис. 14).

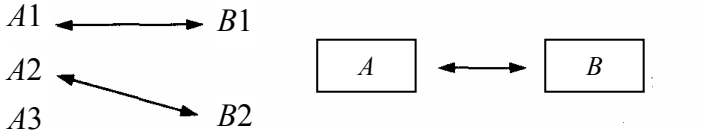

Рис. 14. Графическое изображение реального отношения 1 : 1

Примером связи 1 : 1 может служить связь между информационными объектами *СТУДЕНТ* и *СЕССИЯ*: СТУДЕНТ ↔ СЕССИЯ. Каждый студент имеет определенный набор экзаменационных оценок в сессию.

При связи один ко многим (1 : M) одному экземпляру информационного объекта *A* соответствует 0, 1 или более экземпляров объекта *B*, но каждый экземпляр объекта *B* связан не более чем с 1 экземпляром объекта *A* (рис. 15).

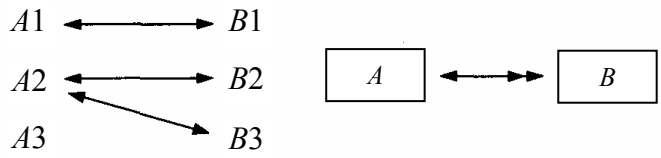

Рис. 15. Графическое изображение реального отношения 1 : M

Примером связи 1 : M служит связь между информационными объектами *СТИПЕНДИЯ* и *СЕССИЯ*: СТИПЕНДИЯ ↔ СЕССИЯ. Установленный размер стипендии по результатам сдачи сессии может повторяться многократно для различных студентов.

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 (рис. 16).

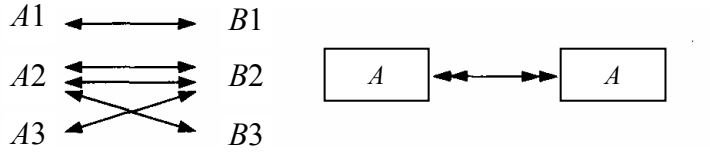

Рис. 16. Графическое изображение реального отношения M : M

Примером данного отношения служит связь между информационными объектами *СТУДЕНТ* и *ПРЕПОДАВАТЕЛЬ*: СТУДЕНТ ↔ ПРЕПОДАВАТЕЛЬ. Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов.

2.4. СТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ МОДЕЛИ

Архитектура СУБД. Базы данных и программные средства их создания и ведения (СУБД) имеют многоуровневую архитектуру, представление о которой можно получить из рис. 17.

Различают концептуальный, внутренний и внешний уровни представления данных БД, которым соответствуют модели аналогичного назначения.

Концептуальный уровень соответствует логическому аспекту представления данных предметной области в интегрированном виде. Концептуальная модель состоит из множества экземпляров различных типов данных, структурированных в соответствии с требованиями СУБД к логической структуре базы данных.

Внутренний уровень отображает требуемую организацию данных в среде хранения и соответствует физическому аспекту представления данных. Внутренняя модель состоит из отдельных экземпляров записей, физически хранимых во внешних носителях.

Внешний уровень поддерживает частные представления данных, требуемые конкретным пользователям. Внешняя модель является подмножеством концептуальной модели. Возможно пересечение внешних моделей по данным. Частная логическая структура данных для отдельного приложения (задачи) или пользователя соответствует внешней модели или

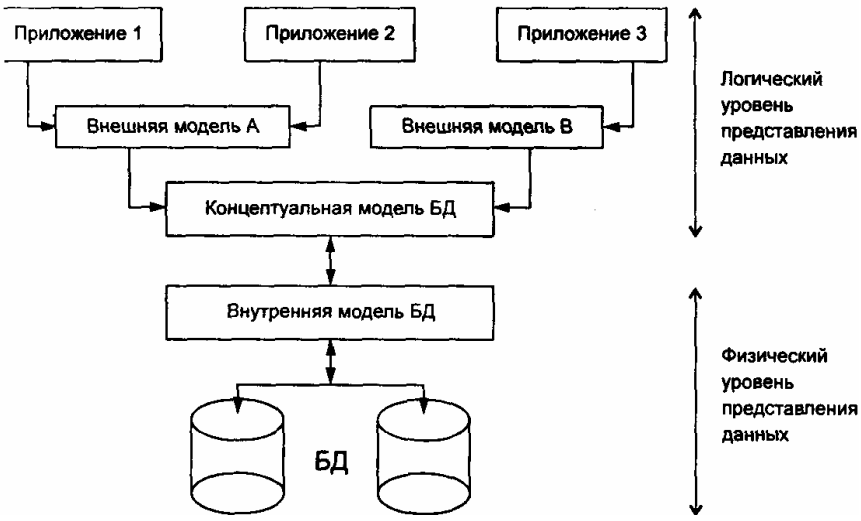

Рис. 17. Многоуровневое представление данных БД под управлением СУБД

подсхеме БД. С помощью внешних моделей поддерживается санкционированный доступ к данным БД приложений (ограничен состав и структура данных концептуальной модели БД, доступных в приложении, а также заданы допустимые режимы обработки этих данных: ввод, редактирование, удаление, поиск).

Пример. Соотношение между концептуальной и внешними моделями базы данных приведено на рис. 18.

Появление новых или изменение информационных потребностей существующих приложений требуют определения для них корректных внешних моделей, при этом на уровне концептуальной и внутренней модели данных изменений не происходит. Изменения в концептуальной модели, вызванные появлением новых видов данных или изменением их структур, могут затрагивать не все приложения, т.е. обеспечивается определенная независимость программ от данных. Изменения в концептуальной модели должны отражаться на внутренней модели, и при неизменной концептуальной модели возможна самостоятельная модификация внутренней модели БД с целью улучшения ее характеристик (время доступа к данным, расхода памяти внешних устройств и др.). Таким образом БД реализует принцип относительной независимости логической и физической организации данных.

Понятие информационно-логической модели. Проектирование базы данных состоит в построении комплекса взаимосвязанных моделей данных. На рис. 19 условно отображены этапы процесса проектирования базы данных.

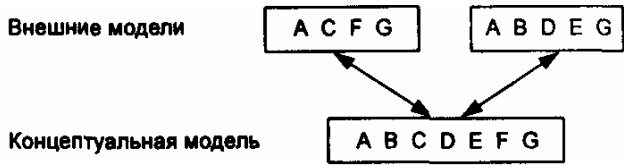

Рис. 18. Пример соотношения между концептуальной моделью и внешними моделями

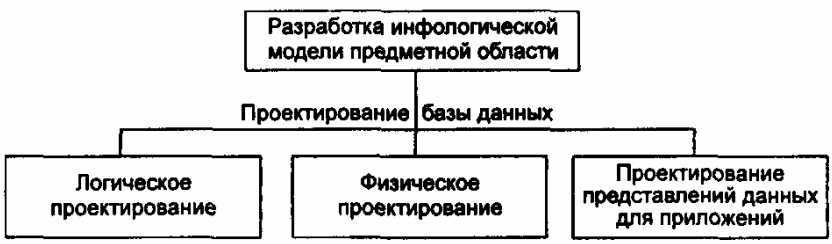

Рис. 19. Этапы процесса проектирования базы данных

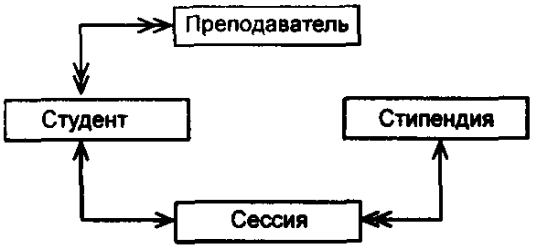

Рис. 20. Пример графического представления инфологической модели

Важнейшим этапом проектирования базы данных является разработка инфологической (информационно-логической) модели предметной области, не ориентированной на СУБД. В инфологической модели средствами структур данных в интег-

рированном виде отражают состав и структуру данных, а также информационные потребности приложений (задач и запросов).

Информационно-логическая (инфологическая) модель предметной области отражает предметную область в виде совокупности информационных объектов и их структурных связей.

Инфологическая модель предметной области строится первой. Предварительная инфологическая модель строится еще на предпроектной стадии и затем уточняется на более поздних стадиях проектирования баз данных. Затем на ее основе строятся концептуальная (логическая), внутренняя (физическая) и внешняя модели.

Пример. На рис. 20 представлена графическая форма информационно-логической модели, связывающей информационные объекты: *Студент*, *Сессия*, *Стипендия*, *Преподаватель*.

3. ЯЗЫК РЕЛЯЦИОННЫХ БАЗ ДАННЫХ SQL

Реляционная модель данных позволяет пользователям работать в ненавигационной манере, т.е. для выборки информации из БД человек должен всего лишь указать список интересующих его таблиц и те условия, которым должны удовлетворять выбираемые данные. СУБД скрывает от пользователя выполняемые ей последовательные просмотры таблиц, выполняя их наиболее эффективным образом. Очень важная особенность реляционных систем состоит в том, что результатом выполнения любого запроса к таблицам БД является также таблица, которую можно сохранить в БД и/или по отношению к которой можно выполнять новые запросы. Таким языком является язык SQL.

SQL – инструмент, предназначенный для обработки и чтения данных, содержащихся в компьютерной базе данных. SQL – это сокращенное название структурированного языка запросов (Structured Query Language). Как следует из названия, SQL является *языком программирования*, который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемых *реляционными.*

Если пользователю необходимо прочитать данные из базы данных, он запрашивает их у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется *запросом* к базе данных: отсюда и название – структурированный язык *запросов.*

Однако это название не совсем соответствует действительности. Во-первых, сегодня SQL представляет собой нечто гораздо большее, чем простой инструмент создания запросов, хотя именно для этого он и был первоначально предназначен. Несмотря на то что чтение данных по-прежнему остается одной из наиболее важных функций SQL, сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю, а именно:

1. **Интерактивный язык запросов.** Пользователи вводят команды SQL в интерактивные программы, предназначенные для чтения данных и отображения их на экране. Это удобный способ выполнения специальных запросов.

2. **Язык программирования БД.** Чтобы получить доступ к базе данных, программисты вставляют в свои программы команды SQL. Эта методика используется как в программах, написанных пользователями, так и в служебных программах баз данных (таких, как генераторы отчетов и инструменты ввода данных).

3. **Язык администрирования БД.** Администратор базы данных, находящейся на миникомпьютере или на большой ЭВМ, использует SQL для определения структуры базы данных и управления доступом к данным.

4. **Язык создания приложений клиент/сервер и программа для персональных компьютеров.** SQL используется для организации связи через локальную сеть с сервером базы данных, в которой хранятся совместно используемые данные. В большинстве новых приложений используется архитектура клиент/сервер, которая позволяет свести к минимуму сетевой трафик и повысить быстродействие как персональных компьютеров, так и серверов баз данных.

5. **Язык распределенных БД.** В системах управления распределенными базами данных SQL помогает распределять данные среди нескольких взаимодействующих вычислительных систем. Программное обеспечение каждой системы посредством использования SQL связывается с другими системами, посылая им запросы на доступ к данным.

6. **Язык шлюзов БД.** В вычислительных сетях с различными СУБД SQL часто используется в шлюзовой программе, которая позволяет СУБД одного типа связываться с СУБД другого типа.

Таким образом, SQL превратился в полезный и мощный инструмент, обеспечивающий людям, программам и вычислительным системам доступ к информации, содержащейся в реляционных базах данных.

Все языки манипулирования данными (ЯМД), созданные до появления реляционных баз данных, были ориентированы на операции с данными, представленными в виде логических записей файлов. Это требовало от пользователей детального знания организации хранения данных и достаточных усилий для указания не только того, какие данные нужны, но и того, где они размещены и как шаг за шагом получить их.

Язык SQL ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц. Особенность предложений этого языка состоит в том, что они ориентированы в большей степени на конечный результат обработки данных, чем на процедуру этой обработки. SQL сам определяет, где находятся данные, какие индексы и даже наиболее эффективные последовательности операций следует использовать для их получения: не надо указывать эти детали в запросе к базе данных.

SQL – это легкий для понимания язык с небольшим количеством операторов (около 30) и в то же время универсальное программное средство управления данными. Операторы SQL выглядят как обычные английские предложения, что упрощает их изучение и понимание. Частично это обусловлено тем, что операторы SQL описывают данные, которые необходимо получить, а не определяют способ их поиска. Таблицы и столбцы в реляционной базе данных могут иметь длинные описательные имена. В результате большинство операторов SQL означают именно то, что точно соответствует их именам, поэтому их можно читать как простые, понятные предложения.

SQL может использоваться как интерактивный (для выполнения запросов) и как встроенный (для построения прикладных программ). В нем существуют:

– предложения определения данных (определение баз данных, а также определение и уничтожение таблиц и индексов);

– запросы на выбор данных (предложение SELECT);

– предложения модификации данных (добавление, удаление и изменение данных);

– предложения управления данными (предоставление и отмена привилегий на доступ к данным, управление транзакциями и другие). Кроме того, он предоставляет возможность выполнять в этих предложениях:

– арифметические вычисления (включая разнообразные функциональные преобразования), обработку текстовых строк и выполнение операций сравнения значений арифметических выражений и текстов;

– упорядочение строк и (или) столбцов при выводе содержимого таблиц на печать или экран дисплея;

– создание представлений (виртуальных таблиц), позволяющих пользователям иметь свой взгляд на данные без увеличения их объема в базе данных;

– запоминание выводимого по запросу содержимого таблицы, нескольких таблиц или представления в другой таблице (реляционная операция присваивания);

– агрегатирование данных: группирование данных и применение к этим группам таких операций, как среднее, сумма, максимум, минимум, число элементов и т.п.

Первый, достаточно полный функционально вариант языка SQL был разработан и реализован в рамках проекта экспериментальной реляционной СУБД SystemR компании IBM (1974 – 1979 гг.). Его исходным названием было SEQUEL (Structured English Query Language).

Благодаря своей элегантности и машинной независимости, а также поддержке промышленными лидерами в технологии реляционных баз данных, SQL был признан стандартным языком и в обозримом будущем сохранит свои позиции.

Официальный стандарт языка SQL является совместной разработкой Американским институтом национальных стандартов (American National Standards Institute – ANSI) и Международной организацией по стандартам (International Standards Organization – ISO).

Работа над официальным стандартом языка SQL началась с 1982 г. В 1989 г. был принят первый стандарт SQL (SQL1 или SQL/89). Он во многих частях имеет чрезвычайно общий характер и допускает очень широкое толкование. Поэтому на фоне завершения разработки этого стандарта была начата работа над стандартом SQL2. Она длилась несколько лет, и в 1992 г. был выработан окончательный проект стандарта SQL2 (SQL/92).

В 2000 г. большинство коммерческих продуктов соответствовало SQL92. Даже сейчас еще не все элементы этого стандарта широко реализованы. Впрочем, многие коммерческие СУБД расширяют SQL за рамки стандарта, добавляя другие возможности. В любой из версий отклонения от стандарта должны описываться в документации на программный продукт. В 1999 г. был принят стандарта SQL3 (SQL/99).

4. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Обзор СУБД. Системой управления базами данных называют программную систему, предназначенную для создания на ЭВМ общей базы данных, используемой для решения множества задач. Подобные системы служат для поддержания базы данных в актуальном состоянии и обеспечивают эффективный доступ пользователей к содержащимся в ней данным в рамках предоставленных пользователям полномочий.

СУБД предназначена для централизованного управления базой данных в интересах всех работающих в этой системе.

По степени универсальности различают два класса СУБД:

– системы общего назначения;

– специализированные системы.

СУБД общего назначения не ориентированы на какую-либо предметную область или на информационные потребности какой-либо группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной системе и поставляется многим пользователям как коммерческое изделие. Такие СУБД обладают средствами настройки на работу с конкретной базой данных. Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, основанных на кинологии баз данных, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Этим СУБД присущи развитые функциональные возможности и даже определенная функциональная избыточность.

Специализированные СУБД создаются в редких случаях при невозможности или нецелесообразности использования СУБД общего назначения.

СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией базы данных информационной системы.

Рынок программного обеспечения ПК располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем управления базами данных общего назначения, а также средствами их окружения практически для всех массовых моделей машин и для различных операционных систем.

Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности, что дает возможность разработчикам гарантировать полную безопасность данных при меньших затратах сил на низкоуровневое программирование. Продукты, функционирующие в среде WINDOWS, выгодно отличаются удобством пользовательского интерфейса и встроенными средствами повышения производительности.

Приведем сравнение (объективное и не объективное) СУБД ACCESS, MySQL, а также СУБД Oracle по некоторым параметрам.

1. Объем памяти на жестком диске, необходимый для самой СУБД: ACCESS (OfficeXP) – 530 Мб, Oracle – > 1 Гб, для работы с MySQL + PHP через Интернет-сервер необходим только браузер (например Internet Explorer – 14,7 Мб), а для работы локально нужен еще сервер, поддерживающий MySQL и PHP (например Apache – 8 Мб).
2. Размер описанной выше базы данных в формате, соответствующем каждой СУБД: ACCESS – 1,73 Мб, MySQL – 113 Кб, Oracle – размер определяется не содержанием самой базы, а задаваемым табличным пространством.
3. Оперативная память, используемая СУБД при работе с той же базой данных: ACCESS – 4528 Кб, сервер Apache + Internet Explorer – 28 612 Кб (из них Internet Explorer – 11 660 Кб), Oracle – ? Кб.
4. Быстродействие: при работе локально разница между временем выполнения запроса в ACCESS и временем выполнения аналогичного запроса в MySQL (Internet Explorer – Apache – Internet Explorer) практически неощутима (десятые доли секунды); при работе же с MySQL через Internet скорость зависит от таких параметров, как трафик сети, удаленность и быстродействие сервера и пр., поэтому в данном случае сравнение быстродействия не совсем уместно.
5. Простота использования: ACCESS, как и все продукты из MS OFFICE, очень наглядна, содержит хорошую систему помощи и опции так называемых «мастеров» создания и заполнения, это все в совокупности позволяет даже неопытному пользователю, не имеющему навыков работы с какими-либо СУБД, довольно-таки быстро научиться создавать и управлять своими базами данных; MySQL – несмотря на то, что приходится прописывать все в ручную, особых трудностей не вызывает, особенно если пользователь обладает хоть какими-то навыками программирования и работы с БД (на своем примере – для того чтобы разобраться с «нуля» и выполнить л.р. № 3 хватило двух дней); Oracle – это СУБД несколько иного уровня, и поэтому требует изучения в течение большего, по сравнению с ACCESS и MySQL, времени.

Производительность СУБД. Производительность СУБД оценивается:

- временем выполнения запросов;
- скоростью поиска информации в неиндексированных полях;
- временем выполнения операций импортирования базы данных из других форматов;
- скоростью создания индексов и выполнения таких массовых операций, как обновление, вставка, удаление данных;
- максимальным числом параллельных обращений к данным в многопользовательском режиме;
- временем генерации отчета.

На производительность СУБД оказывают влияние два фактора:

- СУБД, которые следят за соблюдением целостности данных, несут дополнительную нагрузку, которую не испытывают другие программы;
- производительность собственных прикладных программ сильно зависит от правильного проектирования и построения базы данных.

Самые быстрые программные изделия отнюдь не обладают самыми развитыми функциональными возможностями на уровне процессора СУБД.

Обеспечение целостности данных на уровне базы данных. Эта характеристика подразумевает наличие средств, позволяющих удостовериться, что информация в базе данных всегда остается корректной и полной. Должны быть установлены правила целостности, и они должны храниться вместе с базой данных и соблюдаться на глобальном уровне. Целостность данных должна обеспечиваться независимо от того, каким образом данные заносятся в память (в интерактивном режиме, посредством импорта или с помощью специальной программы).

К средствам обеспечения целостности данных на уровне СУБД относятся:

- встроенные средства для назначения первичного ключа, в том числе средства для работы с типом полей с автоматическим приращением, когда СУБД самостоятельно присваивает новое уникальное значение;
- средства поддержания ссылочной целостности, которые обеспечивают запись информации о связях таблиц и автоматически пресекают любую операцию, приводящую к нарушению ссылочной целостности.

Некоторые СУБД имеют хорошо разработанный процессор СУБД для реализации таких возможностей, как уникальность первичных ключей, ограничение (пресечение) операций и даже каскадное обновление и удаление информации. В таких системах проверка корректности, назначаемая полю или таблице, будет проводиться всегда после изменения данных, а не только во время ввода информации с помощью экранной формы. Это свойство можно настраивать для каждого поля и для записи в целом, что позволяет контролировать не только значения отдельных полей, но и взаимосвязи между несколькими полями данной записи.

Обеспечение безопасности. Некоторые СУБД предусматривают средства обеспечения безопасности данных. Такие средства обеспечивают выполнение следующих операций:

- шифрование прикладных программ;
- шифрование данных;
- защиту паролем;
- ограничение уровня доступа (к базе данных, к таблице, к словарю, для пользователя).

Работа в многопользовательских средах. Практически все рассматриваемые СУБД предназначены для работы в многопользовательских средах, но обладают для этого различными возможностями.

Обработка данных в многопользовательских средах предполагает выполнение программным продуктом следующих функций:

- блокировку базы данных, файла, записи, поля;
- идентификацию станции, установившей блокировку;
- обновление информации после модификации;
- контроль за временем и повторение обращения;
- обработку транзакций (транзакция – последовательность операций пользователя над базой данных, которая сохраняет ее логическую целостность);
- работу с сетевыми системами (LAN Manager, NetWare, Unix).

Импорт-экспорт. Эта характеристика отражает:

– возможность обработки СУБД информации, подготовленной другими программными средствами;

– возможность использования другими программами данных, сформированных средствами рассматриваемой СУБД.

Особый интерес представляют следующие форматы файлов: ASCII-файлы, .DBF, .WK*. .XLS.

Все рассматриваемые здесь СУБД обладают хорошими возможностями импорта-экспорта данных.

Доступ к данным посредством языка SQL. Язык запросов SQL (Structured Query Language) реализован в целом ряде популярных СУБД для различных типов ЭВМ либо как базовый, либо как альтернативный. В силу своего широкого использования является международным стандартом языка запросов. Язык SQL предоставляет развитые возможности как конечным пользователям, так и специалистам в области обработки данных.

Совместимость с SQL-системами играет большую роль, когда предполагается проведение работы с корпоративными данными. СУБД, хорошо подготовленные к работе в качестве средств первичной обработки информации для SQL-систем, могут открыть двери в системы с архитектурой клиент-сервер.

СУБД имеют доступ к данным SQL в следующих случаях:

- базы данных совместимы с ODBC (Open Database Connectivity – открытое соединение баз данных);
- реализована естественная поддержка SQL-баз данных;
- возможна реализация SQL-запросов локальных данных.

Многие СУБД могут «прозрачно» подключаться к входным SQL-подсистемам с помощью ODBC или драйверов, являющихся их частью, поэтому существует возможность создания прикладных программ для них. Некоторые программные продукты совместимы также с SQL при обработке интерактивных запросов на получение данных, находящихся на сервере или на рабочем месте.

Возможности запросов и инструментальные средства разработки прикладных программ. СУБД, ориентированные на разработчиков, обладают развитыми средствами для создания приложений. К элементам инструментария разработки приложений можно отнести:

- мощные языки программирования;
- средства реализации меню, экранных форм ввода–вывода данных и генерации отчетов;
- средства генерации приложений (прикладных программ);
- генерацию исполнимых файлов.

Функциональные возможности моделей данных доступны пользователю СУБД благодаря ее языковым средствам.

Реализация языковых средств интерфейсов может быть осуществлена различными способами. Для высококвалифицированных пользователей (разработчиков сложных прикладных систем) языковые средства чаще всего представляются в их явной синтаксической форме. В других случаях функции языков могут быть доступны косвенным образом, когда они реализуются в форме различного рода меню, диалоговых сценариев или заполняемых пользователем таблиц. По таким входным данным интерфейсные средства формируют адекватные синтаксические конструкции языка интерфейса и передают их на исполнение или включают в генерируемый программный код приложения. Интерфейсы с неявным использованием языка широко используются в СУБД для персональных ЭВМ. Примером такого языка является язык QBE (Query-By-Example).

Языковые средства используются для выполнения двух основных функций:

- описания представления базы данных;
- выполнения операций манипулирования данными.

Первая из этих функций обеспечивается языком описания данных (ЯОД). Описание базы данных средствами ЯОД называется схемой базы данных. Оно включает описание структуры базы данных и налагаемых на нее ограничений целостности в рамках тех правил, которые регламентированы моделью данных используемой СУБД. ЯОД некоторых СУБД обеспечивают также возможности задания ограничений доступа к данным или полномочий пользователей.

ЯОД не всегда синтаксически оформляется в виде самостоятельного языка. Он может быть составной частью единого языка данных, сочетающего возможности определения данных и манипулирования данными.

Язык манипулирования данными (ЯМД) позволяет запрашивать предусмотренные в системе операции над данными из базы данных.

Имеются многочисленные примеры языков СУБД, объединяющих возможности описания данных и манипулирования данными в единых синтаксических рамках. Популярным языком такого рода является реляционный язык SQL.

Все рассматриваемые программные средства обладают автоматизированными средствами создания экранных форм, запросов, отчетов, меню, наклеек, стандартных писем. Для создания указанных визуальных и структурных объектов ряд СУБД использует специальные инструментальные средства, называемые «мастерами» или «волшебниками».

5. ПРИНЦИПИАЛЬНАЯ СХЕМА РАБОТЫ С MICROSOFT ACCESS

Каждая конкретная СУБД имеет свои особенности, которые необходимо учитывать.

Однако имея представление о функциональных возможностях любой СУБД, можно представить обобщенную технологию работы пользователя в этой среде.

В качестве основных этапов обобщенной технологии работы с СУБД, которая схематично представлена на рис. 21, можно выделить следующие:

- создание структуры таблиц базы данных;
- ввод и редактирование данных в таблицах;
- обработка данных, содержащихся в таблицах;
- вывод информации из базы данных.

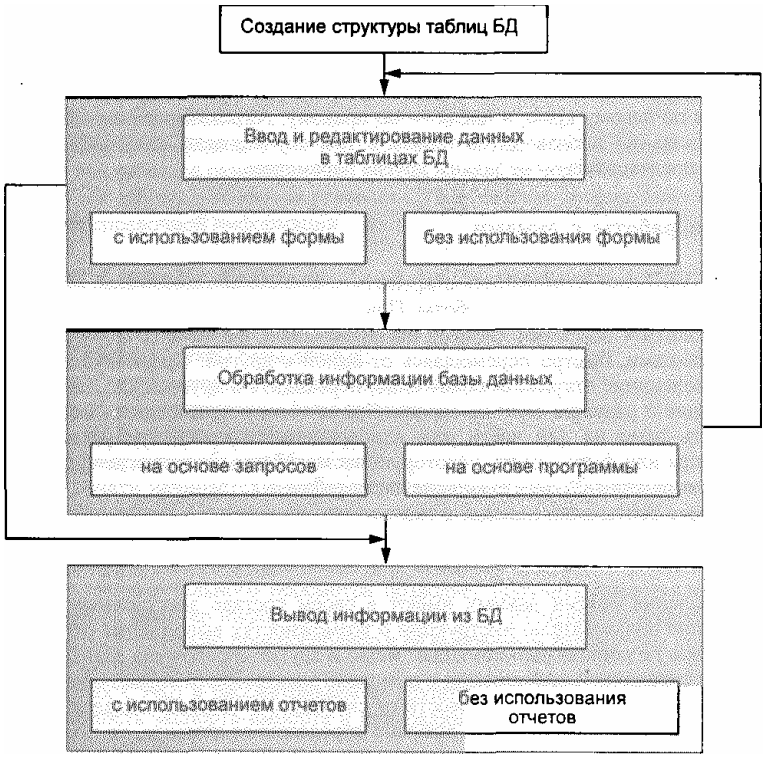

Рис. 21. Схема обобщенной технологии работы в СУБД

Рассмотрим выделенные этапы более подробно на примере MS Access.

MS Access – это система управления реляционными базами данных, предназначенная для работы на автономном ПК или в локальной вычислительной сети под управлением Microsoft Windows. Другими словами, MS Access – это набор инструментальных средств для создания и эксплуатации информационных систем. Средствами Access вы проводите следующие операции.

1. Проектирование базовых объектов ИС – двумерных таблиц, с разными типами данных, включая поля объектов OLE.
2. Установление связей между таблицами, с поддержкой целостности данных, каскадного обновления и удаления записей.
3. Ввод, хранение, просмотр, сортировка, модификация и выборка данных из таблиц с использованием различных средств контроля информации, индексирования таблиц и аппарата логической алгебры (для фильтрации данных).
4. Создание, модификация и использование производных объектов ИС (форм, запросов и отчетов), с помощью которых в свою очередь выполняются следующие операции:

– оптимизация пользовательского ввода и просмотра данных (формы);

– соединение данных из различных таблиц; проведение групповых операций (т.е. операций над группами записей, объединенных каким-то признаком) с расчетами и формированием вычисляемых полей; отбор данных с применением аппарата логической алгебры (запросы);

– составление печатных отчетов по данным, которые содержатся в таблицах и запросах БД.

MS Access обладает мощными, удобными и гибкими средствами проектирования объектов. Это дает возможность пользователю при минимуме предварительной подготовки довольно быстро создать полноценную ИС на уровне таблиц, форм, запросов-выборок и отчетов.

5.1. ОСНОВЫ РАБОТЫ С MS ACCESS

Запуск и завершение работы с MS Access осуществляются любым из стандартных способов, предусмотренных в среде Windows.

Объектом обработки MS Access является файл базы данных, имеющий произвольное имя и расширение MDB. В этот файл входят основные объекты MS Access: таблицы, формы, запросы, отчеты, страницы, макросы и модули.

После загрузки Accessa на панели справа вам предлагается открыть существующую или создать новую БД.

Создание БД. После выбора команды *Новая база данных* на экране появляется стандартное окно, в котором надо указать имя и адрес создаваемой базы данных. Создав файл, Access раскрывает пустое окно базы данных, и в этом окне будут проводиться все операции над объектами БД.

Открытие БД. Последние открывавшиеся файлы БД можно выбрать из списка *Открытие файла*. Если нужной БД нет в списке, то необходимо выбрать команду *Другие файлы*, после чего на экране появляется стандартное окно, в котором необходимо указать имя и адрес существующего файла базы данных. После этого раскрывается окно базы данных, которое обеспечивает доступ к уже созданным объектам БД и возможность создавать новые объекты.

MS Access является многооконным приложением, однако в любой момент может быть открыта только одна база данных. Именно ее окно является главным окном документа в приложении Access.

Однако это окно порождает множество дочерних окон (таблицы, запроса, формы и т.д.), и каждое такое окно может быть закрыто автономно (любым из стандартных способов Windows). Кроме того, не закрывая окна, можно сохранить объект, окно которого находится на экране, и присвоить ему имя, – точно так же, как это делается с файлами: командой [*Файл*/*Сохранить*] или [*Файл*/*Сохранить как...*].

Основные понятия MS Access. Объекты MS Access.

Access работает со следующими объектами:

– *таблицами*;
– *формами*;
– *запросами*;
– *отчетами*.

Кроме того, пользователи могут работать еще с тремя объектами: страницами, макросами и модулями.

Модуль – это программа, написанная на языке Access Basic или VBA.

Таблицы являются базовыми объектами MS Access, в них хранятся все данные, имеющиеся в базе. Кроме того, таблицы хранят также структуру базы (поля, их типы и свойства).

Все остальные объекты являются производными и создаются на базе ранее подготовленных таблиц.

Форма не является самостоятельным объектом Access, она предназначена для оптимизации пользовательского ввода и просмотра данных в таблицах или запросах.

Запросы служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию.

Отчет – это особая форма представления данных, предназначенная для вывода на печать.

Страница – это особый объект, выполненный в коде HTML, размещаемый на Web-странице и передаваемый клиенту вместе с ней. Страницы доступа к данным осуществляют интерфейс между клиентом, сервером и БД, размещенной на сервере.

Макрос – это набор специальных макрокоманд (например, *ОткрытьФорму*, *ПечататьОтчет* и т.п.).

Каждый объект MS Access имеет имя. Длина имени любого объекта MS Access (таблицы, формы и т.д.) не более 64 произвольных символов (за исключением точки и некоторых служебных знаков). В имя могут входить пробелы и русские буквы.

Работа с каждым объектом проводится в отдельном окне, причем предусмотрено два режима работы:

1) оперативный режим – когда в окне решаются задачи ИС, т.е. просмотр, изменение, выбор информации;

2) режим конструктора – когда создается или изменяется макет, структуру объекта.

Кроме этого, в файл базы данных входит еще один документ, имеющий собственное окно: Схема данных. В этом окне создаются, просматриваются, изменяются и разрываются связи между таблицами. Эти связи помогают контролировать данные, создавать запросы и отчеты.

Окно приложения MS Access и окно базы данных. В окне базы данных расположены три командные кнопки (*Создать*, *Открыть* и *Конструктор*) и семь вкладок (по числу объектов Access) с корешками: *Таблицы*, *Запросы*, *Формы*, *Отчеты*, *Страницы*, *Макросы* и *Модули*.

Если выбрана вкладка *Таблица*, в окне этой вкладки отображается список существующих таблиц данной БД, а также основные варианты создания таблиц.

Чтобы открыть существующую таблицу, надо выделить ее имя в этом списке и нажать кнопку *Открыть*.

Чтобы включить в БД новую таблицу, надо нажать кнопку *Создать* или выбрать нужный вариант из списка.

Чтобы исправить макет существующей таблицы, надо выделить ее имя в списке и нажать кнопку *Конструктор*.

Такие же операции выполняются со всеми другими объектами Access.

Типы данных в MS Access.

В MS Access допускаются следующие типы данных:

1) *текстовый* – произвольная последовательность символов длиной до 255. Используется для текста и чисел, не участвующих в вычислениях;
2) *числовой* – любое число, используется для числовых данных, используемых для проведения расчетов;
3) *денежный* – тип данных для хранения денежных значений;
4) *дата/время* – используется для дат (в диапазоне лет от 100 до 9999) и времени суток;

5) *логический* – логические значения, а также поля, которые могут содержать одно из двух возможных значений (TRUE или FALSE);

6) *счетчик* – используется для автоматической нумерации записей;

7) *поле МЕМО* – специальный тип данных для хранения больших объемов текста (до 65 535 символов), комбинация текста и чисел;

8) *поле объекта OLE* – используется для хранения объектов OLE, внедренных в базу данных Access;

9) *гиперссылка* – спициальное поле для хранения URL для Web-объектов Интернета;

10) *мастер подстановок* – создает поле, в котором предлагается выбор значений из списка, или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы. Выбор этого параметра в списке в ячейке запускает мастера подстановок, который определяет тип поля.

Как создавать объекты в MS Access. Любой объект (таблицу, запрос, форму, отчет, страницу) можно создать либо вручную, либо с помощью Мастера.

Функции Мастера предлагают самый быстрый способ создания объектов. Она позволяет создавать новые объекты на базе одного или нескольких готовых объектов.

Рассмотрим последовательность шагов при создании в базе данных новой таблицы с помощи Мастера.

1. В окне базы данных щелкните на ярлычке **Таблицы**, потом выберите вариант *Создание таблицы с помощью мастера*. Или же щелкните на кнопке **Создать**, а затем выделите в *окне диалога* **Новая таблица** вариант *Мастер таблиц* и щелкните на кнопке ОК.
2. Щелкните на переключателе **Деловые** или **Личные**, чтобы просмотреть соответствующий набор примеров таблиц.
3. В *списке* **Образцы таблиц** выделите таблицу, *поля* которой вы хотите просмотреть, затем переместите заинтересовавшие вас поля из *списка* **Образцы полей** в список **Поля новой таблицы** и щелкните на кнопке Далее.

Чтобы переименовать любое из добавленных вами полей, выделите его и воспользуйтесь кнопкой **Переименовать поле**. При этом Access сохранит тип данных поля.

4. В текстовом поле задайте имя новой таблицы, выберите способ определения *ключевого поля* и щелкните на кнопке Далее.
5. В списке **Новая таблица** «*имя таблицы*» настройте все необходимые связи, после чего щелкните на кнопке Далее.
6. Выберите один из вариантов дальнейших действий: **Измененить структуру таблицы**; **Ввести данные непосредственно в таблицу** или **Ввести данные в таблицу с помощью формы, создаваемой мастером**. Щелкните на кнопке Готово.

Access создаст новую таблицу и сохранит ее на диске. Если вы выбрали вариант модификации структуры, откроется окно конструктора. При выборе варианта непосредственного ввода данных запустится режим таблицы, а если вы решили вводить данные в форму, она будет создана автоматически.

Создание производных объектов – запросов, форм, отчетов и страниц – ведется примерно по той же схеме.

Если вы хотите изменить (модифицировать) макет существующего объекта, выберите этот объект в списке соответствующей вкладки окна базы данных и нажмите кнопку Конструктор. На экране появится окно с макетом объекта (такое же, как и при создании). Вы можете исправить макет и сохранить его под тем же или под другим именем.

Конструктор представляет собой наиболее подходящий инструмент для формирования структуры *таблицы*. В этом режиме можно добавлять и удалять поля. Здесь же можно получить полную информацию о свойствах самой таблицы и всех ее полей.

Рассмотрим последовательность шагов при создании в базе данных новой таблицы с помощи Конструктора.

1. В окне базы данных щелкните на вкладке **Таблицы**, потом выберите вариант *Создание таблицы в режиме конструктора*. Или же щелкните на кнопке **Создать**, а затем в окне диалога **Новая таблица** выберите вариант **Конструктор**, после чего щелкните на кнопке ОК.
2. В столбце **Имя поля** введите имя нового поля, нажмите клавишу Tab и укажите для него тип в столбце **Тип данных**.
3. В столбце **Описание** введите информацию, которая будет отображаться в строке состояния, когда точка вставки располагается в этом поле таблицы. На вкладке **Общие** укажите **Размер поля**, т.е. число символов; **Формат поля** и **Маску ввода**, т.е. метод отображения и допустимые символы; **Подпись** для режима таблицы; **Значение по умолчанию**, **Условие на значение** и другие свойства.

Для текстового и числового поля надо указать размер поля, причем для текста – это допустимая длина значения (например, 20 или 40 символов), а для числа – формат представления в машине (байт, целое (два байта), длинное целое и т.д.).

Для поля «Дата/время» обязательно надо указать формат, чтобы система знала, как обрабатывать вводимые данные. Например, если выбрать «Краткий формат даты», система будет ожидать от вас ввода именно даты (в русской версии – ДД.ММ.ГГГГ), а, если выбрать «Краткий формат времени», в этом поле придется набирать ЧЧ:ММ (часы и минуты).

В качестве значения свойства **Условие на значение** вы можете указать правило верификации, т.е. логическое выражение, которое должно принимать значение TRUE (Истина) при вводе данных в это поле.

В свойстве **Обязательное поле** можно указать «Да» (пустые значения не допускаются) или «Нет» (пустые значения допускаются).

4. Щелкните на первой пустой строке таблицы полей и создайте следующее поле базы данных, затем повторите действия 2 и 3. Чтобы вставить поле между двумя существующими, щелкните на кнопке **Добавить строки** панели инструментов Конструктор таблиц.
5. Чтобы выделить то поле таблицы, которое нужно сделать ключевым, щелкните на селекторе, расположенном левее имени поля. После этого нужно щелкнуть на кнопке **Ключевое поле** панели инструментов.
6. Щелкните на кнопке **Сохранить** панели инструментов Конструктор таблиц, введите имя в текстовое поле **Имя таблицы** окна диалога **Сохранение**, после чего щелкните на кнопке ОК.

Access создаст новую таблицу и запишет ее на диск.

Работа с таблицей. После того как таблица создана, ее необходимо заполнить данными.

Ввод данных в *таблицу* или *форму* осуществляется так же, как и ввод данных в лист Excel или таблицу Word.

Шаг за шагом.

1. Откройте *форму* или *таблицу* и перейдите к нужной *записи.* Если необходимо создать новую запись, щелкните на кнопке **Новая запись** панели инструментов.

2. Введите данные в поле. Если необходимо добавить объект OLE, вставьте данные из другого источника или дайте команду **Объект** меню **Вставка** и выберите тип объекта и его имя в соответствующих окнах диалога. Если данные являются элементами предопределенного набора, выбирайте нужный вариант в раскрывающемся списке.

3. Нажмите клавишу Tab для перехода к следующему полю. Повторяйте действие 2 и нажимайте Tab для каждого поля. При необходимости перехода к предыдущему полю нажмите клавиши Shift + Tab.

По достижении последнего поля записи очередное нажатие клавиши Tab переместит точку ввода в первое поле следующей записи.

Если вам не нравится ширина столбца таблицы (например, она слишком велика или, наоборот, мала и скрывает часть данных), ее можно уменьшить или увеличить, точно так же, как вы изменяли ширину столбца в Excel.

Перемещение по таблице. В строке состояния указывается общее число записей в таблице и номер текущей записи. Текущая запись отмечается стрелкой в левой части окна (в области маркировки записей). Для перемещения по таблице служат кнопки переходов в строке состояния (слева направо, переход к первой записи таблицы, к предыдущей записи, к следующей записи и к последней записи таблицы).

Чтобы переместить текстовый курсор в произвольную ячейку таблицы, можно просто щелкнуть на ячейке мышью.

Кроме того, по таблице можно перемещаться с помощью клавиш Tab, Shift + Tab, стрелок курсора.

Редактирование таблицы. При вводе данных используется основной стандарт редактирования. Закончив ввод или модификацию данных в конкретном поле, нажмите Tab или Enter (или щелкните мышью в другой ячейке таблицы).

Для ввода (внедрения) объекта OLE надо щелкнуть правой кнопкой на его поле и выбрать OLE-сервер из списка. После внедрения OLE-объекта, отображаемым в таблице значением его поля, будет название соответствующего OLE-сервера (например, «Документ Microsoft Word»). Чтобы просмотреть или отредактировать объект (или, скажем, чтобы воспроизвести звукозапись), надо, как всегда, дважды щелкнуть на этом названии).

Фильтр. Работая с таблицей в оперативном режиме, вы можете установить фильтр, т.е. задать логическое выражение, которое позволит выдавать на экран только записи, для которых это выражение принимает значение TRUE («Истина»).

Фильтр набирается в окне фильтра. Чтобы установить (или изменить фильтр), выберите команду [*Записи* / *Изменить фильтр*...], отредактируйте фильтр и выберите команду [*Записи* / *Применить фильтр*]. Чтобы восстановить показ всех записей, выберите команду [*Записи* / *Показать все записи*].

Операции с записями и столбцами. С помощью команд меню и кнопок панели инструментов вы можете проводить множество стандартных операций с записями и столбцами: вырезать и копировать в буфер, удалять записи, скрывать столбцы и т.д.

5.2. СВЯЗЬ МЕЖДУ ТАБЛИЦАМИ И ЦЕЛОСТНОСТЬ ДАННЫХ

Общие положения. Между одноименными полями двух таблиц MS Access устанавливает связь. Это означает, что при формировании запроса к этой паре таблиц Access сможет объединить строки таблиц, в которых значения поля совпадают. В общем случае допускается связь по двум, трем и более одноименным полям.

Кроме того, Access позволяет вручную установить связь между таблицами по разноименным полям, однако этой возможностью лучше не пользоваться: это запутывает и аналитиков, и пользователей.

Целостность данных. Итак, если установлена связь между двумя таблицами (автоматически или вручную), данные из обеих таблиц можно объединять. Иногда этого достаточно, однако при создании серьезных баз данных придется позаботиться о дополнительных средствах контроля связанных данных, вводимых в разные таблицы.

Механизм, который обеспечивает согласованность данных между двумя связанными таблицами, называется так: поддержка целостности данных.

Если пользователь включил механизм поддержки целостности, он должен одновременно указать тип связи: «Один-к-Одному» или «Один-ко-Многим».

Целостность данных означает:

1) в связанное поле подчиненной таблицы можно вводить только те значения, которые имеются в связанном поле главной таблицы;

2) из главной таблицы нельзя удалить запись, у которой значение связанного поля совпадает хотя бы с одним значением того же поля в подчиненной таблице.

При попытке нарушить эти запреты MS Access выдает сообщение об ошибке.

Каскадное обновление и удаление записей. Включив механизм поддержки целостности, вы можете (но не обязаны) потребовать, чтобы при модификации данных система запускала следующие процессы:

– каскадное обновление связанных полей;

– каскадное удаление связанных записей.

Каскадное обновление означает, что изменение значения связанного поля в главной таблице автоматически будет отражено в связанных записях подчиненной таблицы.

Каскадное удаление означает, что при удалении записи из главной таблицы, из подчиненной таблицы будут удалены все записи, у которых значение связанного поля совпадает с удаляемым значением.

Техника создания связей между таблицами. Когда между двумя таблицами устанавливается связь, величины одной таблицы ставятся в соответствие величинам из другой таблицы. Чтобы создать связь, в одной или обеих таблицах должно

быть поле, принимающее уникальные значения во всех записях. В родительской таблице поле связи обычно индексировано (как правило, оно является ключевым полем), и в соответствие ему ставится поле дочерней таблицы (оно называется внешний ключ).

Создавая связь, необходимо в окне диалога **Изменение связей** настроить режим обеспечения целостности данных. Обеспечение целостности позволяет избежать наличия несвязанных данных в дочерней таблице, которые образуются в том случае, когда для какой-либо записи нет соответствия в поле родительской таблицы. Если установлен флажок **Каскадное обновление связанных полей**, то при изменении величины связанного поля в родительской таблице изменяются величины полей и во всех соответствующих записях дочерней таблицы. Если установлен флажок **Каскадное удаление связанных записей**, то при удалении записи в родительской таблице будут удалены и соответствующие записи в дочерней таблице. Если эти флажки не были установлены, а обеспечение целостности данных тем не менее было включено, то вы не сможете изменить идентификационное поле родительской таблицы, а также не сможете удалить в ней запись, если в дочерней таблице имеются связанные с этой записью данные.

Создание связей с режимом обеспечения целостности данных.

1. Активизировав окно базы данных, щелкните на кнопке **Схема данных** панели инструментов, чтобы открыть окно **Схема данных** или Сервис / Схема данных.
2. Щелкните на поле родительской таблицы и «дотащите» связь от него до поля дочерней таблицы.
3. Откроется окно диалога **Изменение связей**. В его нижней половине при необходимости включите режим обеспечения целостности данных и настройте правила обновления дочерней таблицы.
4. Для завершения процесса создания связи щелкните на кнопке ОК и закройте окно **Схема данных**.

Связь отображается в виде линии, соединяющей две таблицы. Любую связь можно выделить и удалить нажатием клавиши Delete. Кроме того, можно щелкнуть на линии правой кнопкой мыши, чтобы раскрыть контекстное меню, а затем выбрать команду **Изменить связь**, чтобы открыть окно диалога **Изменение связей**. Кнопка окна диалога **Изменение связей** позволяет настроить тип объединения. Можно щелкнуть на этой кнопке, чтобы настроить для запросов применяемый по умолчанию метод объединения.

5.3. ЗАПРОС-ВЫБОРКА В MS ACCESS

В общем случае запрос – это вопрос о данных. Существуют разные типы запросов (на добавление записей, изменение, объединение), рассмотрим простейший тип: запрос-выборку.

Запрос-выборка – это производная таблица, которая содержит те же структурные элементы, что и обычная таблица (столбцы-поля и строки), и формируется на основе фактических данных системы. При создании макета запроса (т.е. производной таблицы) в общем случае нам необходимо выполнить четыре базовые операции:

1) указать системе, какие поля и из каких таблиц мы хотим включить в запрос;

2) описать вычисляемые поля, т.е. поля, значения которых являются функциями значений существующих полей (например, стоимость продукции – это произведение цены на количество);

3) описать групповые операции над записями исходных таблиц (например, нужно ли объединить группу записей с одним и тем же кодом клиента в одну и просуммировать стоимость заказанной им продукции);

4) указать условие отбора, т.е. сформулировать логическое выражение, которое позволит включить в выборку только записи, удовлетворяющие какому-то условию.

Как создать запрос-выборку. В общем случае для создания произвольного запроса используется универсальный язык SQL. В предложении этого языка (SELECT – Выбрать) можно описать все четыре базовые операции: какие поля и откуда выбрать, какие вычислить, как их сгруппировать (просуммировать, пересчитать, найти среднее и т.п.) и при каких условиях включить записи в выборку. Однако в реальности пользоваться этим языком могут только специалисты (или очень грамотные пользователи). А для обычных людей разработчики придумали упрощенный механизм создания запроса, называемый QBE (Query By Example – Запрос по образцу). Вам предлагают бланк QBE – некую модель, заготовку запроса, и на этом бланке, пользуясь определенными соглашениями, вы сообщаете системе о своих планах: помечаете поля, вводите выражения, значения и т.п. На основании заполненного вами бланка система сама создает соответствующее предложение SELECT и сама выполняет его.

Создание запроса с помощью конструктора. Создать новый запрос можно либо с помощью мастера, либо в режиме конструктора. Оба эти способа достаточно просты, однако создание запросов с помощью мастера выполняет пошаговое формирование запроса. Мастер позволяет настроить дополнительные параметры.

1. Щелкните на ярлычке **Запросы** в окне базы данных, потом выберите вариант *Создание запроса в режиме конструктора*. Или же щелкните на кнопке **Создать**, в окне диалога **Новый запрос** выберите вариант **Конструктор**. Далее в окне диалога **Добавление таблицы** дважды щелкните на именах нужных таблиц, а затем на кнопке **Закрыть**.

При добавлении связанных таблиц Access автоматически создает линию объединения между этими таблицами.

Чтобы создать связь между двумя таблицами, перетащите ключевое поле родительской таблицы на связываемое поле дочерней таблицы.

2. В списках полей таблиц дважды щелкните на тех полях, которые нужно использовать в качестве элементов запроса.
3. В столбцах всех полей, которые нужно сортировать, щелкните на строке **Сортировка** и выберите вариант **По возрастанию** или **По убыванию**.

Поля сортируются в указанном порядке справа налево. Самое левое сортируемое поле является ключевым полем сортировки. Чтобы изменить порядок расположения полей, перетащите заголовки столбцов.

4. Если какие-либо поля запроса должны быть скрыты, сбросьте для них флажки в строке **Вывод на экран**.

Такие поля будут использоваться в запросе, но не будут отображаться на экране.

5. Введите необходимые выражения в качестве условий отбора полей. Щелкните на кнопке **Сохранить** панели инструментов мастера запросов. Введите имя формы в текстовое поле **Имя запроса** окна диалога **Сохранение** и щелкните на кнопке ОК.

Создание запроса с помощью мастера. Мастер запросов предлагает новичкам быстрый способ создания запросов, однако большая гибкость достигается в режиме конструктора.

1. На вкладке **Запросы** окна базы данных выберите вариант *Создание запроса с помощью мастера*. Или же щелкните на кнопке **Создать**, а затем выберите вариант **Простой запрос**.
2. В раскрывающемся списке **Таблицы и запросы** выберите таблицу или запрос.
3. Дважды щелкните на тех полях, которые должны содержаться в запросе.

Для добавления всех необходимых полей повторяйте шаги 2 и 3. Затем щелкните на кнопке Далее.

4. Если нужно отобразить все записи, щелкните на переключателе **Подробный**.

Для вывода только общей информации о записях, такой, как их суммарное количество, щелкните на переключателе **Итоговый** и на кнопке **Итоги**, после чего укажите, какие итоговые значения необходимо вычислять, и щелкните на кнопке ОК.

5. На следующих шагах работы мастера настройте параметры группировки записей и введите имя запроса. Окончив настройку, щелкните на кнопке **Готово**. Сохраните и закройте запрос.

5.4. ОТЧЕТЫ

Отчет – это особая форма представления данных, предназначенная для вывода на печать. Как правило, для формирования отчета создают запрос, в котором собирают данные из разных таблиц, с включением вычисляемых полей, группировкой, условиями отбора (любая операция необязательна). Далее, по общим правилам MS Access, на базе такого запроса проектируют отчет, который позволяет:

- представить данные в удобной для чтения и анализа форме;
- сгруппировать записи (по нескольким уровням) с вычислением итоговых и средних значений;
- включить в отчет и напечатать графические объекты (например, диаграммы).

Создание отчета с помощью Автоотчета. Быстрее всего отчет создается с помощью инструмента **Автоотчет**. Если щелкнуть на кнопке **Создать** на вкладке **Отчеты**, пользователю станут доступны две модификации этого инструмента: **Автоотчет: в столбец** и **Автоотчет: ленточный**. Отчет типа «в столбец» выводит записи по одной в строке, а ленточный отчет отображает каждое поле записи на отдельной строке рядом с подписью. Отчет в столбец намного более распространен, чем ленточный.

1. Перейдите на вкладку **Отчеты** окна базы данных и щелкните на кнопке **Создать**.
2. В окне диалога **Новый отчет** в поле со списком выберите в качестве источника данных отчета таблицу или запрос.
3. Дважды щелкните на строке **Автоотчет: в столбец** или **Автоотчет: ленточный.** Access сформирует отчет, расположит в нем все необходимые поля и выведет его на экран в режиме предварительного просмотра.
4. Чтобы изменить структуру отчета, перейдите в режим конструктора. Чтобы сохранить отчет, выберите команду **Сохранить** меню **Файл**.

Создание отчета с помощью конструктора. В редких случаях может оказаться необходимым создать отчет непосредственно в режиме конструктора на основе пустого бланка. Однако, поскольку процесс добавления элементов управления достаточно утомителен, удобнее сформировать автоотчет, а затем удалить ненужные элементы. Альтернативным способом является использование мастера отчетов. Если нужно сделать отчет с небольшим числом элементов управления или если он должен содержать только подчиненные отчеты, воспользуйтесь конструктором отчетов.

1. Щелкните в окне базы данных на ярлычке **Отчеты**, потом выберите вариант *Создание отчета в режиме конструктора*. Или же щелкните на кнопке **Создать**, а затем в окне диалога **Новый отчет** выберите вариант **Конструктор**.
2. Если вы собираетесь использовать в качестве источника данных таблицу или запрос, выберите соответствующий объект в списке окна диалога Новый отчет. Если отчет используется только как оболочка для серии подчиненных отчетов, оставьте поле источника данных пустым.
3. Дважды щелкните на пункте Конструктор. Access откроет бланк отчета в режиме конструктора.
4. Добавьте надписи, текстовые поля и другие элементы управления.
5. Щелкните на кнопке **Сохранить** панели инструментов Конструктор отчетов или выберите команду **Сохранить** меню Файл, после чего в окне диалога **Сохранение** введите имя нового отчета и щелкните на кнопке ОК.

Создание отчета с помощью мастера отчетов. Мастер отчетов предоставляет пользователю максимальную гибкость в процессе создания отчетов. Здесь можно указать, какие поля из одной или нескольких таблиц или запросов должны быть внесены в отчет, как группировать и сортировать данные. При необходимости пользователь может добавить итоговые поля, а также настроить стиль и оформление отчета. Выполнение тех же операций в режиме конструктора намного более трудоемко. Однако, если нужно создать совсем простенький отчет, стоит воспользоваться инструментом Автоотчет.

Действия мастера отчетов и доступные в нем параметры динамически изменяются в соответствии с вариантами, указанными пользователем на предыдущих шагах.

1. В окне базы данных щелкните на ярлычке **Отчеты**, потом выберите вариант *Создание отчета с помощью мастера*. Или же щелкните на кнопке **Создать**, а затем выберите вариант **Мастер отчетов**.
2. В раскрывающемся окне **Создание отчетов** выберите таблицу или запрос в качестве источника данных. В списке **Доступные поля** дважды щелкните на тех полях, которые должны отображаться в отчете. Переключайтесь с помощью списка **Таблицы и запросы** на любые связанные с основной таблицы и добавляйте дополнительные поля. Затем щелкните на кнопке **Далее**.

Если на экране появилось сообщение об ошибке, информирующее о наличии в отчете полей из несвязанных таблиц, дважды щелкните в списке Выбранные поля на ошибочно добавленных полях для того, чтобы удалить их.

3. В зависимости от набора выбранных полей на двух следующих шагах мастер запросит правила группировки данных. Если поля относятся к разным таблицам, Access попросит указать, как следует выводить информацию. Выберите таблицу, по которой следует группировать данные, и щелкните на кнопке **Далее**.

На следующем шаге выделите поля группировки данных и установите их взаимный приоритет. Если нужно изменить интервал группировки (указать первую букву для текстовых данных, выбрать группировку по месяцу или году для данных типа «дата» или определить диапазон чисел), щелкните на кнопке **Группировка**, настройте необходимые параметры и щелкните на кнопке ОК. Затем щелкните на кнопке **Далее**.

4. Выберите поля, по которым должны сортироваться данные, при этом самое верхнее поле станет первичным ключом сортировки. Щелчком на кнопке режима сортировки выберите вариант упорядочения – по возрастанию или по убыванию.

Если при выполнении операций пункта 3 данные были сгруппированы, то, вероятно, окажется доступной кнопка **Итоги**. Щелкните на ней, чтобы суммировать данные, определить среднее, минимум или максимум значений числового поля и установить режим отображения отдельных записей и итоговой информации. Щелкните на кнопке ОК, чтобы вернуться в окно настройки методов сортировки, в котором после этого следует щелкнуть на кнопке **Далее**.

5. Выберите макет и ориентацию отчета, затем щелкните на кнопке **Далее**. Укажите стиль оформления отчета и щелкните на кнопке **Далее**.

6. На последнем шаге работы мастера отчетов введите название отчета, после чего щелкните на переключателе **Просмотреть отчет** или **Изменить макет отчета** и, наконец, щелкните на кнопке **Готово**.

Access сконструирует новый отчет и запишет его на диск. Если был выбран вариант **Просмотреть отчет**, Access включит режим предварительного просмотра. В противном случае будет открыто окно конструктора отчета.

Для переключения режимов просмотра отчета выберите соответствующую команду в меню – Вид или в раскрывающемся меню кнопки **Вид** панели инструментов Конструктор отчетов.

СПИСОК ЛИТЕРАТУРЫ

1. Гордиенко, А.В. Базы данных : учебное пособие / А.В. Гордиенко. – Новочеркасск : ЮРГТУ, 2002. – 62 с.
2. Дьяков, И.А. Базы данных: Язык SQL : учебное пособие / И.А. Дьяков. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2004. – 80 с.
3. Информатика : учебник для вузов / под ред. Н.В. Макаровой. – М. : Финансы и статистика, 2007. – 768 с.
4. Кузин, А.В. Базы данных : учебное пособие для вузов / А.В. Кузин. – М. : Академия, 2005. – 320 с.
5. Мясникова, Н.А. Методические указания к лабораторным работам по дисциплине «Базы данных» / Н.А. Мясникова. – Новочеркасск : ЮРГТУ, 2004. – 43 с.
6. Советов, Б.Я. Базы данных. Теория и практика : учебник для вузов / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М. : Высшая школа, 2005. – 463 с.
7. Татаренко, С.И. Базы данных : учебное пособие / С.И. Татаренко. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2004. – 80 с.

СОДЕРЖАНИЕ