Глава 8. Базы данных

ГЛАВА ВОСЬМАЯ

БАЗЫ ДАННЫХ

§ 8.1. ОСНОВНЫЕ ПОНЯТИЯ БАЗ ДАННЫХ

Основные идеи информационной технологии основываются на концепции баз данных (БД), в которых данные должны адекватно отображать реальный мир и удовлетворять информационным потребностям пользователей.

База данных в широком смысле слова — это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области.

Под *предметной областью* подразумевается часть реального мира, подлежащего изучению и использованию для решения различных практических задач.

БД представляют собой разновидность информационной модели, для которой определены такие понятия, как информационный объект, реквизит, отношения и связи.

Информационным объектом называется описание реального объекта, процесса или явления в виде совокупности его характеристик, называемых *реквизитами*. Информационный объект однозначно идентифицируется именем и заданием *ключевого реквизита* (*ключа*). Остальные реквизиты в информационных объектах являются *описательными*.

Пример 8.1. Информационный объект **СТУДЕНТ** имеет реквизиты: номер (номер зачетной книжки — ключевой реквизит), фамилия, имя, отчество, дата рождения, код места обучения.

Информационный объект **ЛИЧНОЕ ДЕЛО** имеет реквизиты: номер студента, домашний адрес, номер аттестата о среднем образовании, семейное положение.

Информационный объект **МЕСТО ОБУЧЕНИЯ** включает реквизиты: код (ключевой реквизит), наименование вуза, факультет, группа.

Информационный объект **ПРЕПОДАВАТЕЛЬ**: код (ключевой реквизит), кафедра, фамилия, имя, отчество, ученая степень, ученое звание, должность.

Отношения, существующие между реальными объектами, определяются в информационных моделях как *связи*. Существует три вида связей: один к одному (1 : 1), один ко многим (1 : ∞) и многие ко многим (∞ : ∞).

Связь *один к одному* определяет соответствие одному информационному объекту X не более одного информационного объекта Y и наоборот.

Пример 8.2. Информационные объекты **СТУДЕНТ** и **ЛИЧНОЕ ДЕЛО** будут связаны отношением *один к одному*. Каждый студент имеет определенные уникальные данные в личном деле.

Связь *один ко многим* предполагает, что одному экземпляру информационного объекта X может соответствовать любое количество информационных объектов Y_1, Y_2, ..., Y_n, но каждый объект Y_i, $i = 1, 2, ..., n$, связан не более чем с одним объектом X.

Пример 8.3. Между информационными объектами **МЕСТО ОБУЧЕНИЯ** и **СТУДЕНТ** следует установить связь *один ко многим*. Одно и то же место обучения может многократно повторяться для различных студентов.

Связь *многие ко многим* предполагает соответствие каждому информационному объекту X_j, $j = 1, 2, ..., m$, любого количества экземпляров объекта Y_i, $i = 1, 2, ..., n$, и наоборот.

Пример 8.4. Информационные объекты **СТУДЕНТ** и **ПРЕПОДАВАТЕЛЬ** имеют связь *многие ко многим*. Каждый студент обучается у множества преподавателей, а каждый преподаватель учит множество студентов.

Теперь определим *базу данных* как информационную модель, состоящую из совокупности связанных информационных объектов в исследуемой предметной области. Ядром любой БД является модель данных, которая представляет собой множество структур данных, ограничений

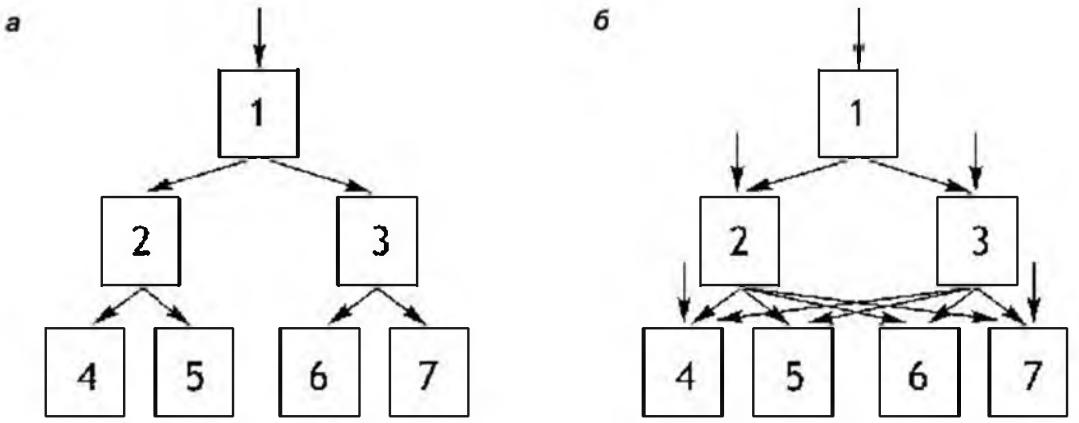


Рис. 8.1

целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных — это совокупность структур данных и операций их обработки.

Различают иерархическую, сетевую и реляционную модели данных.

На рис. 8.1 показаны схемы иерархической и сетевой моделей данных, в которых связи между объектами изображены стрелками.

В *иерархической модели* (рис. 8.1*а*) данные представлены в виде древовидной (иерархической) структуры. Непосредственный доступ возможен лишь к объекту самого высокого уровня. К другим объектам доступ осуществляется по связям от объекта на вершине модели. Таким образом организовано обращение к файлам в операционной системе DOS.

В *сетевых моделях* (рис. 8.1*б*) непосредственный доступ обеспечивается к любому объекту независимо от уровня его расположения в модели, а также по связям к другим объектам. Примером сетевой модели может служить совокупность БД, содержащих информацию о взаимосвязанных производственных цехах (справочные данные о цехах, складах и выпускаемой продукции). Обе эти модели не получили широкого распространения из-за сложности реализации графов в виде машинных структур данных, кроме того, в них сложно осуществить операции поиска информации.

Наибольшее распространение получила третья модель данных — *реляционная модель*, описанию которой будет посвящено содержание § 8.2.

В современной технологии БД предполагается, что создание БД, ее поддержка и обеспечение доступа пользователей к ней осуществляется с помощью специального программного продукта, именуемого системой управления базами данных.

Система управления базами данных (*СУБД*) — это комплекс программных и языковых средств, необходимых для создания БД, поддержания их в рабочем состоянии и организации поиска в них необходимой информации.

Для организации работ в СУБД предполагается наличие некоторого лица, на которое возлагаются функции администрирования данными, хранимыми в БД.

§ 8.2. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Понятие «реляционный» (англ. *relation* — отношение) связано с разработками Е. Кодда — известного специалиста в области БД.

Реляционная модель данных ориентирована на организацию данных в виде двумерной таблицы (отношения), обладающей следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все столбцы в таблице однородные, т. е. все элементы в столбце имеют одинаковый тип (числовой, текстовый и т. д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Таблицу называют *отношением*, строки — *записями*, столбцы — *полями*, а строку заголовков — *схемой отношения*.

Для однозначного определения записей служит *первичный ключ*, который может быть простым или составным.

Простой ключ (*ключевое поле*) — это поле, каждое значение которого однозначно определяет запись.

УСПЕВАЕМОСТЬ1

№ зачётки	Фамилия	Предмет	Балл	Ауд	Корпус
21143	Иванов	Информ	97	135	А
21133	Перова	Эконом	90	221	Б
22201	Сёмин	Матем	40	158	А
21143	Иванов	Матем	96	221	Б
21133	Перова	Информ	88	158	А
22201	Сёмин	Эконом	90	135	А

Рис. 8.2

Составной ключ содержит несколько полей со значениями, однозначно определяющими запись.

Например, схема отношения-таблицы **УСПЕВАЕМОСТЬ1** — на рис. 8.2 будет следующей:

УСПЕВАЕМОСТЬ1(№ зачётки, Фамилия, Предмет, Балл, Ауд, Корпус)

Здесь **УСПЕВАЕМОСТЬ1** — отношение, а **№ зачетки, Фамилия** и т. д. — поля. На рис. 8.2 видно, что ни одно из полей не может быть ключевым, поскольку все они имеют повторяющиеся элементы, поэтому здесь используется составной ключ, содержащий поля **Фамилия, Предмет**. Чтобы устранить многократное повторение данных и уменьшить объем памяти, занимаемой БД, прибегают к *нормализации*, которая заключается в разбиении исходных таблиц на более мелкие, руководствуясь методом нормальных форм.

Е. Коддом выделены три нормальные формы отношений и предложен метод, позволяющий любое отношение преобразовать к третьей, самой совершенной нормальной форме.

Суть метода состоит в последовательном переводе таблицы из одной нормальной формы в другую, причем каждая последующая устраняет определенный вид функциональной зависимости между полями таблицы.

Первая нормальная форма. Отношение называется приведенным к первой нормальной форме, если все его атрибуты неделимы. Разработчики БД изначально строят исходное отношение так, чтобы оно было в первой нормальной форме.

а

УСПЕВАЕМОСТЬ2

Фамилия	Предмет	Балл	Ауд	Корпус
Иванов	Информ	97	135	А
Перова	Эконом	90	221	Б
Сёмин	Матем	40	158	А
Иванов	Матем	96	221	Б
Перова	Информ	88	158	А
Сёмин	Эконом	90	135	А

б

СПИСОК

№ зачётки	Фамилия
21143	Иванов
21133	Перова
22201	Сёмин

Рис. 8.3

Вторая нормальная форма. Для приведения отношений ко второй нормальной форме введем понятие функциональной зависимости.

Функциональная зависимость полей — это зависимость, при которой в строке определенному значению ключевого поля соответствует только одно значение неключевого поля. Функционально неключевое поле зависит от составного ключа, но не зависит от любого поля, входящего в составной ключ.

Например, в отношении **УСПЕВАЕМОСТЬ1(№ зачётки, Фамилия, Предмет, Балл, Ауд, Корпус)** первичным ключом является совокупность полей **Фамилия, Предмет**. Поля **Балл, Ауд** функционально полно зависят от составного ключа.

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждое неключевое поле функционально полно зависит от составного ключа. Например, отношение **УСПЕВАЕМОСТЬ1(№ зачётки, Фамилия, Предмет, Балл, Ауд, Корпус)** находится в первой нормальной форме, однако оно не находится во второй нормальной форме, так как поле **№ зачётки** не имеет полной функциональной зависимости от составного ключа. Для перевода этого отношения во вторую нормальную форму необходимо исключить из него поле **№ зачётки**, так как оно функционально зависит от поля **Фамилия**, а не от всех полей составного ключа **Фамилия, Предмет**, т. е. исход-

а

УСПЕВАЕМОСТЬ3

Фамилия	Предмет	Балл	Ауд
Иванов	Информ	97	135
Перова	Эконом	90	221
Сёмин	Матем	40	158
Иванов	Матем	96	221
Перова	Информ	88	158
Сёмин	Эконом	90	135

б

МЕСТО

Ауд	Корпус
135	А
138	А
221	Б

Рис. 8.4

ное отношение необходимо разбить на два связанных по полю **Фамилия** отношения **УСПЕВАЕМОСТЬ2(Фамилия, Предмет, Балл, Ауд, Корпус)** и **СПИСОК(№ зачётки, Фамилия)**, как это показано на рис. 8.3.

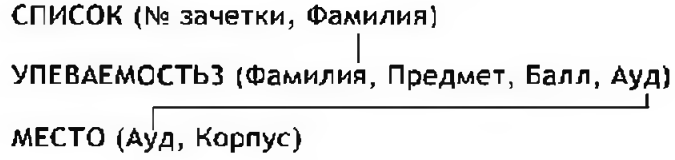
В таблице **СПИСОК** ключевым может быть поле **№ зачётки** или поле **Фамилия**, образующее простой ключ.

Третья нормальная форма. Третья нормальная форма позволяет устранить транзитивную зависимость. В отношении имеет место *транзитивная зависимость*, если существуют два поля, в которых первое зависит от ключа, а второе зависит от первого. Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме, и каждое неключевое поле не транзитивно зависит от ключа.

Например, в отношении **УСПЕВАЕМОСТЬ2(Фамилия, Предмет, Балл, Ауд, Корпус)** поле **Ауд** зависит от составного ключа **Фамилия, Предмет**, а поле **Корпус** — от поля **Ауд**, т. е. имеет место транзитивность. Для ее устранения отношение **УСПЕВАЕМОСТЬ2(Фамилия, Предмет, Балл, Ауд, Корпус)** расщепляется на два отношения: **УСПЕВАЕМОСТЬ3(Фамилия, Предмет, Балл, Ауд)** и **МЕСТО(Ауд, Корпус)**, изображенные на рис. 8.4*а* и 8.4*б*. Связь между этими отношениями-таблицами осуществляется по полю **Ауд**.

Процесс нормализации заканчивается созданием *схемы данных*, в которой указываются все нормализованные таблицы с их полями и взаимосвязями между ними. Указываются типы взаимосвязей.

В нашем случае схема данных будет иметь вид



Связь по полю **Фамилия** между отношениями **СПИСОК** и **УСПЕВАЕМОСТЬ3** относится к типу *один ко многим* (1 : ∞), так как одному элементу этого поля из таблицы **СПИСОК** (рис. 8.3*б*) соответствует несколько (два) элементов поля **Фамилия** из таблицы **УСПЕВАЕМОСТЬ3** (рис. 8.4*а*). Аналогичный тип связи устанавливается по полю **Ауд** между отношениями **МЕСТО** и **УСПЕВАЕМОСТЬ3**. Таблица **УСПЕВАЕМОСТЬ3** называется родительской, а таблицы **СПИСОК** и **МЕСТО** — дочерними. Поля **Фамилия** и **Ауд** в таблице **УСПЕВАЕМОСТЬ3** являются внешними ключами, а в таблицах **СПИСОК** и **МЕСТО** — первичными ключами.

Важным условием эффективной работы БД является обеспечение целостности ее данных.

Целостность данных рассматривается на уровне таблицы и на уровне межтабличных связей.

В первом случае не допускается нарушение уникальности первичного ключа при добавлении данных.

Требование целостности данных на уровне межтабличных связей состоит в том, что для каждого значения внешнего ключа главной родительской таблицы должны найтись строки в дочерней таблице с таким же значением ключа.

Целостность данных обеспечивается корректным выполнением транзакции.

Транзакцией называется совокупность операций с БД, которые должны быть выполнены так, чтобы БД оказалась в непротиворечивом состоянии, т. е. не должна быть нарушена целостность данных.

Например, если в таблице **УСПЕВАЕМОСТЬ3** удаляется запись с фамилией Иванов, то в связанной таблице **СПИСОК** для сохранения непротиворечивости данных должна быть также удалена запись с этой фамилией.

Анализ характеристик различных СУБД показал следующее. Существенно уступая СУБД FoxPro по произво-

дительности, две СУБД: Access и Paradox for Windows — обладают гораздо лучшими возможностями обеспечения целостности и безопасности данных. Эти показатели несколько лучше у СУБД Access, поэтому основное внимание в последующем материале будет уделено изучению основ работы в СУБД Access.

В качестве основных этапов обобщенной работы в любой СУБД можно выделить следующие:

- построение таблиц базы данных;
- сортировка, поиск и фильтрация данных;
- создание запросов к базе данных;
- формирование и вывод отчета.

§ 8.3. ПОСТРОЕНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ

Для построения и последующей обработки в СУБД Access предлагаются три нормализованные таблицы (отношения) **СЕССИЯ**, **АНКЕТА** и **СТИПЕНДИЯ**, изображенные на рис. 8.5*а*, 8.6*а*, 8.7*а*. Соответствующие схемы отношений на рис. 8.5*б*, 8.6*б*, 8.7*б* представляют собой списки полей, размещенных в отдельных столбцах. Выделенные жирным шрифтом поля являются ключевыми и представляют собой простые ключи.

Таблица-отношение **СЕССИЯ** содержит поля **Матем**, **Эконом**, **Информ** с баллами, соответствующими отличной (балл выше 92), хорошей (балл в пределах от 80 до 92), удовлетворительной (балл в пределах от 53 до 79) и

а

СЕССИЯ

Фамилия	Пол	Матем	Эконом	Информ	Оценка
Ежова	ж	66	50	83	УДО
Иванов	м	96	94	97	ОТЛ
Перова	ж	85	90	88	ХОР
Петров	м	72	80	95	УДО
Сёмин	м	40	90	70	УДО
Серова	ж	82	91	96	ХОТ

б

Рис. 8.5

а

АНКЕТА

Фамилия	№ зачётки	Дата рождения	Группа
Ежова	21125	12.05.1983	ЭУ1
Иванов	21143	20.10.1984	ЭУ2
Перова	21133	16.08.1981	ЭУ1
Петров	21045	11.08.1982	ЭУ2
Сёмин	22201	24.09.1983	ЭУ2
Серова	22177	03.11.1979	ЭУ1

б

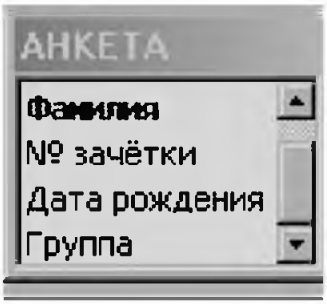


Рис. 8.6

а

СТИПЕНДИЯ

Оценка	Размер
ОТЛ	1000
УДО	0
ХОР	600
ХОТ	800

б

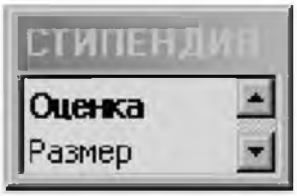


Рис. 8.7

неудовлетворительной (балл ниже 53) оценкам по предметам «Математика», «Экономика», «Информатика».

В качестве ключевого выбрано поле **Фамилия**. Поле **Оценка** содержит общие лингвистические оценки студентов по всем предметам: ОТЛ — все оценки отличные; ХОТ — имеются хорошие и отличные оценки; ХОР — все хорошие оценки; УДО — имеются неудовлетворительные или/и удовлетворительные оценки.

В таблице **АНКЕТА** поле **Фамилия** выбрано в качестве первичного ключа, с помощью которого можно обеспечить связь с таблицей **СЕССИЯ**.

В отношении **СТИПЕНДИЯ** поле **Оценка** является первичным ключом, а в отношении **СЕССИЯ** — внешним ключом.

Два отношения **АНКЕТА** и **СТИПЕНДИЯ** можно связать отношением **СЕССИЯ**, в котором поля **Фамилия** и **Оценка** представляют собой внешние ключи, являющиеся первичными ключами отношений **АНКЕТА**, **СТИПЕНДИЯ** (рис. 8.8).

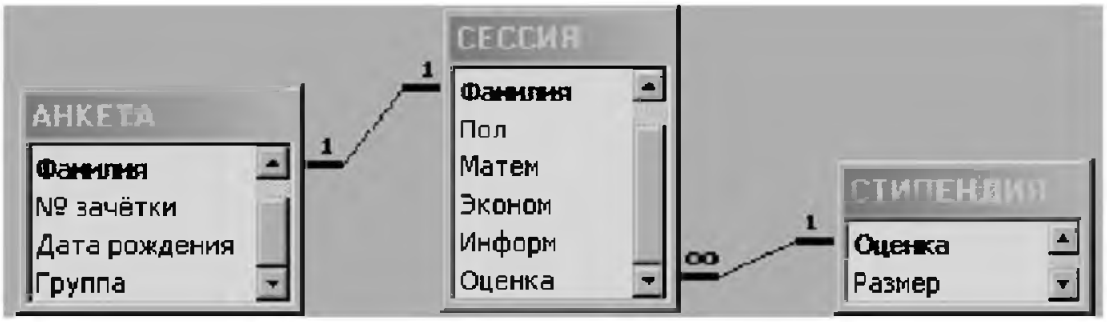


Рис. 8.8

Проанализируем связи отношений. Таблицы **АНКЕТА** и **СЕССИЯ** имеют связь *один к одному* (1 : 1), поскольку каждой записи первой таблицы соответствует не более одной записи второй таблицы и наоборот.

Между таблицами **СТИПЕНДИЯ** и **СЕССИЯ** устанавливается связь *один ко многим* (1 : ∞). В данном случае первичный ключ **Оценка** таблицы **СТИПЕНДИЯ** связывается с внешним ключом таблицы **СЕССИЯ**. При этом каждой записи таблицы **СТИПЕНДИЯ** соответствует несколько записей таблицы **СЕССИЯ**. Введем понятие родительского и дочернего отношения-таблицы.

Таблица **СЕССИЯ** является родительской по отношению к таблицам **АНКЕТА** и **СТИПЕНДИЯ** потому, что она связана с ними через внешние ключи. В свою очередь, таблицы **АНКЕТА** и **СТИПЕНДИЯ** называются дочерними.

Сразу после запуска Access в диалоговом окне *Microsoft Access* предлагается создать новую БД, запустить мастер по созданию БД или открыть существующую БД. Пусть переключатель поставлен в положение *Новая база данных*. На экране появляется окно базы данных (см. рис. 8.9), содержащее 6 основных окон, вызываемых кнопками, из названия которых ясно, с какими объектами предстоит иметь дело.

Таблицы создаются пользователем для одного и нескольких наборов данных из предметной области.

Запросы формируются пользователем для выборки нужных данных из одной или нескольких связанных таблиц.

Формы служат для ввода, просмотра и корректировки БД в удобном виде, соответствующем привычному документу. Формы также могут использоваться для создания панелей управления в приложении пользователя.

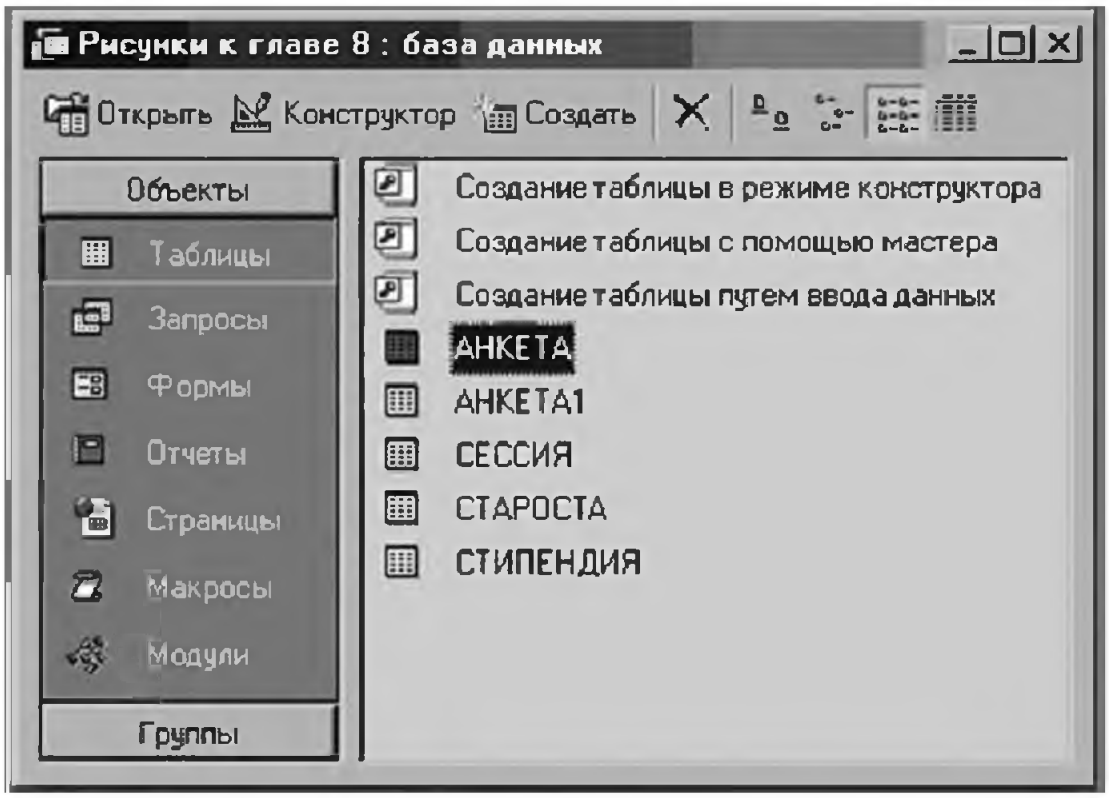


Рис. 8.9

Отчеты предназначены для формирования выходного документа, который следует вывести на печать.

Страницы осуществляют связь переданной Web-страницы с базой данных, находящейся в сервере.

Макросы содержат описание действий, которые должны быть выполнены в ответ на некоторое событие. Каждое действие реализуется макрокомандой. Выбор макрокоманд и задание параметров, используемых ими при выполнении, является достаточно простой задачей. Макрос позволяет объединить разрозненные операции обработки данных в приложении.

Модули содержат программы на языке Visual Basic for Application (VBA), разрабатываемые пользователем для реализации нестандартных процедур при создании приложений.

Для построения таблицы нужно установить вкладку *Таблица* и нажать кнопку *Создать*. Откроется диалоговое окно *Новая таблица*, в котором предлагаются три способа создания таблицы: *Режим таблицы*, *Конструктор* и *Мастер таблицы*. Рассмотрим первые два способа, используя данные таблиц **СЕССИЯ** (рис. 8.5*а*) и **АНКЕТА** (рис. 8.6*а*).

Поле1	Поле2	Поле3	Поле4	Поле5	Поле6

Рис. 8.10

Режим таблицы. При выборе этого режима на экране появляется заготовка таблицы (рис. 8.10), в которой пользователь удаляет имена полей **Поле1**, **Поле2**, **Поле3** и т. д., заменяет их на новые (**Фамилия**, **Пол**, **Матем**, **Эконом**, **Информ**, **Оценка**) и вводит в пустые ячейки необходимую информацию. Ввод и перемещение данных по ячейкам таблицы выполняются так же, как и в Excel. Сохраняется полученная таблица командой **Файл, Сохранить** с последующим вводом имени таблицы, которое затем отображается на вкладке *Таблицы* диалогового окна БД (рис. 8.9).

Режим конструктора. Он наиболее предпочтителен для эффективной работы в СУБД. Если выбрать опцию **Конструктор** в диалоговом окне *Новая таблица*, то на экране появляется окно конструктора таблицы **АНКЕТА**, представленное на рис. 8.11.

Для создания таблицы нужно описать поля, т. е. заполнить как минимум две графы: **Имя поля**, **Тип данных.** Графа **Описание** не является обязательной, она предназначена для подсказки. В нашем случае описание полей таблицы **АНКЕТА** будет иметь вид, приведенный на рис. 8.11.

После ввода типа данных поля задают свойства поля на вкладке *Общие*.

Дадим краткое описание тех свойств, которые чаще всего нуждаются в задании.

Размер поля — максимальное количество символов, которое содержат текстовые данные поля.

Для числовых данных свойство *Размер поля* имеет следующие характеристики:

- с плавающей точкой (8 байт) — 15 десятичных знаков;
- с плавающей точкой (4 байта) — 7 десятичных знаков;
- длинное целое (4 байта) — 10 десятичных знаков;
- целое (2 байта) — от −32 768 до 32 767;

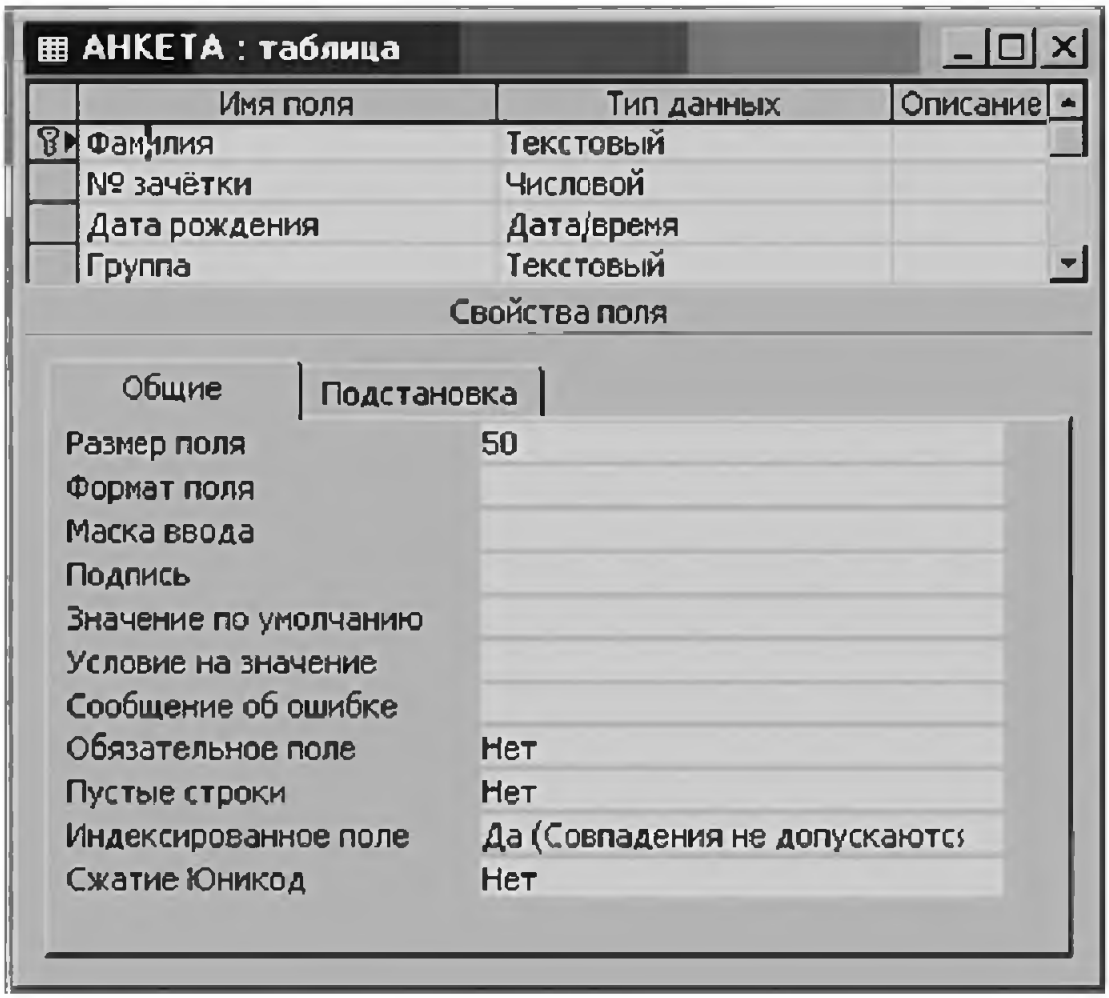


Рис. 8.11

- байт — от 0 до 255;
- денежный (4 байта) — 15 знаков до запятой и 4 знака после нее.

Формат поля — спускающийся список операторов, которые могут быть применены к данному типу поля. Опишем наиболее употребительные форматы для двух типов данных:

- числовой — основной, денежный, фиксированный, с разделителем разрядов, процентный;
- дата/время — краткий, средний и длинный форматы даты и времени.

Значение по умолчанию. Заданные в этом свойстве данные будут появляться во всех случаях, а в некоторых из них возможна замена этих данных.

Обязательное поле. Устанавливается *Да* или *Нет*, чтобы сообщить Access, можно или нет пользователю оставить поле пустым при вводе новой записи.

Программе Access также необходимо сообщить, какое из полей предлагается использовать в качестве ключевого. В нашем случае ключевым будет поле **Фамилия**. В результате проделанной работы на экране появляется таблица с полями **Фамилия, № зачетной книжки, Дата рождения, Группа**, предназначенная для ввода данных.

§ 8.4. СОРТИРОВКА, ПОИСК И ФИЛЬТРАЦИЯ ДАННЫХ

Обработку данных можно осуществлять непосредственно в режиме таблицы. Использование *режима таблицы* — это самый простой способ изменения и поиска данных. Под изменением понимается копирование, обновление, удаление, добавление и сортировка данных в таблице. Поиск данных, помимо традиционной процедуры нахождения нужной информации, включает и фильтрацию.

Режим таблицы устанавливается следующим образом. В окне *Таблицы* базы данных выделяется имя таблицы и нажимается кнопка *Открыть*. Используя команды из меню *Формат*, можно выполнять форматирование таблицы: изменять высоту строк и ширину столбцов, переставлять или убирать с экрана столбцы, применять другой шрифт, скрывать или выводить линии сетки. Команды, находящиеся в меню, позволяют добавить новую запись, изменить, заменить, копировать, вставить и находить нужные данные.

Простейший способ сортировки состоит в следующем. В нужном столбце щелчком выделяется любой элемент и нажимается кнопка панели инструментов *Сортировка по возрастанию* или *Сортировка по убыванию*. Другой вид сортировки совмещен с операцией фильтрации.

Фильтрация — это отбор записей, удовлетворяющих принятому критерию. Фильтрация производится с помощью трех фильтров: *фильтр во выделенному, обычный фильтр, расширенный фильтр*.

Фильтр по выделенному обеспечивает самую простую фильтрацию, критерием которой является значение выделенной ячейки. Например, если в поле **Группа** таблицы **АНКЕТА** выделить ячейку со значением «ЭУ1», то, применяя

данный фильтр, получим записи, содержащие информацию о группе ЭУ1.

Команда **Записи, Изменить фильтр** дает возможность перейти к формированию условий отбора в окне обычного фильтра. С его помощью можно формировать более сложные критерии отбора, использующие логические функции И, ИЛИ и операции отношения (больше, меньше и т. д.).

Гораздо большими возможностями обладает *расширенный фильтр*. Он запускается командой **Записи, Фильтр, Расширенный фильтр**. Расширенный фильтр во многом напоминает бланк запроса, но по сравнению с последним имеет ряд недостатков. Расширенный фильтр может работать только с одной таблицей и не сохраняет свою форму после завершения работы.

§ 8.5. СОЗДАНИЕ ЗАПРОСОВ

Под запросом понимается операция отбора данных из одной или нескольких таблиц. В Access имеется удобное для пользователя графическое средство формирования запроса по образцу QBE (Query By Example), а также язык структурированных запросов SQL (Structured Query Language). Приведем основные типы запросов.

- *запросы на выборку* — выбираются записи, удовлетворяющие условиям отбора;
- *запросы на создание таблицы* — основаны на запросе-выборке, но, в отличие от него, результат запроса сохраняется в новой таблице;
- *запросы на обновление, добавление, удаление* — это запросы-действия, в результате выполнения которых изменяются данные в таблицах.

Для создания запросов надо выбрать вкладку *Запросы* окна базы данных, а затем щелкнуть на кнопке *Создать*.

Появится диалоговое окно *Новый запрос*, предоставляющее возможность вызвать Мастер простого и перекрестного запроса, выборку повторяющихся записей, а также Конструктор запросов. Простой и перекрестный запросы, а также выборка повторяющихся записей применяются

довольно редко, поэтому основное внимание сосредоточим на Конструкторе запросов, содержащем бланк для формирования запроса по образцу QBE.

8.5.1. СРЕДСТВА СОЗДАНИЯ ЗАПРОСОВ

Конструктор запросов запускается следующим образом. В окне *Запросы* нажимается кнопка *Создать*, чтобы открыть диалоговое окно *Новый запрос*. В этом окне выделяется опция **Конструктор**, в появившемся окне *Добавление таблицы* из списка выбирается нужная таблица и нажимается кнопка *Добавить*. Подобные действия, примененные к таблице **СЕССИЯ**, выводят на экран окно конструктора запросов, представленное на рис. 8.12.

Окно конструктора запросов разделено на две панели.

Верхняя панель содержит схему данных запроса, которая включает список полей выбранной для запроса таблицы **СЕССИЯ**.

Нижняя панель называется бланком запроса по образцу (QBE) или просто бланком запроса. Каждый столбец бланка запроса соответствует одному полю таблицы, которое

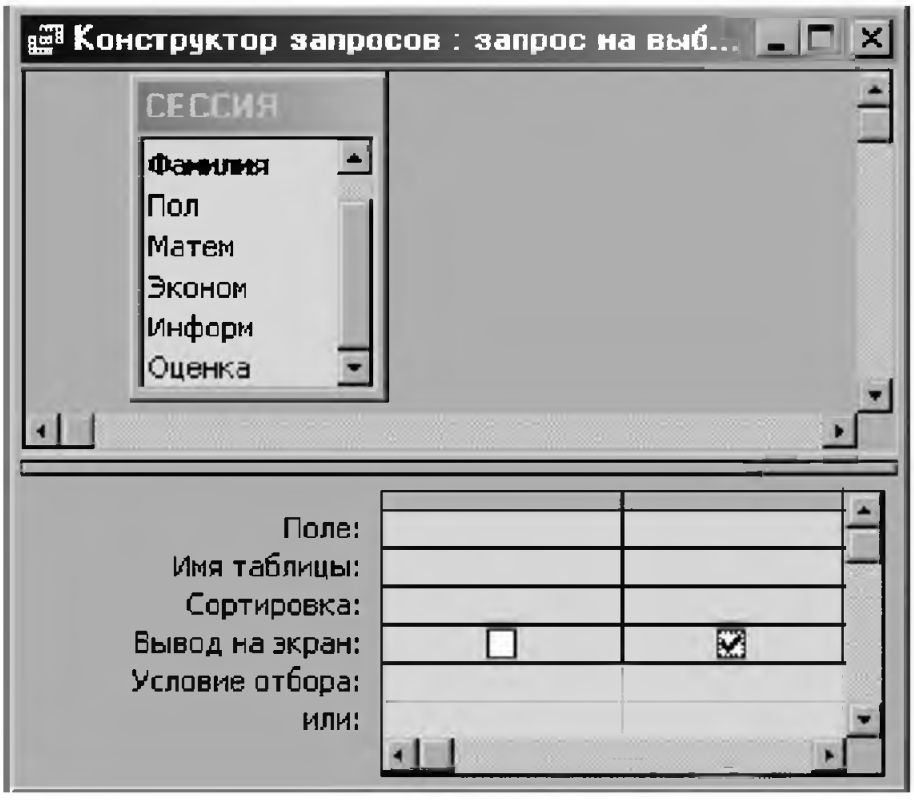


Рис. 8.12

участвует в запросе. При заполнении бланка запроса необходимо в строках:

- *Поле* и *Имя таблицы* — включить соответственно имена полей и таблиц, используемых в запросе;
- *Сортировка* — указать порядок сортировки (по убыванию или возрастанию) записей результата;
- *Вывод на экран* — отметить поля, которые должны быть включены в результирующую таблицу;
- *Условие отбора* — ввести арифметические или логические формулы для отбора записей;
- *Или* — реализовать взаимодействие между полями по логической схеме ИЛИ.

Остановимся на выражениях, которые записываются в строку *Условия отбора*. Они могут состоять из литералов, операторов, констант, идентификаторов, функций.

Литерал — это число, символьная константа, название поля или дата, вводимые в поля таблицы. Литерал типа *Число* вводится без ограничителя, типа *Дата* имеет ограничитель # (#12.11.98#), а типа *Символьная константа* — ограничитель кавычки (") или апострофы (') ("Иванов", 'Петров'). Литералы типа *Название поля* заключаются в квадратные скобки ([ ]).

Оператор указывает действие, которое должно быть выполнено с элементами полей. Имеются следующие группы операторов:

- арифметические: * (умножение), + (сложение), – (вычитание), / (деление), ^ (возведение в степень);
- соединения частей текста: &, например 'Юрий'&' "Иванов';
- сравнения: < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), = (равно), <> (не равно);
- операторы SQL: Like, Between... And, In.

Оператор Like содержит специальные символы (?, *, #) и маски, необходимые для определения строк с неизвестными символами.

Символ ? обозначает любой одиночный символ. Например, если неизвестно, как написать: Carl или Karl, то условие можно задать как `Like "?arl"`.

Символ * обозначает последовательность символов.

Символ # обозначает любую неизвестную цифру.

Оператор Between... And заменяет знаки «больше или равно» и «меньше или равно». Например, условие **Between 1981 And 1994** эквивалентно условию **>= 1981 And <= 1994**.

Оператор In позволяет использовать списки значений. Например, выражение **In ("Иванов И.П.", "Петров Ю.И.")** позволяет отобрать записи с указанными фамилиями и инициалами.

Идентификатор — это имя поля, заключенное в квадратные скобки. Например, **[Оценка]** относится к полю **Оценка** таблицы **СЕССИЯ**.

Функция — это имя специальной программы, предназначенной для выполнения групповой операции — группировки — для любого поля бланка запросов. Под группировкой будем понимать операцию над совокупностью данных одинакового типа одного поля с помощью какой-либо функции. Перечислим наиболее употребительные функции:

Sum — вычисляет сумму значений поля в каждой группе;

Avg — вычисляет среднее арифметическое всех значений поля в каждой группе;

Count — подсчитывает число записей поля в каждой группе;

Min — находит наименьшее значение поля внутри каждой группы;

Max — находит наибольшее значение поля внутри каждой группы.

Для включения полей из таблиц в соответствующие столбцы запроса достаточно перетащить эти поля из списка полей таблицы в схеме данных или выбрать из списка в строке *Поле* бланка запроса.

Рассмотрим примеры заполнения бланка запроса для основных видов запросов.

8.5.2. ЗАПРОСЫ НА ВЫБОРКУ

Приведем примеры заполнения бланков запросов, выполняющих некоторые задачи отбора.

Пример 8.5. В таблице **АНКЕТА** выбрать записи группы ЭУ1 (см. рис. 8.13).

Рис. 8.13

Пример 8.6. В таблице **АНКЕТА** выбрать записи с информацией о студентах Иванове и Сёмине. Для этого необходимо в строке *или* и следующей строке столбца **Фамилия** расположить фамилии **"Иванов"** и **"Сёмин"** или в строке *Условия отбора* записать выражение **"Иванов" Or "Сёмин"**, реализующее логическое условие ИЛИ (рис. 8.14).

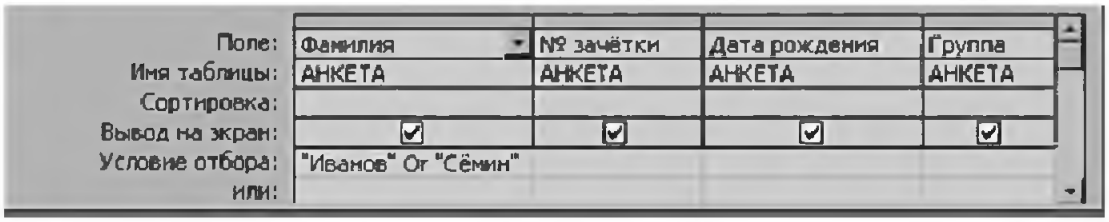


Рис. 8.14

Пример 8.7. Из таблицы **АНКЕТА** вывести информацию о студентах моложе 1981 года рождения. Для этого надо записать условие **=>#01.01.1982#** в поле **Дата Рождения** (рис. 8.15).

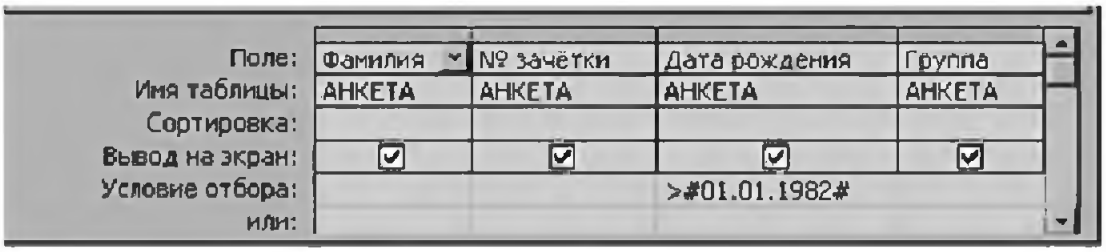


Рис. 8.15

Пример 8.8. Чтобы в таблице **СЕССИЯ** упорядочить в алфавитном порядке записи со студентами мужского пола (рис. 8.16), следует соответственно заполнить строки первого (*Поле* — **Фамилия**, *Сортировка* — По возрастанию) и второго (*Поле* — **Пол**, *Условие отбора* — **"м"**) столбца.

Рис. 8.16

Пример 8.9. Чтобы в таблице **СЕССИЯ** выбрать фамилии отличников во всех трех столбцах (**Матем**, **Эконом**, **Информ**), критерий отбора (> 92) следует расположить в одной строке *Условие отбора*, тем самым реализуя между этими полями логическую операцию И (рис. 8.17).

Рис. 8.17

Пример 8.10. Для получения из таблицы **СЕССИЯ** фамилий неуспевающих студентов условия отбора следует расположить в разных строках столбцов **Матем**, **Эконом**, **Информ**, начиная со строки *или* (рис. 8.18).

Рис. 8.18

В отличие от расширенного фильтра, в бланке запроса требуется дополнительно указывать во всех столбцах имя таблицы и устанавливать флажок *Вывод на экран*, чтобы вывести все поля новой таблицы. Полученный результат

сохраняется в окне *Запросы* окна базы данных. Значительный интерес представляет запрос, сопровождающийся вычислениями содержимого полей. Результат вычисления образует поле в таблице, создаваемой по запросу. При вычислениях могут использоваться арифметические выражения и встроенные функции Access.

Вычисляемое выражение вводится в пустую ячейку строки *Поле* таблицы QBE. Имя вычисляемого поля записывается перед выражением и отделяется от него двоеточием.

Пример 8.11. В таблице **СЕССИЯ** вычислить средние баллы студентов в дополнительном поле **Средний**. На рис. 8.19 приводится бланк запроса с вычисляемым полем.

В дополнительном столбце бланка запроса в строке *Поле* напечатано выражение

```
Средний: (([Матем] + [Эконом] + [Информ])/3,
```

в котором **Средний** — название вычисляемого поля, а выражение, следующее за двоеточием, — формула для вычисления среднего. В этой формуле в квадратные скобки заключены заголовки полей таблицы, участвующих в

а

б

Рис. 8.19

вычислении среднего. Получаемая при таком запросе таблица показана на рис. 8.19*б*.

В дополнительном столбце можно так же, как и в Excel, установить стипендию, например:

- 300 руб., если средний балл выше 92;
- 200 руб., если средний балл в пределах от 80 до 92;
- 0, в остальных случаях.

Для этого в Access воспользуемся функцией IIf, аналогичной функции ЕСЛИ из Excel.

В строке *Поле* дополнительного столбца требуется записать следующее выражение:

```
Стипендия: IIf(([Матем]+[Эконом]+[Информ])/
3>92;300, IIf(([Матем]+[Эконом]+[Информ])/
3>80;200;0)))
```

8.5.3. ИТОГОВЫЕ ЗАПРОСЫ

Итоговые запросы позволяют вычислить или определить с помощью функций **Sum**, **Avg**, **Count**, **Min**, **Max** значения по группам данных. Для нахождения итоговых значений надо нажать кнопку *Групповые операции* Σ на панели инструментов, чтобы в бланке запроса появилась строка *Групповая операция*, которая, помимо функций, содержит установки:

Условие — для ввода условия в строку *Условие отбора*;

Выражение — для ввода математических выражений в строку *Поле*, в которых используются одна или несколько итоговых функций. Приведем пример составления итогового запроса.

Пример 8.12. В таблице **СЕССИЯ** определить количество студенток.

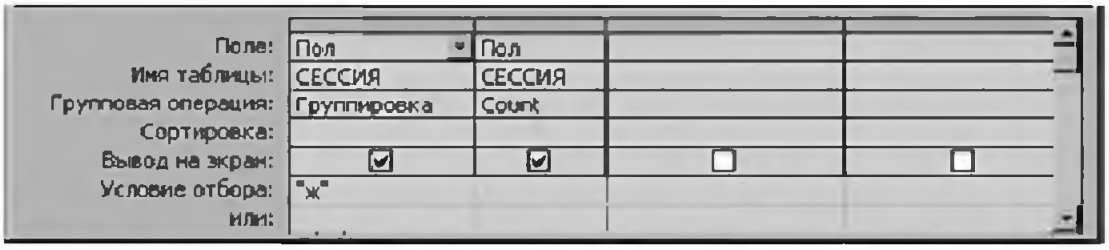
Реализация этого запроса приводится на рис. 8.20*а*.

При таком запросе к таблице **СЕССИЯ** ответом будет одна строка данных, приведенных на рис. 8.20*б*.

8.5.4. МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ

Часто требуется из нескольких таблиц получать данные одновременно, выполнять многотабличные запросы. В этом случае таблицы, содержащие необходимые данные, должны быть связаны с помощью ключевых полей.

а



б

	Пол	Count-Пол
▶	ж	3

Рис. 8.20

Пример 8.13. Вывести фамилии студенток группы ЭУ1 и размер начисленной им стипендии. Для получения этой информации требуются три таблицы: **АНКЕТА**, **СЕССИЯ** и **СТИПЕНДИЯ**, которые следует связать по схеме, представленной на рис. 8.8.

В окне конструктора схема связей и бланк запроса будут иметь вид, показанный на рис. 8.21.

Пример 8.14. Определить суммарный размер стипендии, получаемой студентками группы ЭУ1. На рис. 8.22 приводится бланк запроса, реализующего решение поставленной задачи.

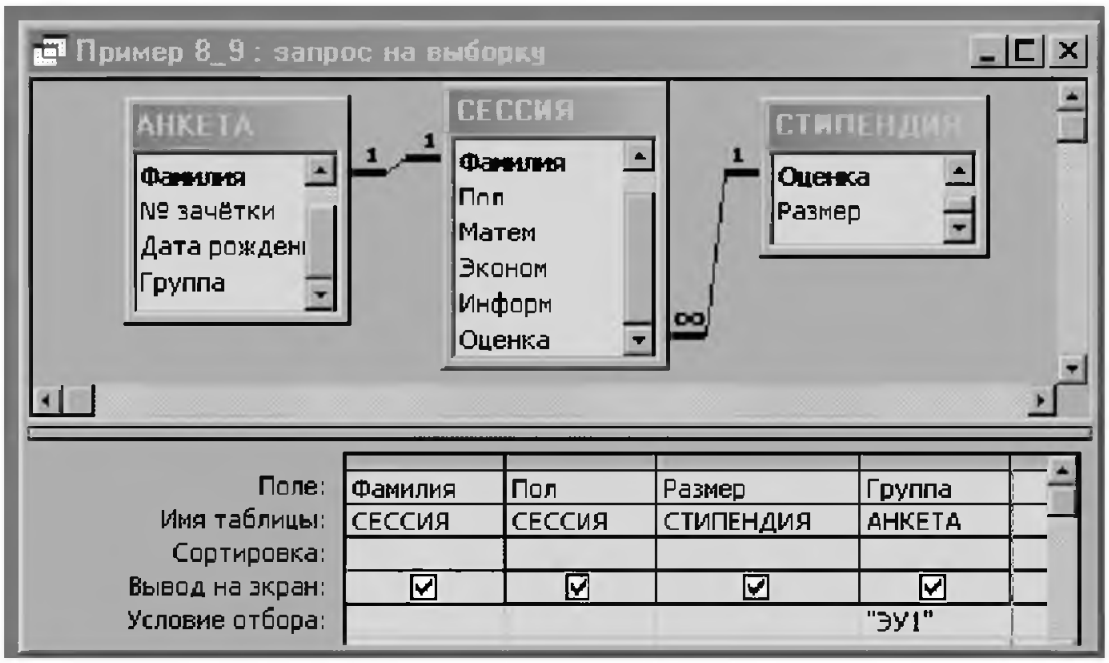


Рис. 8.21

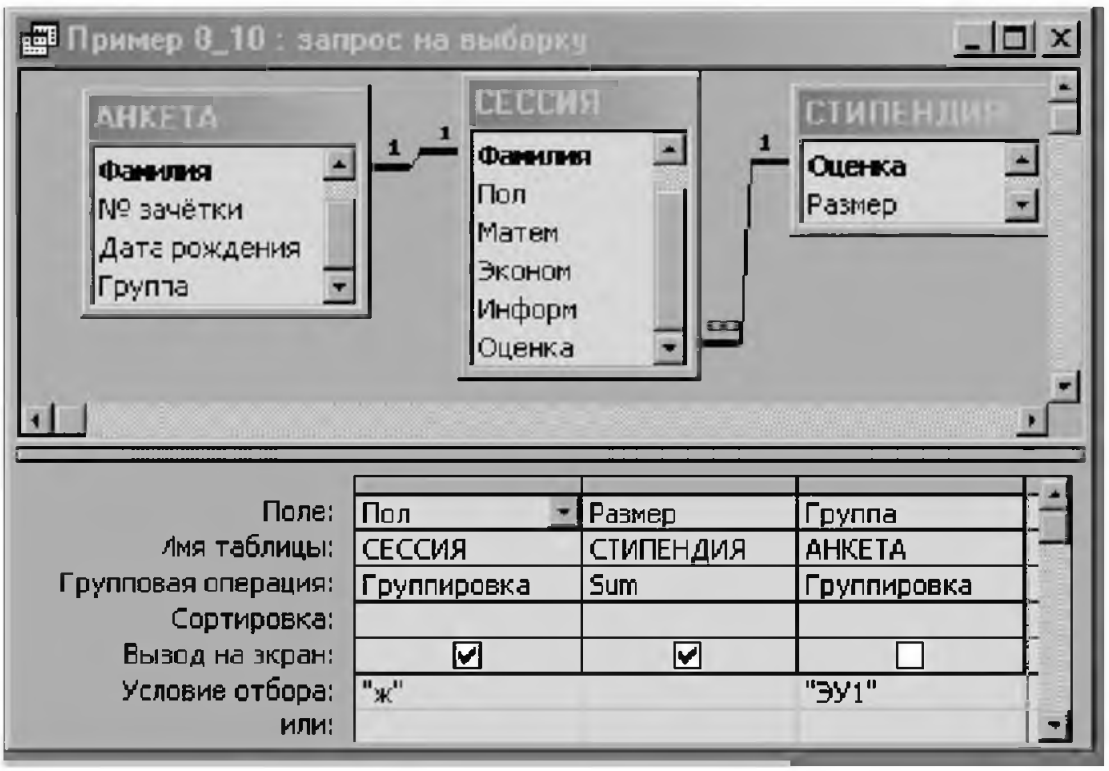


Рис. 8.22

8.5.5. СОЗДАНИЕ SQL-ЗАПРОСОВ

Язык SQL — это наиболее распространенный прикладной язык управления реляционными базами данных архитектуры клиент/сервер. SQL содержит набор из шести групп стандартных команд: 1) описание данных; 2) запрос данных; 3) манипулирование данными; 4) управление курсором; 5) обработка транзакций; 6) администрирование и контроль.

Ограничимся первыми тремя группами команд. Запросы из примера 8.5 можно преобразовать в SQL-запросы, используя шесть команд:

- **SELECT** — список полей, подлежащих выводу;
- **FROM** — имена таблиц;
- **WHERE** — условия отбора;
- **GROUP BY** — условия группировки;
- **HAVING** — условия для результата;
- **ORDER BY** — имя сортируемого столбца [ACS | DESC].

Здесь [ACS | DESC] — указание на то, что сортировка может проводиться по возрастанию (ACS) или по убыванию (DESC).

Опишем более подробно операторы SQL-запроса для примера 8.5 (рис. 8.13):

```
SELECT АНКЕТА.Фамилия, АНКЕТА.[№ зачётки],
АНКЕТА.Группа
FROM АНКЕТА
WHERE (((АНКЕТА.Группа)="ЭУ1"));
```

Оператор **SELECT** выполняет выборку данных полей **Фамилия, № зачётки, Группа** из (команда **FROM**) таблицы **АНКЕТА**, отмеченных «галочкой» в строке *Вывод на экран* запроса на рис. 8.13.

За оператором **WHERE** следует условие отбора **(((АНКЕТА.Группа)="ЭУ1"))** — выборка студентов группы ЭУ1.

Аналогичный вид имеют SQL-запросы для примера 8.6 (рис. 8.14):

```
SELECT АНКЕТА.Фамилия, АНКЕТА.[№ зачётки],
АНКЕТА.[Дата рождения], АНКЕТА.Группа
FROM АНКЕТА
WHERE (((АНКЕТА.Фамилия)="Иванов" Or (АНКЕ-
ТА.Фамилия)="Сёмин"));
```

Для примера 8.7 (рис. 8.15):

```
SELECT АНКЕТА.Фамилия, АНКЕТА.[№ зачётки],
АНКЕТА.[Дата рождения], АНКЕТА.Группа
FROM АНКЕТА
WHERE (((АНКЕТА.[Дата рождения])>#1/1/1982#));
```

Для примера 8.8 (рис. 8.16):

```
SELECT СЕССИЯ.Фамилия, СЕССИЯ.Пол, СЕССИЯ.Матем,
СЕССИЯ.Эконом, СЕССИЯ.Информ, СЕССИЯ.Оценка
FROM СЕССИЯ
WHERE (((СЕССИЯ.Пол)="м"))
ORDER BY СЕССИЯ.Фамилия;
```

За оператором сортировки **ORDER BY** указано поле **Фамилия**, элементы которого сортируются в алфавитном порядке. Продолжим построение SQL-запросов.

Для примера 8.9 (рис. 8.17):

```
SELECT СЕССИЯ.Фамилия
FROM СЕССИЯ
WHERE (((СЕССИЯ.Матем)>92)
```

```
AND ((СЕССИЯ.Эконом)>92)
AND ((СЕССИЯ.Информ)>92));
```

Для примера 8.10 (рис. 8.18):

```
SELECT СЕССИЯ.Фамилия
FROM СЕССИЯ
WHERE (((СЕССИЯ.Матем)<53)
OR ((СЕССИЯ.Эконом)<53)
OR ((СЕССИЯ.Информ)<53));
```

Для примера 8.11 (рис. 8.19*а*):

```
SELECT СЕССИЯ.Фамилия,
(([Матем]+[Эконом]+[Информ])/3) AS Средний
FROM СЕССИЯ;
```

Приведем итоговый SQL-запрос для примера 8.12 (рис. 8.20*а*):

```
SELECT СЕССИЯ.Пол,
Count(СЕССИЯ.Пол) AS [Count-Пол]
FROM СЕССИЯ
GROUP BY СЕССИЯ.Пол
HAVING (((СЕССИЯ.Пол)="ж"));
```

После оператора **SELECT** вычисляется количество элементов поля **Пол** таблицы **СЕССИЯ**: `Count(СЕССИЯ.Пол)`, с образованием в выходной таблице поля с именем `Count_Пол`. Далее осуществляется группировка в поле **Пол** (`GROUP BY СЕССИЯ.Пол`) по новому признаку «ж» (`HAVING(((СЕССИЯ.Пол)="ж"))`). При этом в поле **Пол** результирующей таблицы выводится `"ж"`.

Приведем многотабличные итоговые SQL-запросы для примера 8.13 (рис. 8.21):

```
SELECT СЕССИЯ.Фамилия, СЕССИЯ.Пол,
СТИПЕНДИЯ.Размер, АНКЕТА.Группа
FROM СТИПЕНДИЯ INNER JOIN (АНКЕТА INNER JOIN
СЕССИЯ ON АНКЕТА.Фамилия=СЕССИЯ.Фамилия) ON
СТИПЕНДИЯ.Оценка=СЕССИЯ.Оценка
WHERE (((СЕССИЯ.Пол)="ж")
AND ((АНКЕТА.Группа)="ЭУ1"));
```

и для примера 8.14 (рис. 8.22):

```
SELECT СЕССИЯ.Пол, Sum(СТИПЕНДИЯ.Размер) AS
[Sum-Размер]
```

```
FROM СТИПЕНДИЯ INNER JOIN (АНКЕТА INNER JOIN
СЕССИЯ ON АНКЕТА.Фамилия=СЕССИЯ.Фамилия) ON
СТИПЕНДИЯ.Оценка=СЕССИЯ.Оценка
GROUP BY СЕССИЯ.Пол, АНКЕТА.Группа
HAVING (((СЕССИЯ.Пол)="ж")
AND ((АНКЕТА.Группа)="ЭУ1"));
```

Здесь в новом поле **Sum_Размер** помещается результат суммирования данных поля **Размер** таблицы **СТИПЕНДИЯ** (**Sum(СТИПЕНДИЯ.Размер) As Sum_Размер**).

После предложения **FROM** для ключевого поля **Оценка** таблицы **СТИПЕНДИЯ** устанавливается связь (**INNER JOIN**) с полем **Оценка** главной таблицы **СЕССИЯ** (**ON СТИПЕНДИЯ.Оценка=СЕССИЯ.Оценка**), а также связь между ключевыми полями **Фамилия** таблиц **АНКЕТА** и **СЕССИЯ** (**ON СЕССИЯ.Фамилия=АНКЕТА.Фамилия**).

При этом вычисление суммарной стипендии выполняется для студенток (**(СЕССИЯ.Пол)="ж"**) группы ЭУ1 путем группировки поля **Пол** таблицы **СЕССИЯ** и поля **Группа** таблицы **АНКЕТА** (**GROUP BY СЕССИЯ.Пол, АНКЕТА.Группа**) по признаку «ЭУ1» (**HAVING (((СЕССИЯ.Пол)="ж") AND ((АНКЕТА.Группа)="ЭУ1"))**) с выводом последнего в поле **Группа** итоговой таблицы.

§ 8.6. ФОРМИРОВАНИЕ ОТЧЕТОВ

Работа с БД, как правило, завершается формированием отчета, на основании которого выводится печатный документ. Различные средства создания отчетов вызываются нажатием кнопки *Создать* в окне *Отчеты* базы данных. Появляется окно *Новый отчет* с шестью видами отчетов, среди которых наиболее часто используются Мастер и Конструктор отчетов.

В окне *Новый отчет* выберем *Мастер отчетов* и укажем в качестве источника данных таблицу или запрос. В процессе работы с Мастером отчетов многократно появляются окна с названием «Создание отчетов», требующие:

- выбрать поля для отчета;
- задать уровни и интервалы группировки;
- выполнить вычисления итоговых значений;

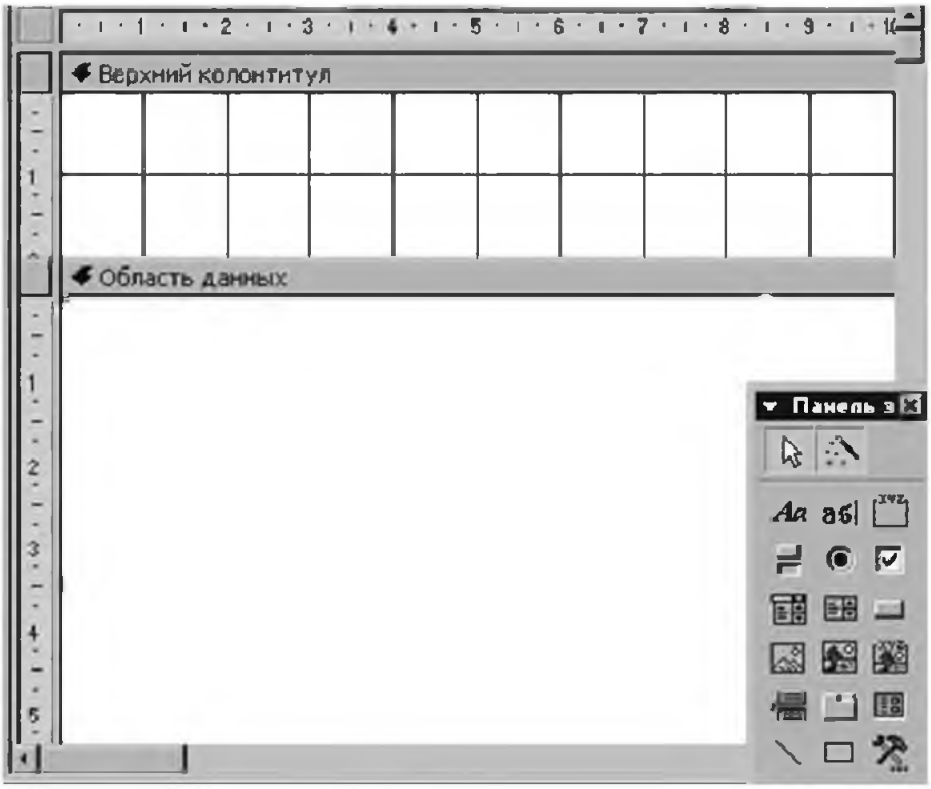


Рис. 8.23

- выбрать макет и стиль отчета;
- присвоить имя и указать режим отображения завершения отчета.

Гораздо более широкие возможности предоставляет *Конструктор отчетов*. Первоначально в окне Конструктора отображаются пустые разделы отчета в виде, представленном на рис. 8.23.

Величину расстояния между разделами можно менять с помощью мыши. Обычно при этом появляется панель элементов. Далее с помощью данных исходной таблицы, клавиатуры и панели элементов пользователь заполняет и оформляет разделы отчета, периодически просматривая его при нажатии кнопки *Предварительный просмотр*. В режиме Конструктора можно пользоваться известными средствами форматирования данных (выравнивание, установка шрифта и т. д.), а также производить вычисление суммы и среднего.