Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Таможенное дело»

Разорёнова Т.Р., Альшевская О.В.

УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ

Учебно-методическое пособие для студентов специальностей

1–96 01 01 “Таможенное дело”
1–26 02 02 “Менеджмент”
1–25 01 08 “Бухгалтерский учет, анализ и аудит”
1–25 01 07 “Экономика и управление на предприятии”,
”Финансовое обеспечение и экономика боевой
и хозяйственной деятельности войск (сил)”

Минск 2010

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Таможенное дело»

Разорёнова Т.Р., Альшевская О.В.

УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ

Учебно-методическое пособие для студентов специальностей

1–96 01 01	“Таможенное дело”
1–26 02 02	“Менеджмент”
1–25 01 08	“Бухгалтерский учет, анализ и аудит”
1–25 01 07	“Экономика и управление на предприятии”, ”Финансовое обеспечение и экономика боевой и хозяйственной деятельности войск (сил)”

Рекомендовано Учебно-методическим объединением высших учебных заведений Республики Беларусь по образованию в области автоматизации технологических процессов и управления в качестве учебно-методического пособия

Минск 2010

УДК 004.65(075.8)
ББК 32.973-018я7
Р17

Рецензенты:
Н.Н. Гурский, Л.И. Дроздович

Разорёнова Т.Р., Альшевская О.В.

Р17 Управление базами данных: учебно-методическое пособие для студентов специальностей 1–96 01 01 “Таможенное дело”, 1–26 02 02 “Менеджмент”, 1–25 01 08 “Бухгалтерский учет, анализ и аудит”, 1–25 01 07 “Экономика и управление на предприятии”, ”Финансовое обеспечение и экономика боевой и хозяйственной деятельности войск (сил)”. – Минск: БНТУ, 2010. – 99 с.

Учебно-методическое пособие предназначено для использования в разделах дисциплин «Компьютерные информационные технологии» и «Технологии организации, хранения и обработки данных», посвященных разработке и использованию баз данных. В пособии рассматриваются реляционные базы данных применительно к СУБД Access: конструирование запросов, отчетов, макросов, элементы автоматизации приложения, язык SQL, уделяется внимание вопросам обеспечения целостности данных и проектирования баз данных.

© Разорёнова Т.Р., Альшевская О.В., 2010

Введение

В процессе сбора и хранения данных некоторым упорядоченным образом получается база данных, которая, в свою очередь, может быть операционной (учитывающую повседневную обработку информации и ее динамическое изменение) или аналитической (отражающую «исторические» данные, используемые для анализа тенденций изменения, принятия решений и разработки стратегий работы предприятия).

В учебно-методическом пособии рассматриваются реляционные базы данных, построенные на реляционной модели, которые разрабатываются на основе таблиц и взаимосвязей между ними, что позволяет получать новые таблицы (отношения) на основе исходных, руководствуясь операциями реляционной алгебры. Таблицы описываются полями (атрибутами), состоят из строк (кортежей) и с помощью уникальных атрибутов (ключей) идентифицируют свои записи. Между таблицами могут существовать связи: один-к-одному, один-ко-многим, многие-ко-многим. Понимание связей приносит большую пользу при работе с запросами SQL к нескольким таблицам. Уделяется большое внимание языку SQL, который является стандартным языком для создания, поддержки и работы с реляционной базой данных.

Обеспечение надежности структуры базы данных основано на корректной разработке таблиц, выборе типов данных для ее атрибутов, устранении аномалий и проведении нормализации модели с целью получения оптимальной структуры таблиц, связанных между собой. В связи с этим одна из тем посвящена вопросам проектирования баз данных, когда в качестве источников для проектирования взяты документы, бланки, таблицы, в которых фиксируются различные данные предметной области.

Для работы с базой данных разрабатываются интерфейсные элементы, позволяющие управлять вводом, отображением и визуализацией данных, с которыми работает пользователь. Применение элементов управления с автоматизацией (разработкой программ на VBA) позволяет продемонстрировать возможности СУБД MS Access не только как настольного приложения по управлению базами данных, но и как клиента в технологиях клиент/серверных систем, способных управлять и использовать серверные приложения офисного пакета. Создание управляющей кнопочной формы, применение макросов превращает разработанный прототип приложения в реальный продукт, способный управлять данными, хранимыми в базе данных.

Особенностью работы с базами данных является получение данных для отчетов, презентаций и анализа (графиков, диаграмм, сводных таблиц). В пособии рассматриваются вопросы разработки отчетов сложной структуры, использованию агрегирующих функций и группировок.

Представленное учебное издание является логическим продолжением и расширением ранее разработанного методического пособия «Технологии организации, хранения и обработки данных», в котором студенты по предложенному описанию создают прототип учебной базы «Студенты», состоящей из трех связанных таблиц, описывающих факультеты, специальности и студентов. Поскольку любая информационная система развивается и претерпевает изменения, авторам показалось необходимым дополнить исходную базу дополнительной информацией. Эти дополнения изменяют не только структуру таблиц, но и позволяют добавить внешние данные, демонстрируя этим возможности запросов-действий и необходимость обеспечения целостности данных. К данному учебно-методическому пособию разработаны электронные документы (доступны в сети БНТУ), которые позволят ускорить набор дополнительных данных и программ, необходимых при изучении тем и выполнения заданий лабораторных работ. По ходу изложения приведены тексты программ и таблицы с дополнительной информацией. В приложении приведены структуры исходных таблиц и данные этих таблиц

Пособие может использоваться студентами специальностей не только экономического, но и технического профиля, которые изучают системы управления базами данных в соответствующих курсах по учебным планам своих специальностей.

Тема 1. Целостность данных

Теоретические сведения

Термин **целостность** используется для описания непротиворечивости данных, хранимых в базе данных. Аспекты целостности необходимо учитывать как при проектировании базы данных, так и во время ее использования, так как в процессе эксплуатации база данных может претерпевать различные изменения: корректируются существующие или добавляются новые записи, добавляются новые таблицы или новые поля в имеющиеся таблицы. Поддержка целостности реализуется с помощью нескольких видов ограничений, накладываемых с целью защиты базы данных от нарушения согласованности сохраняемых в ней данных. К типам поддержки целостности данных относятся:

- обязательные данные (для некоторых полей требуется наличие в каждой записи конкретного и допустимого значения);
- ограничения для доменов полей (определяется область допустимых значений данного поля);
- корпоративные ограничения целостности (требования конкретного предприятия);
- целостность сущностей (первичный ключ таблицы должен иметь уникальное непустое значение в каждой записи);
- ссылочная целостность.

Рассмотрим ссылочную целостность подробнее.

Структура базы данных задается с помощью схемы данных. В ней определяются и запоминаются связи между таблицами. Это позволяет Access автоматически использовать связи, один раз определенные в схеме данных, при конструировании форм, запросов, отчетов на основе взаимосвязанных таблиц. Схема данных открывается командой **Сервис—Схема данных** (в Access_2007 кнопка **Схема данных** на закладке **Работа с базами данных**) и графически отображается в отдельном окне, где таблицы представлены списками полей, причем ключевые поля выделены жирным шрифтом, а связи — линиями между полями разных таблиц. Существует четыре типа связей между таблицами: один-к-одному, один-ко-многим, много-к-одному, много-ко-многим. В Access возможно установление связей один-к-одному или один-ко-многим.

Одно-однозначная связь (1:1) устанавливается, когда каждому экземпляру одной таблицы соответствует только один экземпляр другой и наоборот.

Одно-многозначная связь (1:M) устанавливается, когда каждому экземпляру одной таблицы, являющейся главной, может соответствовать несколько экземпляров другой, подчиненной таблицы. Эти связи являются основными, т.к. связи 1:1 используются лишь в случаях, когда приходится разделять большое количество полей, определяемых одним и тем же ключом, по разным таблицам.

Access автоматически определяет тип связи. Если поле связи является уникальным ключом как в главной таблице так и в подчиненной, устанавливается связь **1:1**. Если поле связи является уникальным ключом в главной, а в подчиненной является не ключевым или входит в составной ключ, устанавливается связь **1:M** (рис. 1.1).

Установление связи между таблицами возможно при следующих условиях:

- связываемые поля имеют одинаковый тип данных, причем имена полей могут отличаться;
- обе таблицы сохраняются в одной базе данных Access;
- главная таблица связывается с подчиненной по уникальному ключу главной таблицы.

При выборе в качестве поля связи в главной таблице неключевого поля тип отношения не может быть определен, и между таблицами устанавливается **связь-объединение**. В этом случае производится объединение каждой записи из одной таблицы с каждой записью из другой при условии равенства значений в поле связи.

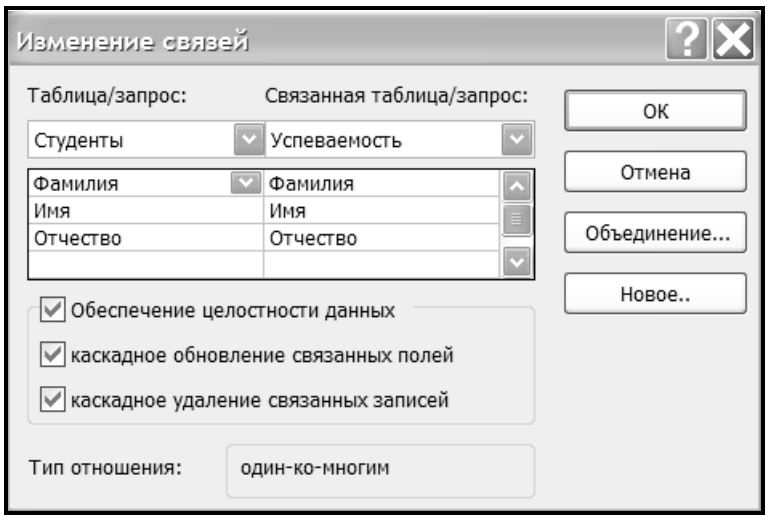


Рис. 1.1. Диалоговое окно создания и изменения связей в схеме данных

Обеспечение целостности данных означает выполнение следующих условий корректировки базы данных:

- в подчиненную таблицу не может быть добавлена запись с несуществующим в главной таблице значением ключа связи;
- в главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчиненной таблице;
- изменение значений ключа связи главной таблицы должно приводить к изменению соответствующих значений в записях подчиненной

При попытке пользователя нарушить эти условия выводится соответствующее сообщение и операция не выполняется (рис. 1.2).

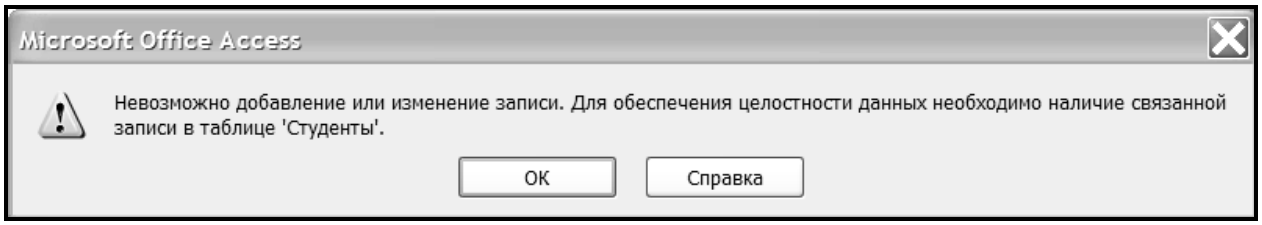


Рис. 1.2. Сообщение о невозможности добавления или изменения записи таблицы

Параметры целостности задаются установкой флажков *Обеспечение целостности данных, каскадное обновление связанных полей* и *каскадное удаления связанных записей* в диалоговом окне **Связи** (рис. 1.1). Эти флажки установить нельзя, если ранее введенные в таблицы данные не отвечают требованиям целостности. Например, в подчиненной таблице имеются записи со значениями полей связи, которые отсутствуют в ключевых полях главной таблицы (рис. 1.3).

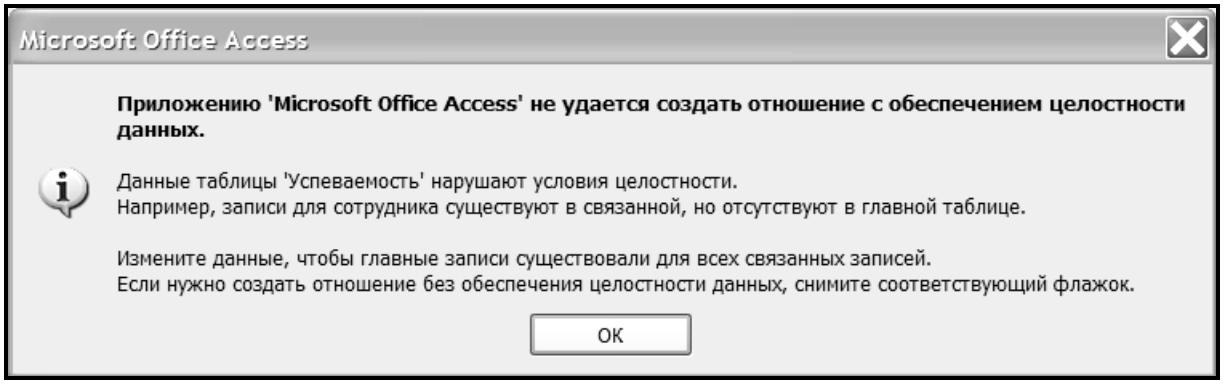


Рис. 1.3. Сообщение о невозможности создать связь с параметрами целостности

В режиме каскадного обновления при изменении значения в поле связи главной таблицы Access автоматически изменит значения в соответствующем поле в записях подчиненных таблиц.

В режиме каскадного удаления при удалении записи из главной таблицы Access выполняет каскадное удаление связанных записей во всех подчиненных таблицах.

Практические задания

Задание 1.1. В режиме **Конструктора** добавить в таблицу **Студенты** следующие поля:

Имя поля	Подпись	Тип данных	Размер	Подстановка (список значений)	Значения по умолчанию
Форма	Форма обучения	текстовый	4	**б** или **вн/б**	**б**
Плата	Плата за обучение	денежный			**0**
Фото	Фотография	текстовый	150		

Задание 1.2. Добавить поля **Форма** и **Плата** в форму **Студенты**. Дополнить записи информацией о форме обучения и оплате за обучение, используя форму.

Фамилия	Имя	Отчество	Форма обучения	Плата за обучение
Андреев	Андрей	Андреевич	б	0
Иванов	Иван	Иванович	вн/б	1800000
Исаченко	Елена	Сергеевна	б	0
Комарова	Ольга	Сергеевна	вн/б	2000000
Краснова	Ирина	Петровна	вн/б	2500000
Крюк	Инна	Федоровна	вн/б	2500000
Миронов	Игорь	Семенович	б	0
Мухина	Любовь	Ивановна	вн/б	2100000
Петров	Петр	Петрович	вн/б	2500000
Степанов	Степан	Степанович	вн/б	2100000

Задание 1.3. Импортировать в свою базу данных с помощью команды **Файл — Внешние данные — Импорт** (в Access_2007 кнопка  на закладке **Внешние данные**) таблицы **Новые студенты** и **Успеваемость** из базы данных **Дополнение**, расположенной *Мои документы → Преподаватели → ТОХОД*. Просмотреть данные этих таблиц.

Задание 1.4. Создать связь **Студенты** $\overset{1:M}{\rightarrow}$ **Успеваемость** с параметрами целостности данных. Обратить внимание на появившиеся сообщения и сделать вывод о причинах их появления.

Примечание. Создайте связь между таблицами *Студенты* и *Успеваемость* без параметров целостности и внимательно просмотрите содержимое этих таблиц.

Задание 1.5. Создать в режиме **Конструктора** запрос на добавление записей (рис. 1.4, 2.4) из импортированной таблицы **Новые студенты** в таблицу **Студенты** и выполнить его.

Задание 1.6. Задать параметры целостности для связи **Студенты** $\overset{1:M}{\rightarrow}$ **Успеваемость**

Примечание. Если параметры целостности все еще не устанавливаются, внимательно просмотрите содержимое таблиц *Студенты* и *Успеваемость*. Вероятнее всего были сделаны опечатки при вводе фамилии, имени или отчества в таблицу *Студенты*, найдите и исправьте их.

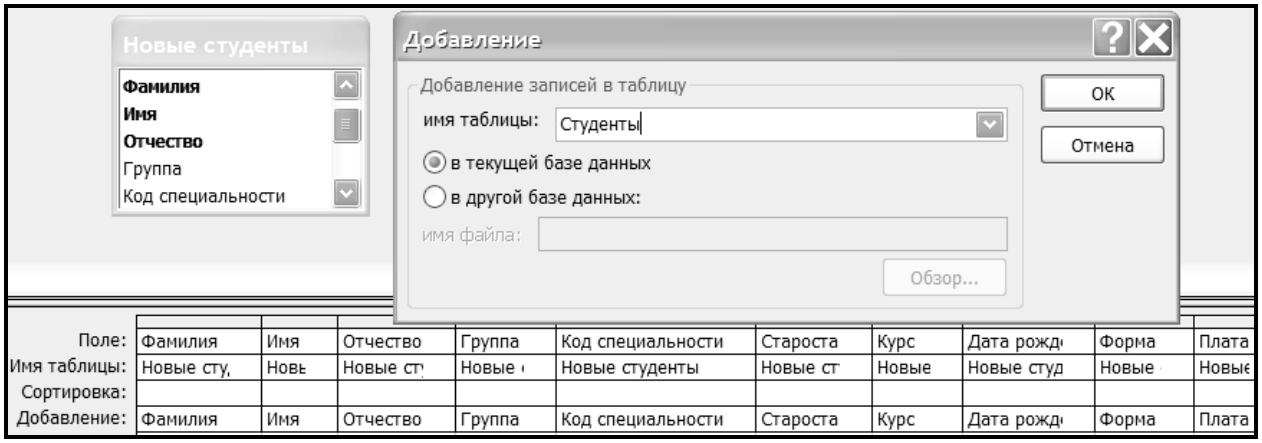


Рис. 1.4. Создание запроса на добавление

Задание 1.7. Проверить поддержку параметров целостности данных следующим образом:

a) в таблицу **Успеваемость** добавить запись.

Фролова	Валентина	Ивановна	История	8

Перейти к другой записи и ознакомиться с появившимся сообщением. Сделать выводы и закрыть таблицу;

b) в таблицу **Студенты** добавить запись и закрыть таблицу.

Фролова	Валентина	Ивановна	108415	2	□	I	04.11.1988	б	0,00 р.

Добавить запись, данную в п. **a)** в таблицу **Успеваемость** и закрыть таблицу.

c) в таблице **Студенты** изменить фамилию *Фролова* на фамилию *Антонова* и закрыть таблицу. Открыть таблицу **Успеваемость** и проследить изменение фамилии. Закрыть таблицу;

d) удалить из таблицы **Студенты** запись об *Антоновой* и закрыть таблицу. Проследить изменения в таблице **Успеваемость**.

Контрольные вопросы

1. Как реализуется поддержка целостности в базе данных?
2. Назовите типы поддержки целостности данных.
3. Что представляет собой схема данных в базе данных?
4. Опишите алгоритм создания связи между таблицами в СУБД Access.
5. Как добавить или удалить таблицу в схеме данных?
6. Какие бывают виды связей между таблицами базы данных и какие могут быть реализованы в СУБД Access?
7. При каких условиях возможно установление связи между таблицами?
8. Что такое составной ключ и как создать по нему связь?
9. Как изменить или удалить связь между таблицами?
10. Как установить параметры целостности базы данных?
11. Как обеспечивается ссылочная целостность при корректировке таблиц базы данных?
12. В каких случаях невозможно создать связь с параметрами целостности?
13. Что представляет собой режим каскадного обновления связанных полей?
14. Что происходит в режиме каскадного удаления связанных записей?

Тема 2. Конструирование запросов

Теоретические сведения

Запросы создаются для выборки данных из одной или нескольких связанных таблиц по заданным условиям, для проведения вычислений и статистической обработки данных. С помощью запроса можно также обновить, удалить, добавить данные в таблицу, создать новые таблицы.

При разработке запросов в СУБД Access можно использовать: 1) режим мастера (для некоторых типов запросов); 2) режим конструктора, являющийся графическим инструментом языка QBE (Query-by-Example — Язык запросов по образцу); 3) язык SQL (см. тему 5).

Запрос на выборку представляет собой стандартный запрос, который позволяет отобразить записи из одной или нескольких таблиц по указанным полям.

Для создания запроса на выборку в режиме **Конструктора** нужно нажать кнопку **Создать** на закладке **Запросы** и выбрать строку **Конструктор** (рис. 2.1) (в Access_2007 кнопка **Конструктор запросов** на закладке **Создание**).

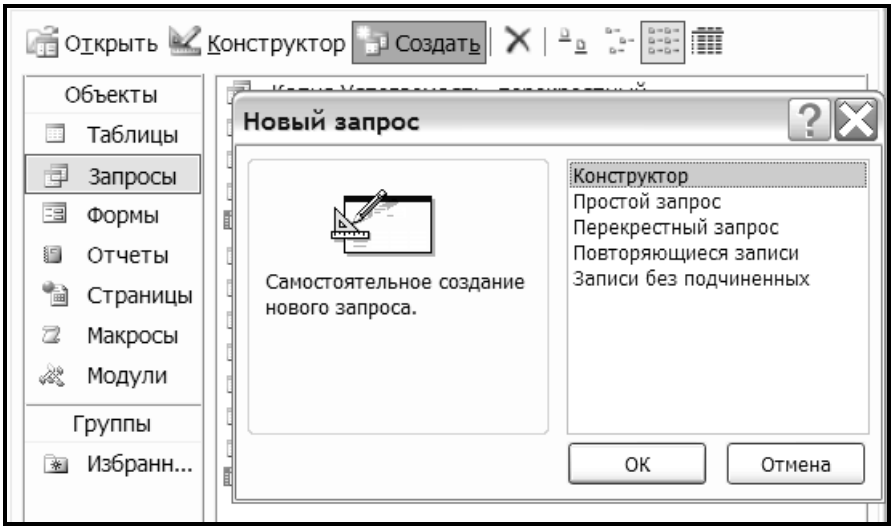


Рис. 2.1. Создание запроса в режиме конструктора

Окно конструктора запросов состоит из схемы данных запроса и бланка запроса (рис. 2.2). Схема данных запроса отображает таблицы или запросы, выбранные в диалоге **Добавление таблицы**, а также связи между ними.

В бланке запроса каждый столбец соответствует одному полю, которое включается в запрос. При заполнении бланка необходимо:

- в строку **Поле** включить имена полей, используемых в запросе. Можно также изменить свойства поля: подпись, формат и число десятичных знаков. Для этого нужно в бланке запроса щелкнуть правой клавишей мыши по имени поля и выбрать команду **Свойства**;
- в строке **Сортировка** выбрать, если нужно, порядок сортировки записей результата;
- в строке **Вывод на экран** отметить поля, которые должны быть включены в результирующую таблицу;
- в строке **Условие отбора** задать условия, по которым отбираются записи.

Условия отбора записей задаются для одного или нескольких полей в зависимости от задач, которые должен решать данный запрос. Условием отбора является выражение, состоящее из **операторов сравнения** (= , < , > , <> , <= , >=) и **операндов**, используемых для сравнения: чисел, текста (заключается в кавычки), дат (заключается в #), имен полей (заключается в [ ]), встроенных функций (подробнее о создании выражений см. тему 3).

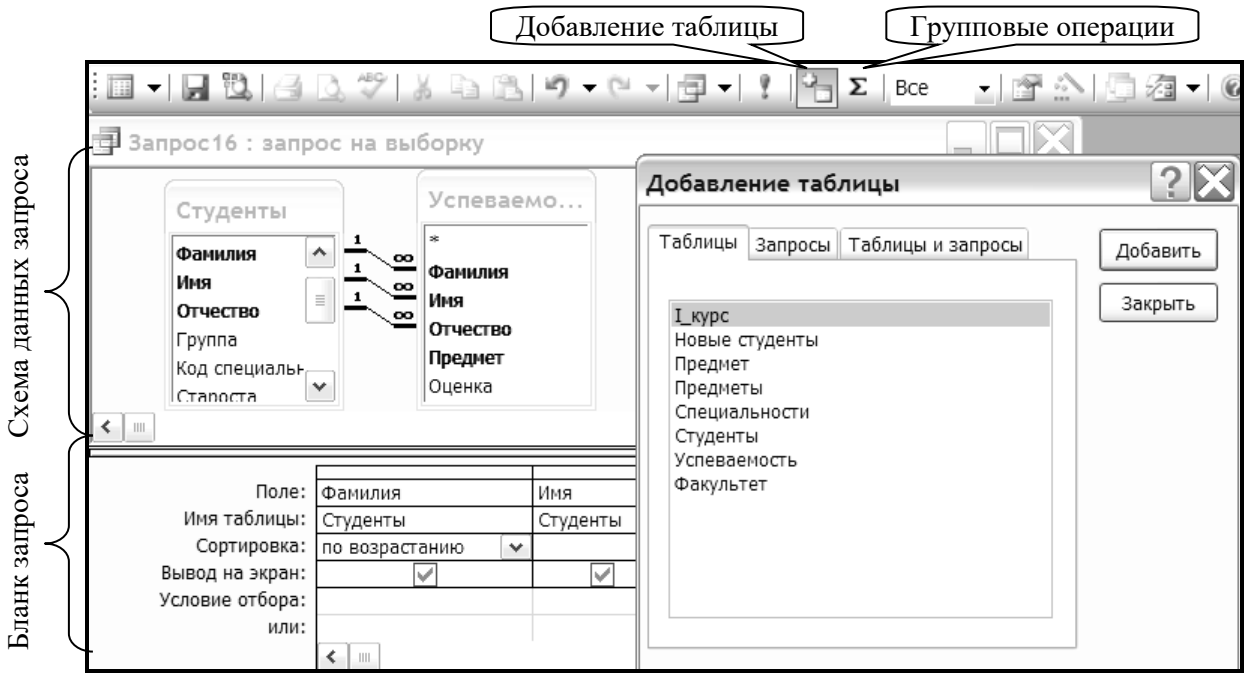


Рис. 2.2. Окно конструктора запросов

В условия отбора могут также включаться:

- логические операторы: **And** (и), **Or** (или), **Not** (не). Эти операторы позволяют объединять несколько условий по следующим правилам: к условиям, заданным для нескольких полей в одной строке, применяется оператор And; к условиям, заданным для одного поля в строках “Условие отбора” и “или” — оператор Or;
- оператор **Between** — задает интервал значений. Синтаксис: *Between ... And ...* . Например, *Between* **30** *And* **85** — это интервал от 30 до 85. Это же условие можно задать с помощью операторов сравнения и логических: >= 30 *And* <= 85.
- оператор **In** — позволяет отобрать записи, в которых значение данного поля равно любому значению из списка, заданного в скобках. Синтаксис: *In(... ; ... ; ...)*. Например, условие *In*(**2**; **6**; **9**) выбирает записи со значениями поля 2 или 6 или 9. Это же условие можно задать с помощью логических операторов: **2** *Or* **6** *Or* **9**.
- оператор **Like** используется для поиска записей, в которых значения текстового поля соответствуют указанному шаблону. Для шаблонов используются следующие символы:

  ? или **_** заменяет любой текстовый символ;

  # — любую одиночную цифру (0 – 9);

  ***** или **%** — любое количество символов;

  [а-л] — символы в заданном интервале;

  [!м-ю] — символы вне заданного интервала;

Например, условие *Like “М*”* выбирает записи со значениями поля, которые начинаются на букву **М**, условие *Like “225-34-??”* выбирает записи со значениями поля, которые начинаются на **225-34-**, а два последних символа — любые.

Запрос на выборку с параметром — это запрос, при выполнении которого появляется приглашение для ввода данных.

Для выбора записей с конкретным значением какого-либо поля можно ввести это значение в строку “Условие отбора”. Однако, это не всегда удобно, так как это значение может меняться в зависимости от ситуации. Тогда вместо конкретного значения в “Условие отбора” вводят параметр. Имя параметра вводится в квадратных скобках. Оно должно отражать сущность хранимых в поле данных и отличаться от имени поля.

При выполнении такого запроса имя параметра отображается в диалоговом окне **Введите значение параметра**. При каждом выполнении запроса нужно вводить конкретное значение параметра, которое будет использовано для формирования условия отбора. В запросе можно задать несколько параметров, тогда порядок их ввода через диалоговые окна определяется порядком расположения полей с параметрами в бланке запроса. Для задания условий с использованием параметров могут также использоваться операторы, рассмотренные выше.

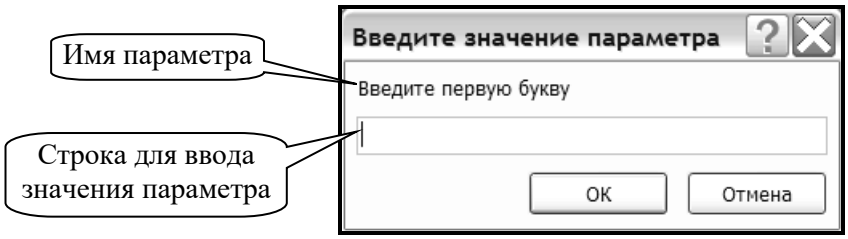


Рис. 2.3. Окно для ввода значения параметра

Запрос на выборку с групповыми операциями

Групповые операции позволяют выделить группы записей с одинаковыми значениями в указанных полях группировки и выполнить статистические вычисления по каждой группе в указанном поле по заданной функции:

- Sum — вычисляет сумму значений поля по группе;
- Min (Max) — находит минимальное (максимальное) значение поля в группе;
- Avg — возвращает среднее от всех значений поля в группе;
- Count — подсчитывает количество непустых значений поля в группе.

Результат запроса содержит по одной записи для каждой группы.

При конструировании запроса в бланк включаются поля, по которым производится группировка, и поля, для которых выполняются групповые функции. Затем выполняется команда **Вид—Групповые операции** или нажимается кнопка 

(рис. 2.2) (в Access_2007 кнопка **Итоги** на закладке **Конструктор**), после чего в бланке запроса появится строка “Групповая операция”. В этой строке для полей, по которым должны выполняться вычисления, вместо **Группировка** следует выбрать нужную функцию.

Перекрестный запрос представляет результаты в виде сводной таблицы с группировкой по строкам и столбцам и вычислениями по заданной функции (Sum, Min, Max, Avg, Count). Таким образом, в перекрестном запросе значения группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй — в верхней строке.

Создание перекрестного запроса в режиме конструктора включает следующие шаги:

1. Создать запрос на выборку в режиме конструктора.
2. Добавить таблицы или запросы, на основе которых будет строиться перекрестный запрос.
3. Добавить поля в бланк запроса и задать (если нужно) условия отбора.
4. Изменить тип запроса на *Перекрестный* (рис. 2.4), после чего бланке запроса появятся две новые строки: “Групповая операция” и “Перекрестная таблица”.
5. Для поля или полей, значения которых группируются по строкам, в строке “Перекрестная таблица” выбрать **Заголовки строк**.
6. Для поля, значения которого должны быть представлены в запросе как заголовки столбцов, в строке “Перекрестная таблица” выбрать **Заголовки столбцов**.

7. Для поля, по которому будут производится вычисления, в строке “Перекрестная таблица” выбрать **Значения**, а в строке “Групповая операция” — нужную функцию.
8. Для полей, по которым задано условие, но не производится группировка или вычисления, в строке “Групповая операция” выбирается **Условие**, а строка “Перекрестная таблица” оставляется пустой. Эти поля не выводятся в результатах запроса.

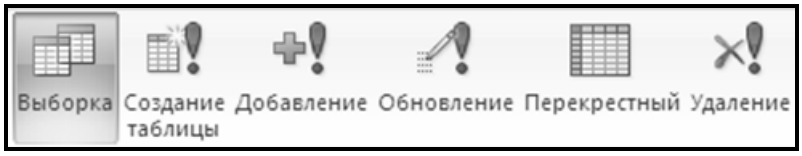


Рис. 2.4. Изменение типа запроса

Запрос **Повторяющиеся записи** предназначен для поиска записей с одинаковыми значениями заданных полей. Строится с помощью мастера (рис. 2.1).

Запрос **Записи без подчиненных** предназначен для поиска записей основной таблицы, у которых нет связанных записей в подчиненной таблице. Строится с помощью мастера (рис. 2.1). При заполнении диалогов мастера, следует выбирать анализируемую (главную) таблицу, подчиненную таблицу и поля связи.

Пример 2.1. Вычислить сумму оплаты по каждому факультету.

Выполнение:

1. Создать новый запрос в режиме конструктора (рис. 2.1).

2. Добавить в запрос таблицы *Факультет*, *Специальности* и *Студенты* и включить групповые операции (рис. 2.2).

3. В бланке запроса выбрать поля *Название факультета* и *Плата*. Следует обратить внимание, что в схему данных запроса была добавлена таблица *Специальности*, хотя поля из нее не используются в запросе. Это сделано для того, чтобы связать таблицы *Факультет* и *Студенты*, в противном случае запрос будет выдавать неверный результат. Аналогичная ситуация встретится в заданиях 2.2.6, 2.2.7, 2.3.1, 2.3.3.

4. В строке “Групповая операция” для поля *Плата* выбрать функцию *Sum* (рис. 2.5).

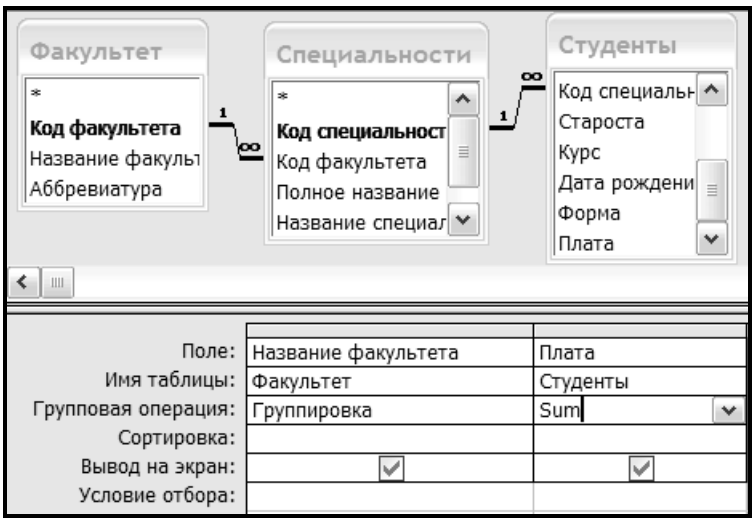


Рис. 2.5. Конструктор запроса для примера 2.1

Пример 2.2. Вычислить средний балл по каждому курсу на каждой специальности.

Выполнение:

1. Создать запрос на выборку в режиме конструктора.

2. Добавить таблицы *Специальности*, *Студенты* и *Успеваемость*.

3. Добавить в бланк запроса поля *Название специальности*, *Курс*, *Оценка*.

4. В данном запросе производится группировка по двум полям *Курс* и *Название специальности*, поэтому для наглядности представления результата его следует сделать перекрестным. Изменить тип запроса на *Перекрестный* (рис. 2.4).

5. Для поля *Название специальности* в строке “Перекрестная таблица” выбрать **Заголовки строк**.

6. Для поля *Курс* в строке “Перекрестная таблица” выбрать **Заголовки столбцов**.

7. Для поля *Оценка* в строке “Перекрестная таблица” выбрать **Значения**, а в строке “Групповая операция” — функцию *Avg* (рис. 2.6).

8. В контекстном меню поля *Оценка* выбрать *Свойства…* и задать: *Формат поля* — Фиксированный; *Число десятичных знаков* — 2 (рис. 2.6).

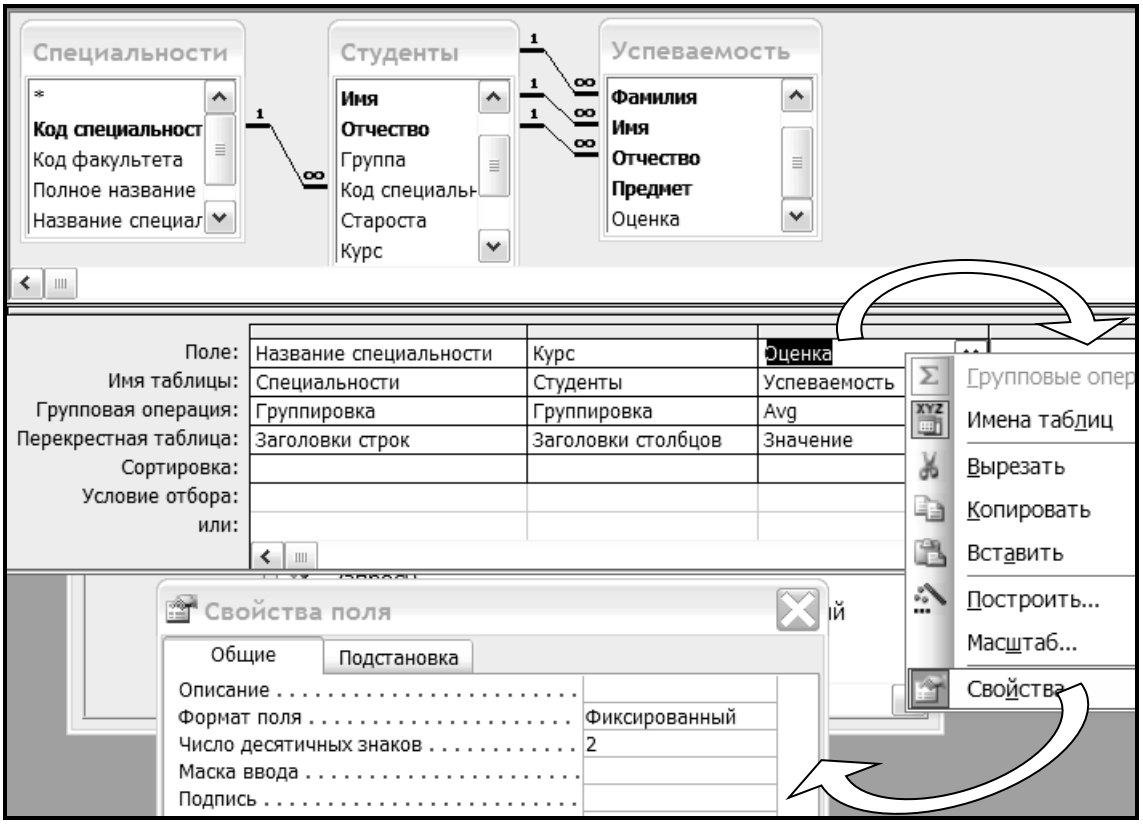


Рис. 2.6. Конструктор запроса для примера 2.2

Практические задания

Рекомендуется присваивать запросам имена, в которых присутствует номер выполняемого задания.

Задание 2.1. Разработать в режиме **Конструктора** запросы на выборку:

2.1.1. Выбрать студентов, обучающихся на бюджете.

2.1.2. Выбрать студентов, номер группы которых начинается на 108.

2.1.3. Вывести информацию о студентах (фамилия, группа, специальность, курс), получивших оценки 6,7,8 по математике и информатике. Разработать **три варианта** данного запроса, используя разные операторы для задания условий отбора. Сравнить результаты по каждому запросу (если условия заданы корректно, результаты должны быть одинаковы).

2.1.4. Вывести информацию об оценках студентов, обучающихся на определенной специальности, при этом название специальности вводить как параметр.

Задание 2.2. Разработать в режиме **Конструктора** запросы на выборку с групповыми операциями:

2.2.1. Подсчитать количество специальностей на факультетах.

2.2.2. Подсчитать количество студентов по каждой форме обучения.

2.2.3. Вычислить общую сумму, полученную за обучение на каждой специальности.

2.2.4. Подсчитать количество студентов, которые учатся на I курсе.

2.2.5. Подсчитать, сколько старост учится по бюджету.

2.2.6. Подсчитать по каждому факультету общее количество студентов.

2.2.7. Подсчитать по каждому факультету количество студентов по внебюджету.

Задание 2.3. Разработать в режиме **Конструктора** перекрестные запросы:

2.3.1. Вычисление суммы оплаты по курсам и факультетам.

2.3.2. Вычисление среднего балла в каждой группе по каждому предмету.

2.3.3. Вычисление количества студентов по специальностям, сдававших каждую дисциплину.

Задание 2.4. Создать запрос с помощью **Мастера запросов** «Записи без подчиненных» для вывода специальностей, на которых нет студентов.

Контрольные вопросы

1. Опишите структуру окна конструктора запросов.
2. Как добавить в запрос новую таблицу, поле, изменить порядок полей, удалить поле?
3. Как изменить подпись и формат поля в запросе?
4. Как задаются условия отбора записей?
5. Назовите логические операторы и их назначение.
6. Назначение операторов Between, In, Like.
7. Как и для чего в запрос вводятся параметры?
8. Каково назначение запросов с групповыми операциями и перекрестных запросов?
9. Какие функции используются запросах с групповыми операциями и перекрестных запросах?
10. Опишите алгоритм создания перекрестного запроса в режиме конструктора.

Тема 3. Создание вычисляемых полей. Запросы-действия

Теоретические сведения

Вычисляемые поля используются для отображения результатов вычислений и могут создаваться в запросах, отчетах и формах. Для создания вычисляемых полей применяют выражения.

Выражением называется сочетание математических и логических операторов, констант, функций, имен и свойств полей, в результате обработки которого получается единственное значение. Выражение может выполнять математические вычисления, обрабатывать текст или осуществлять проверку данных. Для создания выражений используются встроенные функции и операторы:

- функции даты и времени: **Date**(), **Now**() — возвращают текущую дату; **Day**(…), **Month**(…), **Year**(…)— возвращают соответственно день, месяц и год из даты, заданной в качестве аргумента функции;
- статистические функции: **Avg**(…), **Count**(…), **Max**(…), **Min**(…), **Sum**(…);
- функции преобразования типов данных: **Str**() преобразует число в текст, **Val**() преобразует текст в число, **CInt**(…), **CStr**(…), **Cbool** (…), **Cdate**(…) — преобразуют заданное выражение соответственно к целому типу данных, к текстовому, к логическому, к дате;
- функции управления: **IIF**(условие; выражение1; выражение2) — непосредственное условие (Immediate IF). Данная функция проверяет заданное *условие*. Если оно истино, то выполняется *выражение1*, если ложно, то — *выражение2*;
- функции для работы с текстом:
  LCase(…) выводит текст строчными буквами, **UCase**(…) — заглавными буквами;
  Left(…; n), **Right**(…; n) выводят **n** левых (т.е. первых) или **n** правых (т.е. последних) символов, **Mid**(…; k; n) — **n** символов, начиная с **k**-го;
  Ltrim(…), **Rtrim**(…), **Trim**(…) убирают пробелы соответственно слева (в начале текстовой строки), справа (в конце текстовой строки), слева и справа;
  Format(выражение; "…") выводит выражение в формате, заданном в кавычках;
  InStr(k; текст1; текст2) — ищет *текст2* в аргументе *текст1*, начиная с **k**-го символа, и возвращает номер первого вхождения первого символа аргумента *текст2*;
- оператор **&** позволяет объединять тексты и функции в одно строковое выражение;

В качестве аргументов встроенных функций используются имена полей, константы, другие функции. Например:

Month([Дата рождения]) — выводит номер месяца из поля *Дата рождения*;

Avg([Оценка]) — вычисляет среднее значение по полю *Оценка*;

[Фамилия] & " " & Left([Имя];1) & "." & Left([Отчество];1) & "." — выводит фамилию и инициалы, используя значения полей *Фамилия*, *Имя*, *Отчество*;

IIF(Right(Str([Группа]);1)="6";"Выпускник";"Нет") — преобразует поле *Группа* к текстовому типу данных и проверяет последний символ. Если он равен 6 (поступил в 2006 и заканчивает в 2011), то выводится текст *Выпускник*, в противном случае текст *Нет*.

Правила синтаксиса выражений:

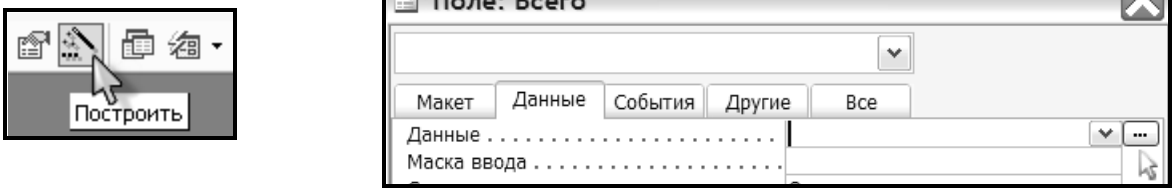
- при создании вычисляемого поля в конструкторе запроса в строке *Поле* вводится имя поля, двоеточие, пробел и затем нужное выражение (*Имя_поля: Выражение)* Например, если в строке *Поле* написано — **Всего: [Оклад] + [Премия]**, то значит будет найдена сумма полей *Оклад* и *Премия* и результат выведен в вычисляемом поле *Всего*;
- тексты заключаются в кавычки ("XXXX");
- даты заключаются в символы # (#ДД.ММ.ГГ#);

- аргументы встроенных функций заключаются в круглые скобки ( ) и разделяются в режиме конструктора точкой с запятой (в режиме SQL и VBA — запятой);
- десятичным разделителем в режиме конструктора является запятая (в режиме SQL и VBA — точка);
- если в именах таблиц, форм, полей есть пробелы или знаки препинания, то они заключаются в квадратные скобки [ ];
- имена объектов разделяются восклицательным знаком или точкой. *Замечание*. Если в запросе или отчете используется несколько таблиц, то в выражении перед именем поля обязательно должно быть имя таблицы, если такое имя встречается в другой таблице. Например, **[Специальности]![Код факультета]**;
- если символ звездочка * используется в выражении, то он обозначает запись. Например: **=Count(*)** — подсчитывает количество записей.

При написании выражений удобно использовать построитель выражений, который автоматически поддерживает большинство рассмотренных правил.

При создании вычисляемых полей в конструкторе запросов для открытия построителя нажимается кнопка на панели инструментов (рис. 3.1, а).

При создании вычисляемых полей в конструкторе отчетов, чтобы зайти в построитель выражений, нужно создать новое поле, открыть его свойства и на закладке **Данные** щелкнуть по многоточию справа от строки **Данные** (рис. 3.1, б). При написании выражения непосредственно в поле (без использования построителя выражений) перед выражением вводится знак =.



а) *б)*

Рис. 3.1. Открытие построителя выражений

С помощью запросов можно не только делать различные выборки, но и обновлять значения полей, удалять или добавлять в таблицу новые записи, создавать новые таблицы, используя данные существующих. Такие запросы называются **активными** или **запросами-действиями**.

Запрос на обновление используется для обновления значений указанных полей таблицы новыми значениями. Запрос строится как запрос на выборку, в который включается обновляемая таблица. Если данные для обновления содержатся в другой таблице, то она также должна быть добавлена, и между этими таблицами нужно провести связь по соответствующим полям. Затем тип запроса меняется на *Обновление* (рис. 2.4) после чего в бланке запроса появится строка “Обновление”. В бланк запроса включается обновляемое поле, для которого в строке “Обновление” вводится значение для обновления или выражение, которое его вычисляет. Если нужно обновить только определенные записи, то в бланк добавляют также поля, по которым задаются условия, и вводят необходимые условия.

Запрос на добавление предназначен для вставки записей из одной или нескольких таблиц в одну целевую таблицу (см. задание 1.5).

Запрос на создание таблицы служит для создания новой таблицы на основе записей существующих таблиц, как правило использует группировку или содержит вычисляемые поля. Может применяться в случае, когда нужно обновить поля какой-либо таблицы данными, которые вычислены в таблице, созданной этим запросом. Строится как

запрос на выборку, а затем тип запроса меняется на *Создание таблицы* (рис. 2.4) и в появившемся диалоговом окне задается имя, которое будет присвоено создаваемой таблице, причем имя должно отличаться от имен существующих таблиц. Затем заполняется бланк запроса согласно поставленной задаче. После выполнения запроса в базе данных появляется новая таблица.

Запрос на удаление предназначен для удаления записей, удовлетворяющих заданному условию. Строится как запрос на выборку, содержащий таблицу, из которой требуется удалить записи. Затем тип запроса меняется на *Удаление* (рис. 2.4). В бланке запроса появляется строка “Удаление”. В первом столбце бланка выбирается звездочка * (что означает «все поля»), после чего в строке “Удаление” будет выведен текст **Из**. В следующих столбцах бланка выбираются поля, для которых задаются условия отбора в зависимости от поставленной задачи.

Пример 3.1. Создать запрос на обновление, который позволит изменить номера групп, начинающиеся на 106, на номера, которые будут начинаться на 102.

Выполнение:

1. Создать запрос на выборку в режиме конструктора.

2. Добавить таблицу *Студенты*.

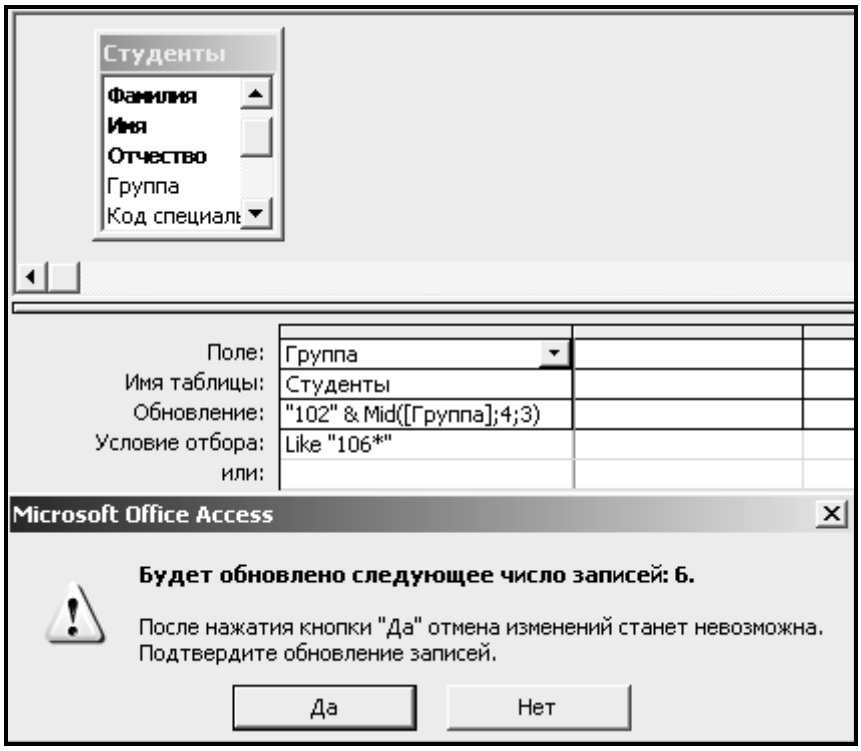
3. Добавить в бланк запроса поле *Группа* и задать для него условие **Like “106*”**

4. Изменить тип запроса на *Обновление* (рис. 2.4).

5. В строке “Обновление” написать выражение: **“102” & Mid([Группа];4;3)**

В данном выражении соединяются два текстовых значения: **“102”** и результат, получаемый с помощью функции **Mid**, которая выводит три символа поля Группа, начиная с четвертого символа.

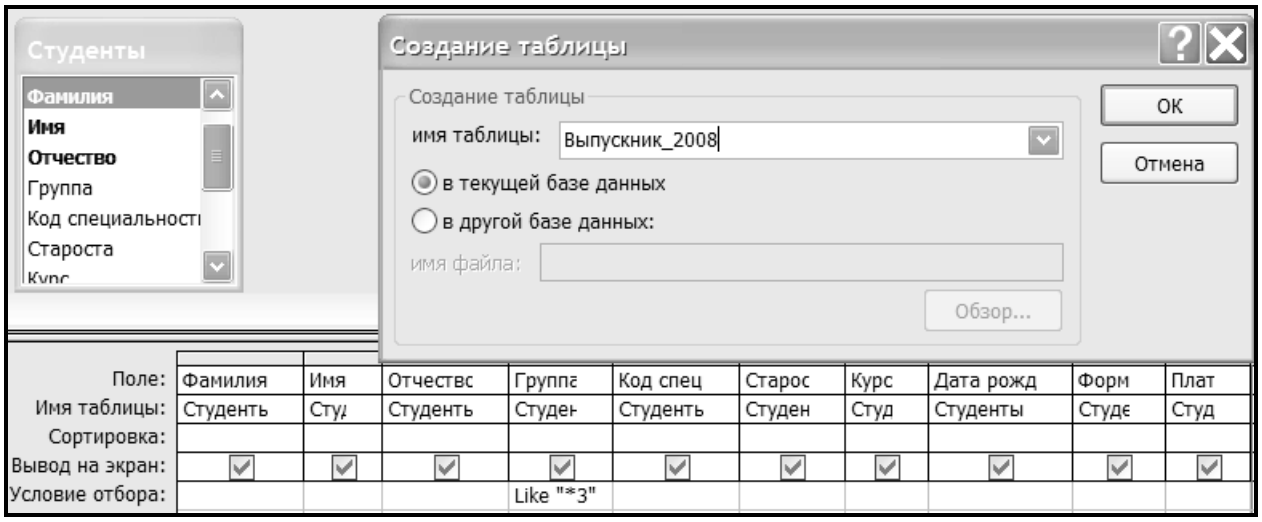
6. При запуске данного запроса появится сообщение, в котором нужно нажать **Да** для подтверждения обновления. Затем открыть таблицу *Студенты* и проверить изменения в поле *Группа*.



Пример 3.2. Сконструировать запрос, который будет создавать таблицу *Выпускник_2008* с данными о студентах групп, номера которых заканчиваются на 3.

Выполнение:

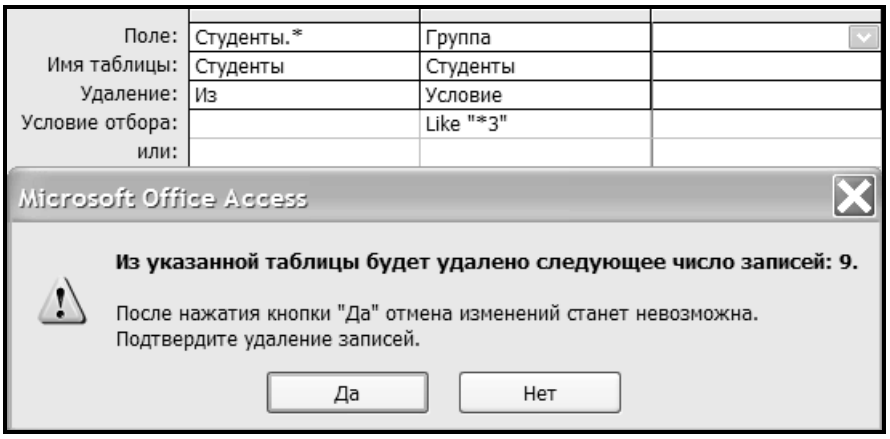
1. Создать запрос на выборку в режиме конструктора.
2. Добавить таблицу *Студенты*.
3. Добавить в бланк запроса все поля. Для поля *Группа* задать условие **Like “*3”**
4. Изменить тип запроса на *Создание таблицы* (рис. 2.4). В открывшемся диалоге задать имя таблицы *Выпускник_2008*.
5. Выполнить запрос, нажимая **Да** во всех появляющихся сообщениях, после чего в базе данных появится таблица *Выпускник_2008*.



Пример 3.3. Создать запрос на удаление из таблицы *Студенты* записей о студентах групп, номера которых заканчиваются на 3.

Выполнение:

1. Создать запрос на выборку в режиме конструктора.
2. Добавить таблицу *Студенты*.
3. Добавить в бланк запроса звездочку * и поле *Группа*, для которого задать условие **Like “*3”**
4. Изменить тип запроса на *Удаление* (рис. 2.4).
5. При запуске данного запроса появится сообщение, в котором нужно нажать **Да** для подтверждения удаления. Затем открыть таблицу *Студенты* и просмотреть ее содержимое.



Практические задания

Задание 3.1. В отчет **Студенты** добавить в область данных поля **Плата** и **Фото**, при этом отчет должен остаться компактным.

Задание 3.2. Создать в области данных отчета **Студенты** вычисляемое поле, которое будет выводить текст **бюджет** или **внебюджет** в зависимости от формы обучения (использовать функцию **IIF**).

Задание 3.3. В разделе отчета **Примечание группы `Курс`** (если такого раздела нет, добавить его) создать: а) вычисляемое поле, которое будет подсчитывать сумму оплаты, полученную по каждому курсу; б) вычисляемое поле, которое содержит поясняющий текст вида: *Сумма оплаты по … курсу составила*. Вместо многоточия должен выводиться соответствующий курс.

Задание 3.4. Создать запрос с полем, вычисляющим долю внебюджетных студентов по каждому факультету (на основе запросов 2-2-6 и 2-2-7). Результат вывести в процентах с одним десятичным знаком.

Задание 3.5. Создать запрос с полем, вычисляющим год рождения студента. Доработать этот запрос так, чтобы он отбирал студентов, родившихся в году, который вводится как параметр.

Задание 3.6. Создать запрос с полем, вычисляющим месяц рождения студента. Доработать этот запрос так, чтобы он определял количество дней рождений по факультетам за каждый месяц (перекрестный запрос).

Задание 3.7. Создать запрос на обновление, повышающий оплату на 10%.

Задание 3.8. Создать запрос на удаление из таблицы **Студенты** записей о студентах I курса специальности ПОИТ. Запрос не выполнять!

Контрольные вопросы

1. Что такое выражение и для чего оно используется в запросах и отчетах?
2. Какие правила синтаксиса применяют при создании вычисляемых полей?
3. Встроенные функции даты и времени.
4. Встроенные функции и операторы для работы с текстом.
5. Встроенные функции преобразования типов данных.
6. Функция IIF.
7. Опишите работу с построителем выражений.
8. Как создать запрос на удаление?
9. Назначение запроса на обновление. Алгоритм его создания.
10. Опишите алгоритм конструирования запроса на создание таблицы.

Тема 4. Конструирование отчетов

Теоретические сведения

Отчеты используются для формирования выходного документа, предназначенного для вывода на печать.

Отчет состоит из нескольких разделов, причем некоторые, кроме области данных, могут отсутствовать в зависимости от назначения отчета:

- Заголовок/примечание отчета — выводится только в начале/конце отчета.
- Верхний/нижний колонтитул — выводится в верхней/нижней части каждой страницы.
- Заголовок/примечание группы — выводится в начале/конце группы при группировке записей по какому-либо полю.
- Область данных — содержит записи таблицы или запроса, относящиеся к данной группе.

Разработка и форматирование отчета аналогично формам. Основные отличия заключаются в следующем:

- в отчетах не используются кнопки и поля со списками;
- создание и изменение группировки и сортировки, а также добавление или отключение разделов *Заголовок группы* и *Примечание группы* осуществляется командой **Сортировка и группировка**, которая доступна в меню **Вид** или в контекстном меню любой области отчета;
- можно расположить каждую группу записей на отдельной странице. Для этого контекстном меню раздела *Примечание группы* выбирается команда **Свойства** и на закладке **Макет** для свойства *Конец страницы* выбирается *После раздела*;
- для добавления вычислений в отчетах создаются вычисляемые поля. Для расчетов по каждой записи вычисляемое поле располагают в области данных. Поля с итоговыми вычислениями размещаются в разделах *Примечание группы* и *Примечание отчета*.

Разработка отчета в режиме конструктора состоит из следующих этапов:

1. Создать запрос на выборку, включающий необходимые поля из таблиц, на основе которых должен базироваться отчет. Если отчет строится на одной таблице, то запрос создавать не нужно.
2. На закладке **Создание** выбрать кнопку **Конструктор отчетов**. В качестве источника данных выбрать созданный запрос или таблицу (рис. 4.1).
3. Нажать кнопку **Добавить поля** на закладке **Конструктор** и включить строку **Показать только поля в текущем источнике записей**, расположенную в нижней части списка полей.
4. В режиме конструктора отчетов задать необходимые уровни группировки. Для создания группировки нужно нажать кнопку **Группировка**. После этого в нижней части экрана появится область для создания группировки (рис. 4.2, а), в которой нужно нажать кнопку **Добавить группировку** и выбрать поле группировки (рис. 4.2, б). В созданной группировке нажать кнопку Больше ► и выбрать строку с разделом примечания (рис. 4.2, в). Аналогично добавить все необходимые группировки. Порядок выбора полей влияет на порядок группировки записей.
5. Перетащить поля из **Списка полей** в нужные разделы отчета. При этом поля, по которым была задана группировка, размещаются по заголовкам групп, а остальные поля — в области данных.
6. Создать вычисляемые поля (см. тему 3).
7. Отформатировать отчет так, чтобы информация размещалась компактно и хорошо читалась, не было пустых страниц. Например, можно добавить линии, обрамление для отдельных полей, дату и номер страницы в колонтитулы и т.д.

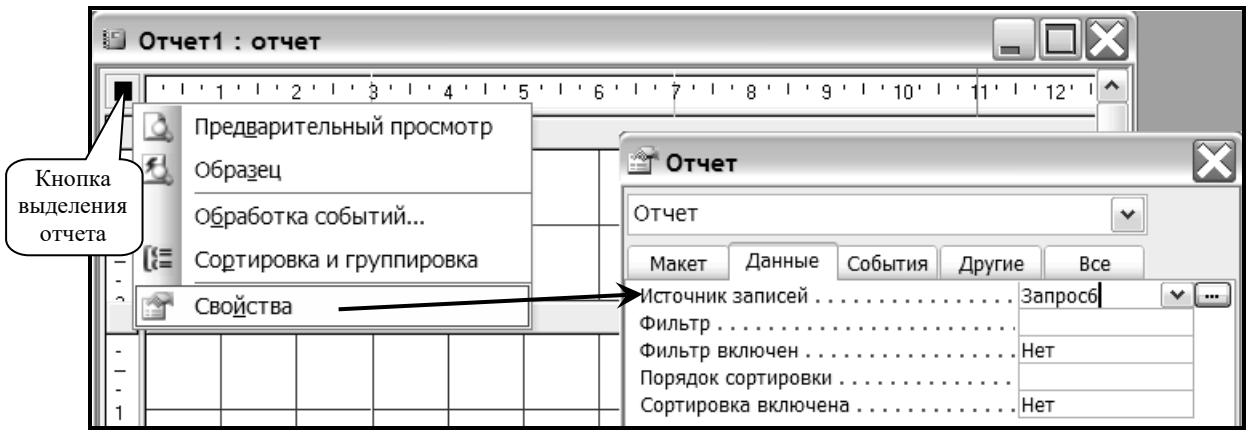
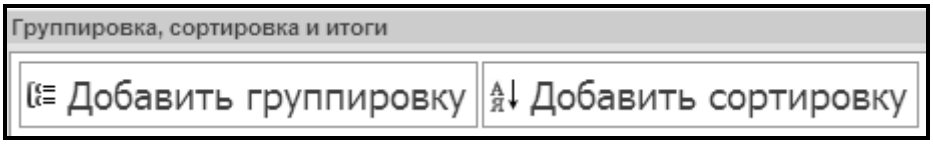
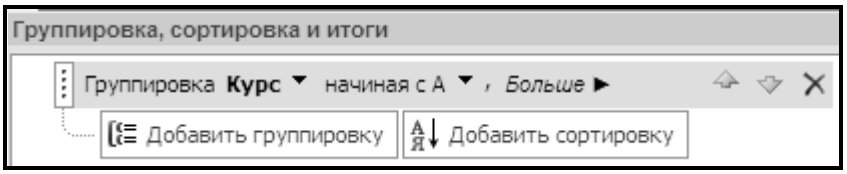


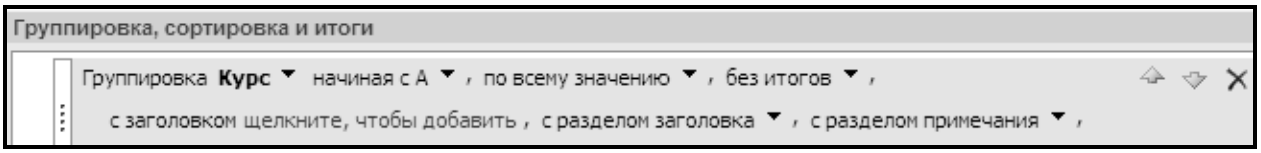
Рис. 4.1. Подключение источника записей



а)



б)



в)

Рис. 4.2. Создание группировки в Access 2007

Практические задания

Задание 4.1. Разработать в режиме конструктора отчет по внебюджетным студентам с группировкой студентов по факультетам и специальностям и подсчетом суммы оплаты по факультетам и количества студентов по специальностям. Разместить каждый факультет на отдельной странице. *Заголовок отчета* оформить как титульный лист на отдельной странице. В *Примечании отчета* вывести общие сумму оплаты и количество студентов.

Рекомендации по выполнению:

1) Создать запрос на выборку на основе таблиц *Факультет*, *Специальности*, *Студенты*, включающий поля *Название факультета*, *Название специальности*, *Фамилия*, *Имя*, *Отчество*, *Группа*, *Форма*, *Плата*. Для поля *Форма* задать условие “вн/б”.
2) Создать отчет в режиме конструктора, выбрав в качестве источника данных созданный запрос.
3) Задать группировку по полям: 1) *Название факультета*; 2) *Название специальности*.
4) Разместить поля и надписи по разделам отчета как показано на рис. 4.3.

5) Создать поля для вычисления суммы оплаты (в примечании группы *‘Название факультета’* и примечании отчета) и количества студентов (в примечании группы *‘Название специальности’* и примечании отчета).
6) В заголовок отчета добавить надписи, которые, по вашему мнению, могут располагаться на титульном листе отчета по внебюджетным студентам.
7) Для разделов *Заголовок отчета* и *Примечание группы ‘Название факультета’* задать в свойстве *Конец страницы* значение *После раздела*. Отформатировать отчет и просмотреть его в режиме предварительного просмотра.

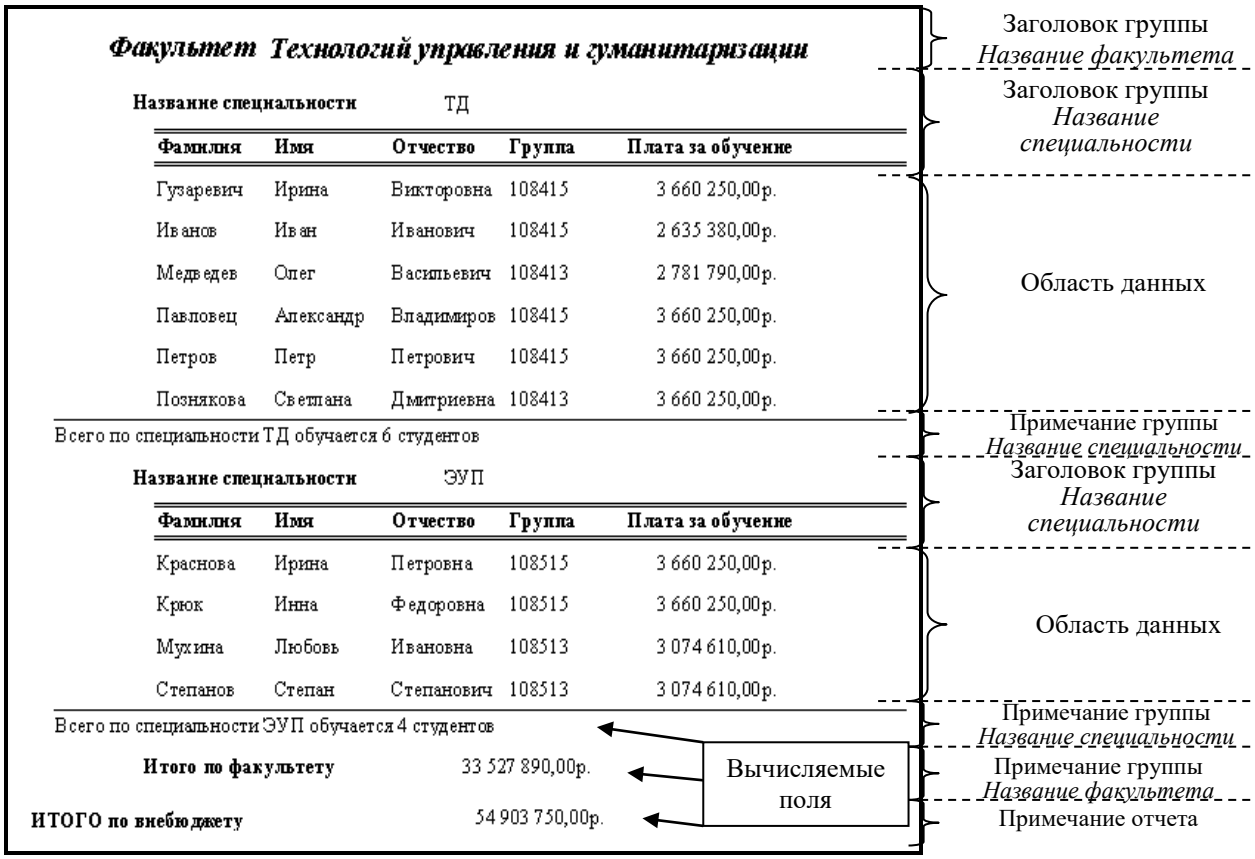


Рис. 4.3. Вид одной из страниц отчета в режиме просмотра

Задание 4.2. Разработать в режиме конструктора отчет по бюджетным студентам с группировкой по факультетам и группам. Фамилии упорядочить по алфавиту. Создать поле, в котором будет вычисляться стипендия студента в зависимости от его среднего балла:

Средний балл	Стипендия
< 5	0
5 – 8	110000
> 8	160000

Вычислить общий итог по стипендиям на каждом факультете и по всему вузу.

Рекомендации по выполнению:

1) Создавайте отчет, руководствуясь этапами, описанными в теоретических сведениях.
2) Для использования в качестве источника данных отчета создайте запрос на выборку на основе таблиц *Факультет*, *Специальности*, *Студенты*, *Успеваемость*, включающий поля *Название факультета*, *Фамилия*, *Имя*, *Отчество*, *Группа*, *Форма*, *Оценка*. Для поля *Форма* задайте условие “б”. Включите группировку и для поля *Оценка* выберите функцию **Avg**.

3) Поле **Стипендия** создается в области данных с использованием функции **IIF**.
4) Поля итогов по стипендиям создаются в примечаниях групп *'Название факультета'* и *'Группа'*. Для них используется функция **Sum**, в которую вставляется формула, разработанная для поля **Стипендия**.

Задание 4.3. Создать в режиме конструктора отчет по успеваемости с полной информацией по оценкам всех студентов и с их группировкой по группам и затем по предметам в этих группах. Для студента должна выводиться фамилия и инициалы. Вычислить средний балл по каждому предмету в группе и средний балл группы по итогам всей сессии. Вычисляемые поля должны сопровождаться поясняющими надписями. Разместить каждую группу на отдельной странице. Сравнить результаты вычислений с результатами задания 2.3.2.

Рекомендации по выполнению:

Создавайте отчет, руководствуясь этапами, описанными в теоретических сведениях, и опытом, полученным при разработке предыдущих отчетов. Структура отчета представлена на рис. 4.4.

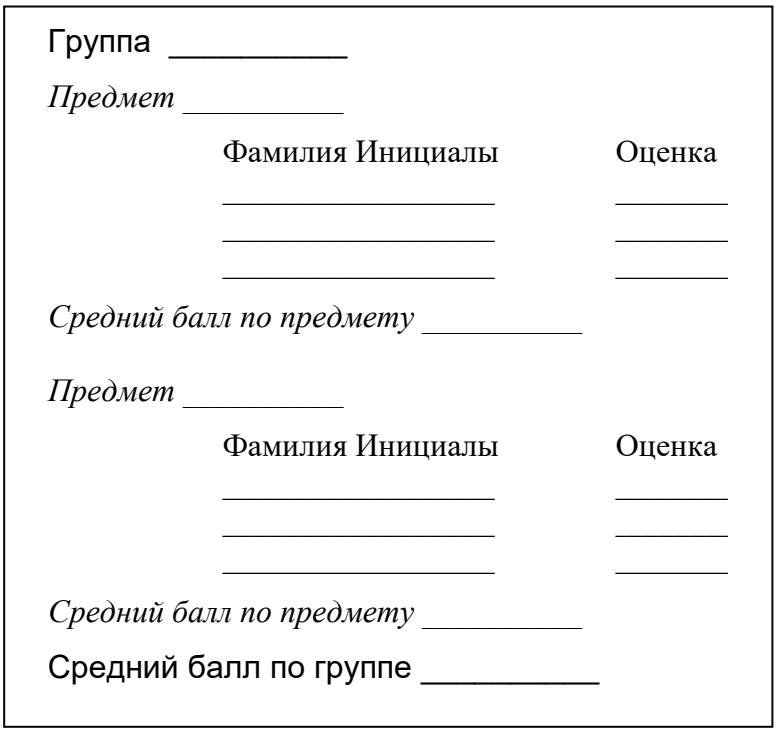


Рис. 4.4. Структурный макет отчета

Контрольные вопросы

1. Для чего служат отчеты и из каких разделов они могут состоять?
2. Опишите порядок создания отчета в режиме конструктора.
3. Какие элементы форматирования используются для оформления отчетов?
4. Как расположить каждую группу записей на отдельной странице?
5. Как в отчете задаются уровни группировки и сортировка?
6. С какой целью и в какие разделы отчета добавляются вычисляемые поля?

Тема 5. Язык SQL

Теоретические сведения

Самый базовый прототип реляционной базы данных был создан в 1974–1975 г., тогда же и был разработан структурированный английский язык запросов (Structured English Query Language, SEQUEL), позднее измененный на SQL (Structured Query Language, произносится «эскюэль»), который в настоящее время получил очень широкое распространение и фактически превратился в стандартный язык реляционных баз данных[1].

Стандарт на язык SQL был выпущен Американским национальным институтом стандартов (ANSI) в 1986 г., а в 1987 г. Международная организация стандартов (ISO) приняла его в качестве международного. Нынешний *стандарт* SQL известен под названием SQL/92 или SQL2, продолжается разработка нового стандарта SQL3. Под реализацией языка SQL понимается программный продукт SQL соответствующего производителя, например диалект SQL, используемый компанией Microsoft для своих продуктов SQL Server, называется Transact-SQL, для MS Access – Jet SQL[2].

SQL вобрал в себя достоинства лежащего в его основе математического аппарата реляционной алгебры и реляционного исчисления. Реляционная модель, предложенная Е.Ф. Коддом в 1970 г., позволяет определять: а) структуры данных; б) операции по запоминанию и поиску данных; в) ограничения, связанные с обеспечением целостности данных. Взаимосвязь реляционной модели данных, стандарта языка SQL и различных его реализаций можно условно изобразить в виде пирамиды (рис. 5.1).

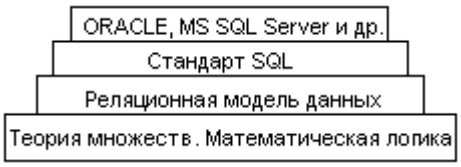


Рис. 5.1. Взаимосвязь реляционной модели и реализаций языка SQL

Основы реляционной алгебры

Термин “relation” (отношение) – это математическое название таблицы. Основное множество реляционной алгебры – это множество отношений (реляций) с определенными над ними операциями, где операндами служит одно или несколько отношений, а результатом – новое отношение.

Отношения должны обладать следующими свойствами:

- отсутствие одинаковых кортежей;
- отсутствие упорядочения кортежей (сверху вниз и слева направо);
- каждый кортеж содержит ровно одно значение для каждого атрибута.

Реляционной алгеброй называется процедурный язык, содержащий множество операторов высокого уровня, применение которых к таблицам (отношениям) приводит к генерации новых таблиц (отношений), содержащих ответы на запросы. Реляционная алгебра определяет над отношениями пять основных операций и три дополнительных, которые можно выражать через основные. Эти операции представлены в таблице 5.1.

Основные идеи реляционной алгебры:

- создание новых таблиц на основе имеющихся;
- упрощённое создание запросов, так как имеется возможность экспериментировать с частичными решениями до тех пор, пока не будет найдено работающее решение.

[1] Майкл Дж.Хернандес, Джон Л.Вьескас. SQL запросы для простых смертных. Практическое руководство по манипулированию данными в sql. Издательство «Лори», 2003, 473с.

[2] Определяет встроенный язык баз данных Microsoft Jet SQL 4.0 и дополнения из стандарта языка баз данных ANSI SQL-92.

Таблица 5.1 – Операции реляционной алгебры

Основные операции над отношениями			**Дополнительные операции над отношениями**		
проекция	–	*PROJECT*	пересечение	–	*INTERSECT*
селекция (выборка)	–	*SELECT*	соединение	–	*JOIN*
объединение	–	*UNION*	деление	–	*DEVIDE BY*
разность	–	*MINUS*			
декартово произведение	–	*TIMES*			

Операции запоминания и поиска делятся на две группы: *операции на множествах* (объединение, пересечение, разность, произведение) и *реляционные операции* (выбирать, спроецировать, соединить, разделить). Они представлены в таблицах 5.2 и 5.3.

Таблица 5.2 – Традиционные теоретико-множественные операции

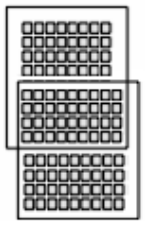
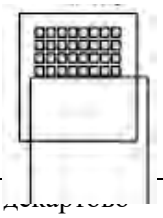
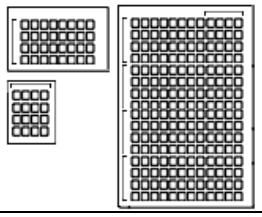
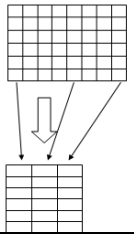
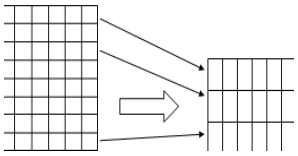
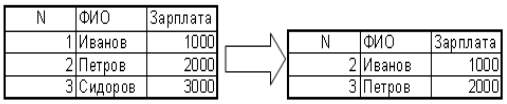
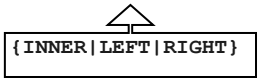
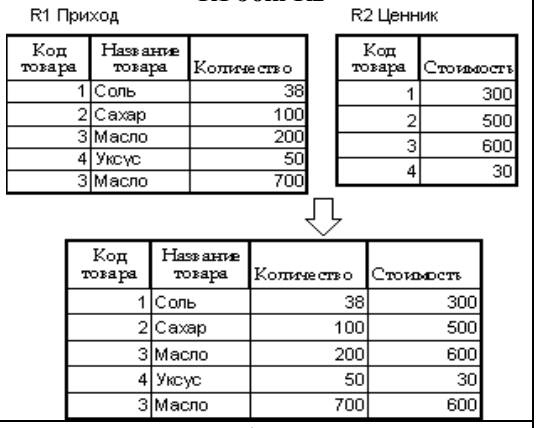
Операция	**Описание**	**Пример**
объединение (∪) **A Union B** 	Объединением двух совместимых по типу отношений **A** и **B** называется отношение с тем же заголовком, что у **A** и **B** и телом, состоящим из кортежей, принадлежащих или **A**, или **B**, или обоим отношениям (дублирование исключается).	**A ∪ B** A N ФИО Зарплата 1 Иванов 1000 2 Петров 2000 3 Сидоров 3000 B N ФИО Зарплата 1 Иванов 1000 2 Пушкин 2500 4 Сидоров 3000 A ∪ B N ФИО Зарплата 1 Иванов 1000 2 Петров 2000 3 Сидоров 3000 2 Пушкин 2500 4 Сидоров 3000
пересечение (∩) **A Intersect B** 	Пересечением двух совместимых отношений **A** и **B** называется отношение с тем же заголовком, что у **A** и **B** и телом, состоящим из кортежей, принадлежащих одновременно **A** и **B**.	**A ∩ B** A N ФИО Зарплата 1 Иванов 1000 2 Петров 2000 3 Сидоров 3000 B N ФИО Зарплата 1 Иванов 1000 2 Пушкин 2500 4 Сидоров 3000 A ∩ B N ФИО Зарплата 1 Иванов 1000
разность (–) **A Minus B** 	Строками таблицы **U = A – B** будут те строки, которые есть в **A**, но которые отсутствуют в **B**.	**A – B** A N ФИО Зарплата 1 Иванов 1000 2 Петров 2000 3 Сидоров 3000 B N ФИО Зарплата 1 Иванов 1000 2 Пушкин 2500 4 Сидоров 3000 A – B N ФИО Зарплата 2 Петров 2000 3 Сидоров 3000
декартово произведение (⊗) **A Times B** 	Имеет результатом отношение, кортежами которых являются конкатенации соответствующих кортежей из отношений-операндов. (Присоединение к каждой строке таблицы **A** каждой строки таблицы **B**).	**A ⊗ B** A N_P ФИО_P 1 Иванов 2 Петров 3 Сидоров B N_D Деталь 1 Болт 2 Гайка 3 Винт A ⊗ B N_P ФИО_P N_D Деталь 1 Иванов 1 Болт 1 Иванов 2 Гайка 1 Иванов 3 Винт 2 Петров 1 Болт 2 Петров 2 Гайка 2 Петров 3 Винт 3 Сидоров 1 Болт 3 Сидоров 2 Гайка 3 Сидоров 3 Винт

Таблица 5.3 – Специальные реляционные операции

Операция	Описание	Пример
проекция ($\pi_{[X,Y,\ldots Z]}(A)$) или **A[X,Y,…,Z],** где X,Y,…Z - заголовки 	“вертикальный срез” – используется для выделения данных, в которых удалены все дубликаты данного среза	**A [Город_Р]** или $\pi_{[Город\_Р]}(A)$ 
селекция ($\delta_{предикат(A)}$) или **A WHERE C** , где C – условие (предикат) 	“горизонтальный срез” – используется для создания таблицы из имеющихся, производя отбор строк из старой таблицы на основании некоторого условия	**A WHERE Зарплата < 3000** или $\delta_{[Зарплата<3000]}(A)$ 
соединение **A Join B** 	Отношение называется соединением, если каждая его запись состоит из записей декартова произведения отношений при выполненном условии отбора (например, равенству полей). Операция позволяет соединять данные из двух таблиц и является обратной к операции проекции (разрезания)	**R1 Join R2** 
деление **A Devide By B**	Делением отношения **A**(x_1, x_n, y_1, y_n) на **B**(y_1, y_n) называется отношение с заголовком (x_1, x_n) и телом, содержащим множество кортежей (x_1, x_n) таких, что для всех кортежей (y_1, y_n) из **B** в отношении **A** найдется кортеж (x_1, x_n, y_1, y_n) Отношение называется делением, если каждая его запись вместе с любой записью из делителя образует запись, имеющуюся в делимом. Смысл операции: в запросах, реализованных с помощью операции деления, в формулировке есть слово «все» (напр., Какие поставщики поставляют <u>все</u> детали?)	**X Devide By Y** 

Изучение SQL дает навыки, необходимые для извлечения информации из любой реляционной базы данных, а разница между диалектами языка изучается быстрее, имея опыт и навыки, полученные при изучении SQL в Microsoft Access 2000 (и выше).

Команды языка SQL можно разделить на категории, представленные в таблице 5.4.

Таблица 5.4 – Основные категории команд языка SQL

DQL – язык *запросов*;	Язык *запросов* (Data Query Language, DQL) наиболее известен пользователям *реляционной базы данных*, несмотря на то, что он включает одну команду SELECT. Эта команда вместе со своими многочисленными опциями и предложениями используется для формирования *запросов* к *реляционной базе данных*.
DML – язык манипулирования данными;	Язык манипулирования данными (Data Manipulation Language, DML) используется для манипулирования информацией внутри объектов *реляционной базы данных* посредством трех основных команд: INSERT, UPDATE, DELETE.
DDL – язык определения данных;	Язык определения данных (Data Definition Language, DDL) позволяет создавать и изменять структуру объектов *базы данных*, например, создавать и удалять *таблицы*. Основными командами языка DDL являются следующие: CREATE TABLE / INDEX, ALTER TABLE / INDEX, DROP TABLE / INDEX.
DCL – язык управления данными;	Язык управления данными (Data Control Language, DCL) позволяет управлять доступом к информации, находящейся внутри *базы данных*. Как правило, он используется для создания объектов, связанных с доступом к данным, а также служит для контроля над распределением привилегий между пользователями. Команды управления данными следующие: GRANT, REVOKE.
TCL – язык управления транзакциями[1].	Язык управления транзакциями (Transaction Control Language, TCL) содержит команды, позволяющие управлять транзакциями *базы данных*: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION. TCL-команды используются для управления изменениями данных, производимыми DML-командами. С их помощью несколько DML-команд могут быть объединены в единое логическое целое, называемое *транзакцией*. При этом все команды на изменение данных в рамках одной транзакции либо завершаются успешно, либо все могут быть отменены в случае возникновения каких-либо проблем с выполнением любой из них. Транзакции есть одно из средств поддержания целостности и непротиворечивости данных и являются одной из важнейших функций современных СУБД.
Команды администрирования данных[2];	С помощью команд администрирования данных осуществляется контроль за выполняемыми действиями и анализируются операции *базы данных*; они также могут оказаться полезными при анализе производительности системы. Не следует путать администрирование данных с администрированием *базы данных*, которое представляет собой общее управление *базой данных* и подразумевает использование команд всех уровней.

[1] Используются в SQL Server
[2] Используются в SQL Server

Для создания управляющих запросов на SQL после создания пустого запроса в режиме Конструктора выполняется команда **Запрос — Запрос SQL — Управление** или в настройках параметров среды Access (рис. 5.2) в режиме *Таблицы и запросы* следует включить режим «**текущая база данных**» в группе *Синтаксис для SQL Server (ANSI 92)*.

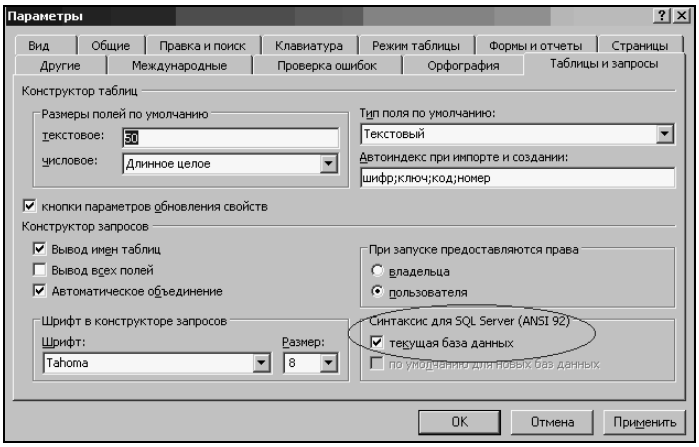


Рис. 5.2. Настройка параметров для управляющих запросов

Каждая команда SQL начинается с глагола — ключевого слова, которое описывает действие, выполняемое командой. После глагола идет одно или несколько предложений. Предложение начинается с ключевого слова и описывает данные, с которыми работает команда, или содержит уточняющую информацию о действии, выполняемом командой. Предложения содержат также имена таблиц и полей БД, константы и выражения.

Рассмотрим основные категории команд SQL.

Язык запросов и команда SELECT

Краткий формат команды SELECT представлен ниже:

```
SELECT [ALL | DISTINCT ] {*|[имя_столбца   [AS новое_имя]]} [,...n]
  FROM  имя_таблицы [[AS] псевдоним] [,...n]
  [WHERE  <условие_поиска>]
  [GROUP BY  имя_столбца [,...n]]
  [HAVING <критерии выбора групп>]
  [ORDER BY имя_столбца [,...n]];
```

Обработка элементов оператора SELECT выполняется в следующей последовательности:

- FROM – определяются имена используемых таблиц;
- WHERE – выполняется *фильтрация строк* объекта в соответствии с заданными условиями;
- GROUP BY – образуются *группы строк*, имеющих одно и то же значение в указанном столбце;
- HAVING – фильтруются группы строк объекта в соответствии с указанным условием;
- SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
- ORDER BY – определяется упорядоченность результатов выполнения операторов.

Отметим, что только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены.

Таблица 5.5 – Основные предикаты оператора SELECT и примеры выборок

Описание выборки	Примеры
Простая выборка `SELECT поле1, …, поле2, …, полеN` `FROM таблица1;` *Замечания*: Колонкам можно присваивать алиасные имена (***псевдонимы***). Каждой таблице можно присваивать алиасные имена (***псевдонимы***). Имена полей, содержащие пробелы или разделители, заключаются в квадратные скобки. Символ-заменитель ***** означает выборку всех полей, что соответствует ключевому слову ***ALL***. Предикат ***DISTINCT*** следует применять в тех случаях, когда требуется отбросить блоки данных, содержащие дублирующие записи в выбранных полях (как в проекции).	Выбрать информацию о студентах: `SELECT Фамилия, Имя, Отчество, Группа` `FROM Студенты;` Выбрать фамилии сотрудников: `SELECT fio AS ФИО` `FROM Sotrudniki;` Использование псевдонимов – выбрать информацию о количестве поставок (поле kol) из таблицы postavki: `SELECT Поставки.kol AS [Количество]` `FROM postavki AS Поставки;` Выбрать всю информацию из таблицы *Контакты*: `SELECT ALL *` `FROM Контакты;` или `SELECT *` `FROM Контакты;` Выбрать, без повторов, список имен студентов: `SELECT DISTINCT Имя` `FROM Студенты;`
Выборка с условием отбора (операция селекция создается по тем же правилам, что и в Конструкторе) `SELECT поле1, …, поле2, …, полеN` `FROM таблица1` `WHERE условие;` *Замечания*: – Даты заключаются в символы “решетки” *(#) и* вводятся в SQL в американском формате: ***мм/дд/гг***. – В условиях могут использоваться: операторы сравнения: **=** – равенство; **<** – меньше; **>** – больше; **<=** – меньше или равно; **>=** – больше или равно; **<>** – не равно. Более сложные предикаты могут быть построены с помощью логических операторов ***AND***, ***OR*** или ***NOT***, а также скобок, используемых для определения порядка вычисления выражения. – В условиях отбора могут применяться операторы принадлежности к диапазону или множеству	Выбрать информацию о студентах группы 108513: `SELECT Фамилия, Имя, Отчество, Группа` `FROM Студенты` `WHERE (Группа="108513");` Выбрать из таблицы *Товары* только те записи, категория товаров в которых равна 2: `SELECT *` `FROM Товары` `WHERE Категория = 2;` Выбрать вклады, совершенные не позднее 18 июня 2009 года: `SELECT [№ вклада]` `FROM [Совершенные вклады]` `WHERE ([Дата вклада]<=#06/18/2009#);` Выбрать фамилии преподавателей, имеющих ученую степень кандидата экономических наук и работающих в должности доцентов: `SELECT ФИО` `FROM Преподаватель` `WHERE (СТ="К.э.н" AND ДОЛЖН="Доцент");` Выбрать информацию о преподавателях, имеющих ученые степени докторов или кандидатов экономических наук: `SELECT ФИО` `FROM Преподаватель` `WHERE (СТ="К.э.н" OR СТ="Д.э.н");`

Продолжение таблицы 5.5

Описание выборки	Примеры
или соответствия шаблону или значению NULL: – ***IN*** (***NOT IN***) – используется для сравнения некоторого значения со списком заданных значений; – ***BETWEEN*** (***NOT BETWEEN***) – используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и максимальным значениями; – ***LIKE*** – можно выполнять сравнение выражения с заданным шаблоном, где допускается использование символов-заменителей: ******* – вместо этого символа может быть подставлено любое количество произвольных символов; ***?*** – заменяет один символ строки; *[ ]* – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях; *[^]* – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях; – ***IS NULL*** (***IS NOT NULL***) – для проверки пустого значения.	Выбрать информацию о преподавателях, работающих в должности старших преподавателей и кандидатах технических наук, работающих в должности доцентов: `SELECT ФИО, СТ, ДОЛЖН` `FROM Преподаватель` `WHERE ((СТ="К.т.н." AND ДОЛЖН="Доцент")` `OR (ДОЛЖН="Ст.преподаватель"));`
	Выбрать товары с ценами в диапазоне от 200 до 2000: `SELECT *` `FROM Товары` `WHERE Цена BETWEEN 200 AND 2000;`
	Выбрать клиентов из Беларуси и Украины: `SELECT Фамилия, Имя, Страна` `FROM Клиенты` `WHERE Страна IN ("Беларусь", "Украина");`
	Выбрать данные для города, начинающегося на требуемую букву. `PARAMETERS [Введите первую букву] TEXT;` `SELECT *` `FROM КодыГородов` `WHERE КодыГородов.Город Like [Введите первую букву] & "*";`
	Выбрать предметы, названия которых начинаются на букву м: `SELECT НП FROM Предмет WHERE (НП Like "м*");`
	Список студентов с именами от A до E: `SELECT Студенты.Фамилия, Студенты.Имя` `FROM Студенты` `WHERE (Студенты.Имя LIKE "[A-E]*");`
	Список преподавателей без ученой степени: `SELECT ФИО, СТ` `FROM Преподаватель WHERE CT Is Null;` Список преподавателей с ученой степенью (любой): `SELECT ФИО, СТ FROM Преподаватель WHERE CT Is Not Null;`
Выборка с параметром `PARAMETERS [Имя параметра1] ТипДанных1, [Имя параметра2] ТипДанных2;`	Выбрать информацию по любой группе: `PARAMETERS =[Введите номер группы] TEXT;` `SELECT Фамилия, Имя, Отчество, Группа` `FROM Студенты` `WHERE (Группа=[Введите номер группы]);`

Окончание таблицы 5.5

Описание выборки	Примеры
SELECT поле1, …, полеN **FROM таблица1** **WHERE (поле2=[Имя параметра1]** **And полеN= [Имя параметра2]);** Описание типа параметра необходимо для текста, а также в перекрестных запросах.	Выбрать студентов, у которых день рождения в месяце, название которого вводится как параметр: **PARAMETERS [Введите месяц] Text;** **SELECT Фамилия, Имя, ДатаРождения** **FROM Студенты** **WHERE (MonthName(Month(ДатаРождения))=** **[Введите месяц]);**
Выборка с вычислениями **SELECT поле1, …, поле2, …, выражение1 AS имя для вычисляемого поля, …** **FROM таблица1** *Замечание*. Для проведения вычислений можно использовать функции разных категорий: **математические**: ***Sqr(), Abs(), Cos(), Sin(),*** и др.; **даты и времени:** ***Date(), Now(), Day(), Month(),Year(), Weekday()*** **статистические**: ***Avg(), Count(), Max(), Min(), Sum();*** **для работы с текстом**: ***LCase(), UCase(), Left(), Right(), Mid(), Ltrim(), Rtrim(), Ttrim(), InStr(), Str(), Val();*** **финансовые функции**: ***PV(), FV(), SLN();*** **функции смешанного типа**: ***IIF(), CCur(), CInt(), CStr(),Format().***	На основании данных поля ФИО, содержащего информацию вида *Иванов Иван Иванович,* сформировать выражение в виде *Иванов И.* **SELECT Таблица1.ФИО, Left([ФИО], InStr([ФИО]," ")+1) & "." AS Выражение** **FROM Таблица1;**
	Вывести списки студентов по году рождения, вычисленному по полю [Дата рождения]: **SELECT Фамилия, [Дата рождения], Year([Дата рождения]) AS Год** **FROM Студенты** **WHERE (Year([Дата рождения])=[Введите год]);**
	Определение среднего значения по оценкам из таблицы *Успеваемость*: **SELECT Avg(оценка) FROM Успеваемость;** Определение самой высокой оценки: **SELECT Max(оценка) AS Балл** **FROM Успеваемость;** Формирование записи об успеваемости по оценкам: **SELECT IIF(оценка<4,"двоечник",** **IIF(оценка>8,"отличник","сдал")) AS Ранг** **FROM Успеваемость;**
Выборка с упорядочением **SELECT поле1, …, поле2, …, полеN** **FROM таблица1** **ORDER BY поле1 [ASC|DESC];** Позволяет управлять порядком вывода результирующей выборки: ***ASC*** – по возрастанию (умолчание), ***DESC*** – по убыванию.	Выбрать информацию из таблицы *Контакты* и расположить ее в порядке убывания даты заказа: **SELECT * FROM Контакты** **ORDER BY ДатаЗаказа DESC;** Вывести список преподавателей (по алфавиту) и занимаемых ими должностей: **SELECT ФИО, ДОЛЖН** **FROM Преподаватель** **ORDER BY Преподаватель.ФИО;**
Выборка по связанным таблицам **SELECT поле1, …, полеN** **FROM таблица1 INNER JOIN таблица2 ON таблица1.полеСвязи = таблица2.полеСвязи;** Связь таблиц может управляться соединением ***INNER JOIN***, ***LEFT JOIN*** или ***RIGHT JOIN*** для возможности получения и контроля данных.	Выдать список студентов, кто сдавал экзамены: **SELECT Студент.НС, Успеваемость.ОЦЕНКА** **FROM Студент INNER JOIN Успеваемость ON (Студент.НС = Успеваемость.НС);** Выдать список студентов с учетом и тех, кто не сдавал экзамены: **SELECT Студент.НС, Успеваемость.ОЦЕНКА** **FROM Студент LEFT JOIN Успеваемость ON (Студент.НС = Успеваемость.НС);**

Таблица 5.6 – Агрегирование в операторе SELECT и примеры выборок

Описание выборки	Примеры
Группировка и итоговые функции `SELECT поле1, …, итоговая функция AS имя для вычисляемого поля` `FROM таблица1` `GROUP BY поля группировки;` Замечание: Все имена полей, приведенные в списке предложения ***SELECT,*** должны присутствовать и во фразе ***GROUP BY*** – за исключением случаев, когда имя столбца используется в итоговой функции. Обратное правило не является справедливым – во фразе ***GROUP BY*** могут быть имена столбцов, отсутствующие в списке предложения ***SELECT***.	Вычислить средний объем покупок, совершенных каждым покупателем: `SELECT Клиент.Фамилия, Avg(Сделка.Кол_во) AS Средний_Объем` `FROM Клиент INNER JOIN Сделка ON Клиент.КодКлиента=Сделка.КодКлиента` `GROUP BY Клиент.Фамилия;`
	Подсчитать количество студентов в каждой группе: `SELECT Группа, Count(Фамилия) AS [Число студентов]` `FROM Студенты` `GROUP BY Группа` `ORDER BY Группа;`
	Определить суммарную стоимость каждого товара за каждый месяц. `SELECT Товар.Название, Month(Сделка.Дата) AS Месяц, Sum(Товар.Цена*Сделка.Кол_во) AS Стоимость` `FROM Товар INNER JOIN Сделка ON Товар.КодТовара=Сделка.КодТовара` `GROUP BY Товар.Название, Month(Сделка.Дата);`
Группировка и условие отбора `SELECT поле1, …, итоговая функция AS имя для вычисляемого поля` `FROM таблица1` `WHERE условие` `GROUP BY поля группировки;` Если совместно с ***GROUP BY*** используется предложение ***WHERE***, то оно обрабатывается первым, а группированию подвергаются только те строки, которые удовлетворяют условию поиска.	Определить суммарную стоимость каждого товара первого сорта за каждый месяц. `SELECT Товар.Название, Month(Сделка.Дата) AS Месяц, Sum(Товар.Цена*Сделка.Кол_во) AS Стоимость` `FROM Товар INNER JOIN Сделка ON Товар.КодТовара=Сделка.КодТовара` `WHERE Товар.Сорт="Первый"` `GROUP BY Товар.Название, Month(Сделка.Дата);`
Фильтрация после группировки `SELECT поле1, …, итоговая функция AS имя для вычисляемого поля` `FROM таблица1` `GROUP BY поля группировки` `HAVING условие ;` При помощи ***HAVING*** отражаются все предварительно сгруппированные посредством ***GROUP BY*** блоки данных, удовлетворяющие заданным в ***HAVING*** условиям.	Вывести список товаров, проданных на сумму более 10000 руб. `SELECT Sum(Товар.Цена*Сделка.Кол_во) AS Стоимость, Товар.Название` `FROM Товар INNER JOIN Сделка ON Товар.КодТовара=Сделка.КодТовара` `GROUP BY Товар.Название` `HAVING Sum(Товар.Цена*Сделка.Кол_во)>10000;`

Окончание таблицы 5.6

<table>
<tr><th>Описание выборки</th><th>Примеры</th></tr>
<tr><td>Перекрестный запрос
TRANSFORM
Статистическая_функция
SELECT имена полей, по которым будет группировка по строкам
FROM таблица
GROUP BY поля группировки по строкам
PIVOT имя поля, из значений которого формируются заголовки столбцов перекрестного запроса;
Перекрестные запросы являются особенностью MS Access, позволяют формировать результат выборки в виде сводной таблицы, где слово PIVOT определяет подписи столбцов, а GROUP BY – определяет подписи строк при выборке и группировании агрегированных данных статистической функцией в слове TRANSFORM.</td><td>Определить средний балл по каждому предмету на каждом курсе.
TRANSFORM Avg(Оценка) AS [Средняя_оценка]
SELECT Предмет
FROM Студенты INNER JOIN Успеваемость ON
(Студенты.Фамилия=Успеваемость.Фамилия)
AND
(Студенты.Имя=Успеваемость.Имя) AND
(Студенты.Отчество=Успеваемость.Отчество)
GROUP BY Предмет
PIVOT Курс;
Результат запроса
<table><tr><th>Предмет</th><th>I</th><th>II</th><th>III</th></tr><tr><td>Информатика</td><td>4,82</td><td>6,73</td><td>7,17</td></tr><tr><td>История</td><td>6,00</td><td>7,29</td><td>5,60</td></tr><tr><td>КИТ</td><td></td><td>6,00</td><td></td></tr><tr><td>Математика</td><td>5,91</td><td>5,91</td><td>5,83</td></tr><tr><td>Менеджмент</td><td>5,33</td><td>7,18</td><td>5,67</td></tr><tr><td>Программирование</td><td>5,29</td><td>7,33</td><td>6,17</td></tr></table></td></tr>
<tr><td>Ограничения на выборку
Предикат TOP n [PERCENT] вводится после оператора SELECT и возвращает определенное число записей, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY.
Предикат TOP не осуществляет выбор между равными значениями.</td><td>Получить список 25 лучших студентов выпуска 1994 года.
SELECT TOP 25 Имя, Фамилия
FROM Студенты
WHERE ГодВыпуска = 1994
ORDER BY СреднийБалл DESC;
Определить самое популярное имя.
SELECT TOP 1 Имя, Count(Имя) AS Кол
FROM Студенты
GROUP BY Имя
HAVING (Count(Имя)>1)
ORDER BY Count(Имя) DESC;</td></tr>
</table>

Таблица 5.7 – Нетривиальные запросы (сложные выборки с подзапросами)

<table>
<tr><th>Описание выборки</th><th>Примеры</th></tr>
<tr><td>Вложенные подзапросы
Подзапрос – это инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором. Текст подзапроса должен быть заключен в скобки.</td><td>Предположим, известна фамилия студента (Воронова Т.В.) и группа (108113), но неизвестно поле НС для него. Чтобы извлечь данные обо всех оценках этого студента из таблицы Успеваемость, можно записать запрос:
SELECT *
FROM Успеваемость
WHERE (Успеваемость.НС =(SELECT НС
FROM Студент WHERE ФИО="Воронова Т.В."
AND Успеваемость.НГ=108113));</td></tr>
</table>

Окончание таблицы 5.7

Описание выборки	Примеры
Подзапрос может извлекать единичное значение (это *скалярный* подзапрос) и множество значений (это *табличный* подзапрос).	Найти самого молодого студента. `SELECT Фамилия, [Дата рождения]` `FROM Студенты` `WHERE ([Дата рождения]=` `(SELECT MAX([Дата рождения])` `FROM Студенты));` Определить названия предметов, по которым сдавались экзамены: `SELECT НазваниеПредмета` `FROM Предмет` `WHERE КодПредмета In` `(SELECT КодПредмета FROM Успеваемость);`
Ключевое слово EXISTS Ключевые слова ***EXISTS*** и ***NOT EXISTS*** предназначены для использования только совместно с *подзапросами*, поскольку по ним проверяется лишь наличие строк в результирующей таблице *подзапроса*. Оператор ***EXISTS*** (существует) генерирует значение истина или ложь и используется в условиях отбора. Для ключевого слова ***EXISTS*** результат равен ***TRUE*** в том и только в том случае, если в возвращаемой *подзапросом* результирующей таблице присутствует хотя бы одна строка. Если результирующая таблица *подзапроса* пуста, результатом обработки операции ***EXISTS*** будет значение ***FALSE***.	Получить данные о студентах, не явившихся на экзамен (оценка=0). `SELECT Distinct НС` `FROM Успеваемость Ф` `WHERE EXISTS` `(SELECT *` `FROM Успеваемость У` `WHERE Оценка=0 AND Ф.НС=У.НС);` Выдать список студентов, имеющих задолженность по экзаменам: `SELECT ФИО, НГ` `FROM Студент` `WHERE NOT EXISTS (SELECT НС,НГ` `FROM Успеваемость` `WHERE Студент.НС=Успеваемость.НС AND` `Студент.НГ=Успеваемость.НГ);` Демонстрация реляционной операции деления – определить, какой поставщик поставляет ВСЕ детали. `SELECT DISTINCT post` `FROM post` `WHERE not exists` `(select *` `from detal` `where not exists` `(select *` `from postavki` `where postavki.n_p=post.n_p` `and postavki.n_d=detal.n_d)` `);`

Язык манипулирования данными

Таблица 5.8 – Запросы-действия

Описание выборки	Примеры
Команда INSERT INTO – запрос добавления `INSERT INTO назначение SELECT имена полей FROM источник` Или `INSERT INTO <имя_таблицы> [(имя_столбца [,...n])] {VALUES (значение[,...n])` Форма оператора ***INSERT*** с параметром ***VALUES*** предназначена для вставки единственной строки в указанную таблицу. Форма оператора ***INSERT*** с параметром ***SELECT*** позволяет скопировать множество строк из одной таблицы в другую.	Добавить в новую таблицу *Младшие_курсы* сведения о студентах III курса: `INSERT INTO Младшие_курсы SELECT * FROM Студенты WHERE (Курс="III");` Добавить в таблицу Предмет новую запись: `INSERT INTO Предмет (Название, Часы, Семестр) VALUES("ТОХОД", 36, 3);` Если значения следуют в порядке, определенном структурой таблицы, то можно применить упрощенную команду: `INSERT INTO Предмет VALUES("КИИТ", 34, 4);`
Команда DELETE – предназначена для удаления группы записей из таблицы. `DELETE [таблица.*] FROM <имя_таблицы> [WHERE <условие_отбора>]`	Удалить из таблицы *Младшие_курсы* сведения о студентах 1986 года рождения и старше. `DELETE * FROM Младшие_курсы WHERE ([Дата рождения]<#12/31/1986#);`
Команда UPDATE – применяется для изменения значений в группе записей или в одной записи указанной таблицы. `UPDATE имя_таблицы SET имя_столбца=<новоеЗначение или выражение>[,...n] [WHERE <условие_отбора>]`	Изменение фамилии, вводимой как параметр, новой фамилией, которая также вводится в диалоге с пользователем. `PARAMETERS [Старая фамилия] Text (50), [Новая фамилия] Text ( 50 ); UPDATE Студенты SET Фамилия = [Новая фамилия] WHERE (Фамилия = [Старая фамилия]);`

Язык определения данных

Таблица является основным объектом для хранения информации в реляционной базе данных. Главное в команде создания таблицы (CREATE TABLE) – определение имени таблицы и описание набора имен полей, которые указываются в соответствующем порядке. Кроме того, этой командой оговариваются типы данных и размеры полей таблицы. *Индексы* представляют собой структуру, позволяющую выполнять ускоренный доступ к строкам таблицы на основе значений одного или более ее столбцов. Индексы обычно создаются с целью удовлетворения определенных критериев поиска после того, как таблица уже находилась некоторое время в работе и увеличилась в размерах. Ключевые поля таблиц всегда индексируются. С течением времени структура базы данных меняется: создаются новые таблицы, а прежние становятся ненужными и удаляются из базы данных с помощью оператора DROP TABLE. Структура существующей таблицы может быть модифицирована с помощью команды ALTER TABLE.

Таблица 5.9 – Команды определения данных

Описание команды	Примеры
Команда CREATE TABLE – создание таблицы `CREATE TABLE имя_таблицы (имя_столбца тип_данных [размер] <ограничение> [NULL | NOT NULL ] [,...n] )` Атрибут ***NULL*** или ***NOT NULL*** используется как специальный маркер, обозначающий тот факт, что поле допускает или нет неопределенное или пропущенное (пустое) значение. Типы данных позволяют задавать текстовые, числовые и денежные значения (character, text, varchar, bit, byte, integer, float, datetime, money, counter и др.). Размер поля указывается только для текстовых и двоичных полей. В синтаксисе команды ***CREATE TABLE*** есть слово ***Constraint*** (ограничение). Разрешено пять типов ограничений: 1. Первичный ключ (***Primary Key***) таблицы. 2. Уникальность (***Unique***) таблицы. 3. Ссылка (***Foreign Key***) таблицы. 4. Значение по умолчанию (***Default***) колонки. 5. Проверка значения колонки (***Check***). Чтобы создать *первичный ключ*, который следит за уникальностью значений по его выражению первичного ключа, включают ограничение первичного ключа: `CREATE TABLE имя_табл (поле1 тип_данных_1 CONSTRAINT имя_ключа PRIMARY KEY, поле2 тип_данных_2, ...);` *Целостность сущностей* определяет, что в базовой таблице ни одно поле первичного ключа не может содержать отсутствующих значений, обозначенных ***NULL***. Ограничение ***Default*** устанавливает значение по умолчанию для колонки. Оно имеет следующий синтаксис: `[CONSTRAINT имя_ограничения] DEFAULT {константа | функция_без_аргументов | NULL}`	Создание таблицы без ключевых полей: `CREATE TABLE Product_Info ( Product_Name char(20) NOT NULL, Description char(30) NULL, Price smallmoney NOT NULL );` Два способа создания таблиц с ключевыми полями: **Первичный ключ** (**PRIMARY KEY**) – это специальный случай комбинирования ограничений **UNIQUE** и **NOT NULL** `CREATE TABLE Product_Info(Product_ID INTEGER NOT NULL UNIQUE, Product_Name char(20) NOT NULL, Description char(30) NULL, Price smallmoney NOT NULL );` `CREATE TABLE Product_Info_PrimaryKey ( Product_ID INTEGER PRIMARY KEY, Product_Name char(20) NOT NULL, Description char(30) NULL, Price smallmoney NOT NULL );` Использование ограничения Constraint для определения первичного ключа: `CREATE TABLE Предметы (КодПредмета Byte CONSTRAINT PrKey PRIMARY KEY, Предмет Text (70));` Создание таблицы, колонка «дата» которой будет принимать значение текущей даты при добавлении новой записи в таблицу, если не будет указано явное значение. `CREATE TABLE Вклад (паспорт char(20), сумма money, фио varchar(25), дата datetime DEFAULT date());`

Окончание таблицы 5.9

Описание команды	Примеры
Внешние ключи используются для установления логических связей между таблицами и построения *ссылочной целостности*. Связь устанавливается путем присвоения значений внешнего ключа одной таблицы значениям ключа другой с помощью ключевого слова ***FOREIGN KEY*** и ***REFERENCES***, где также должна быть указана таблица, с которой организуется связь. `[CONSTRAINT имя_связи] [FOREIGN KEY (поле1 [, поле2 [..., поле16]])] REFERENCES таблица_связи [(поле_связи1 [, поле_связи2 [..., поле_связи16]])]` Дополнительно необходимо, чтобы в родительской таблице обязательно присутствовал первичный ключ по тем же полям.	Создание таблицы STUD со связью с таблицей SPEC по полю KODSPEC: `CREATE TABLE STUD( FAM CHARACTER(20) NOT NULL, IM CHARACTER(20) NOT NULL, OT CHARACTER(20)NOT NULL, CONSTRAINT FIO PRIMARY KEY (FAM,IM,OT), KURS CHAR(2), KODSPEC INTEGER NOT NULL, CONSTRAINT KODSP FOREIGN KEY (KODSPEC) REFERENCES SPEC(KODSPEC) ON UPDATE CASCADE ON DELETE CASCADE)` Заданное ограничение имеет имя `KODSP` и в случае возникновения нарушения ссылочной целостности, если, например, при вводе устанавливать ссылки на отсутствующие строки в родительской таблице, то это имя будет присутствовать в сообщении об ошибке.
Команда ALTER TABLE – модификация таблицы**:** `ALTER TABLE имя_таблицы {[ADD [COLUMN] имя_столбца тип_данных [NULL | NOT NULL ]] | [DROP [COLUMN] имя_столбца]}` Команда позволяет добавлять или удалять столбцы и ограничения.	Удалить ограничение первичного ключа в таблице STUD: `ALTER TABLE STUD DROP CONSTRAINT FIO;` Добавить ключевое поле (счетчик) в таблицу STUD: `ALTER TABLE STUD ADD КодСт COUNTER(1,1) PRIMARY KEY NOT NULL;`
Команда DROP TABLE – удаление таблицы: `DROP TABLE имя_таблицы [RESTRICT | CASCADE]`	Удаление таблицы Группа: `DROP TABLE Группа;`
Команды CREATE INDEX и DROP INDEX – создают и удаляют индексы: `CREATE [UNIQUE] INDEX имя_индекса ON имя_таблицы (имя_столбца[ASC|DESC][,...n])` `DROP INDEX имя_индекса ON имя_таблицы;` Указанные в операторе ***Create Index*** *столбцы* составляют *ключ индекса*. Индексы - это наборы уникальных значений для некоторой *таблицы* с соответствующими ссылками на данные.	Создание уникального индекса по полю ФИО в таблице Студент: `CREATE UNIQUE INDEX index_fio ON Студент (ФИО);` Удаление индекса в таблице Студент: `DROP INDEX index_fio ON Студент;`

Практические задания

Разработать в базе данных *Студенты* **запросы** в режиме **SQL**.

Рекомендуется присваивать запросам имена, в которых присутствует номер выполняемого задания. При проведении вычислений задавайте полям корректные алиасные имена (псевдонимы).

Задание 5.1. Запрос **на выборку** указанных полей по одной таблице:

5.1.1. Выбрать аббревиатуры факультетов.

5.1.2. Выбрать фамилии студентов из таблицы Успеваемость, исключив повторяющиеся записи.

Задание 5.2. Запрос **по связанным таблицам**:

5.2.1. Вывести название специальностей и соответствующие им номера групп, исключив повторы.

5.2.2. Вывести аббревиатуры факультетов и названия специальностей для этих факультетов.

Задание 5.3. Запросы **с условиями отбора** и **параметрами**:

5.3.1. Вывести информацию о студентах (фамилия, группа, факультет, предмет), имеющих оценки 2 и 3.

5.3.2. Отобрать студентов, родившихся до 1988 года.

5.3.3. Выбрать студентов, обучающихся в определенной группе, при этом группу вводить как параметр.

5.3.4. Выбрать студентов, имеющих имя, запрашиваемое как параметр.

Задание 5.4. Запросы с фразой **BETWEEN** или **IN**:

5.4.1. Отобрать студентов-отличников с оценками от 9 до 10.

5.4.2. Выбрать студентов, имеющих положительные оценки (6, 7 или 8).

Задание 5.5. Запросы с фразой **LIKE**:

5.5.1. Отобрать студентов, которые родились в апреле.

5.5.2. Найти студентов, в чьих фамилиях имеется мягкий знак.

Задание 5.6. Запрос **с сортировкой**:

5.6.1. Создать список групп в порядке возрастания номеров.

5.6.2. Создать упорядоченный по убыванию и без повторов список оценок, полученных студентами.

Задание 5.7. Запросы **с вычислениями** (без группировки):

5.7.1. Вычислить общую сумму по оплате за обучение.

5.7.2. Подсчитать, сколько студентов сдавали экзамены и какой получился средний балл по Успеваемости.

Задание 5.8. Запросы с фразой **GROUP BY**:

5.8.1. Определить количество студентов по группам.

5.8.2. Определить количество студентов каждого курса, обучающихся на платной основе.

5.8.3. Вычислить средний балл каждого бюджетного студента, вывести также информацию об его группе и факультете.

5.8.4. Вычислить суммы по оплате вне бюджетников по каждому факультету.

Задание 5.9. Запросы с фразой **HAVING**:

5.9.1. Определить количество платных студентов младших курсов (I и II).

5.9.2. Выбрать бюджетных студентов, чей средний балл выше семи.

Задание 5.10. Запросы **с подзапросами**:

5.10.1. Найти самого взрослого студента.

5.10.2. Вывести информацию о студентах (фамилию, имя, группу), у которых есть тезки (совпадают имена).

Задание 5.11. Провести **модификацию** базы данных – **добавить** поле **Пол** (текстовое, размер 3) в таблицу **Студенты**.

Задание 5.12. **Обновить** поле **Пол** соответствующими значениями (**муж** или **жен** в зависимости от значений двух последних символов поля **Отчество**).

Задание 5.13. **Проиндексировать** таблицу **Студенты** по полю **Пол**.

Задание 5.14. Разработать **перекрестный** запрос, определяющий количество юношей и девушек в каждой группе.

Задание 5.15. Разработать **перекрестный** запрос, определяющий количество оценок 2, 3, … 9, 10 по каждому предмету.

Задание 5.16. Разработать **перекрестный** запрос, определяющий количество оценок 2, 3, … 9, 10 в каждой группе факультета, аббревиатура которого вводится как параметр.

Задание 5.17. Провести **модификацию** базы данных – **добавить** поле **Кол_студ** с типом данных *Byte* в таблицу **Специальности**.

Задание 5.18. Разработать запрос-действие **на создание** таблицы, в которой подсчитывается количество студентов на каждой специальности.

Задание 5.19. Разработать запрос **на обновление** поля **Кол_студ** данными таблицы, созданной по запросу задания 5.18.

Задание 5.20. Разработать запрос **на удаление** таблицы, созданной в задании 5.18.

Контрольные вопросы

1. Каково назначение запроса-выборки?
2. Как задается и используется псевдоним для таблиц и полей?
3. Какими средствами языка SQL выполняются такие операции реляционной алгебры, как проекция и селекция? Приведите примеры.
4. Можно ли вывести из таблицы первые пять записей? Ответ пояснить.
5. Опишите и зарисуйте, какой результат получим по представленному ниже запросу

```
SELECT DISTINCT UCase([Фамилия]) AS ФИО, "отличник" AS Результат
FROM Успеваемость
WHERE Оценка>8;
```

   Изменится ли количество результирующего набора, если в запросе поле [Фамилия] изменить на поле [Имя] и почему?
6. Как задается уникальность в описании столбцов при создании таблиц в базе данных?
7. Пояснить, будет ли корректно выполняться представленная команда вставки в таблицу новой строки:

```
INSERT INTO STUDENT (Id, city, name) VALUES (101, NULL, "Туров", 200)
```

8. В разработанной базе данных имеются недостатки, которые не позволяют вводить в нашу базу данных студентов с одинаковыми данными в фамилии, имени, отчестве. А как можно исправить эту ситуацию?
   Напишите команды ***CREATE TABLE*** для создания корректной структуры таблиц базы данных **Студенты**.

Самостоятельная работа

По предложенным схемам, без использования компьютера разработать *запросы* в режиме ***SQL*****,** представить *SQL-запросы* в ***режиме Конструктора*** и реализовать ***операции реляционной алгебры***.

Вариант 1. База данных «Книги» имеет схему данных, представленную на рис. 5.3.

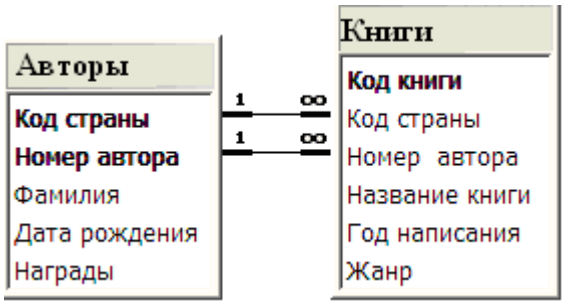


Рис. 5.3. Схема базы данных «Книги»

Содержание таблицы **Авторы**

Код страны	Номер автора	Фамилия	Дата рождения	Награды
1	1	Туров	12.02.1956	☑
1	2	Малышев	22.04.1934	☑
1	3	Белов	10.02.1860	☐
2	4	Браден	19.11.1970	☑
2	5	Сайлер	02.12.1981	☐
1	6	Коваленко	29.05.1983	☐
1	7	Гарин	04.06.1965	☐
2	8	Смит	06.05.1942	☑
1	9	Романов	13.12.1961	☐

Содержание таблицы **Книги**

Код книги	Код страны	Номер автора	Название книги	Год написания	Жанр
1	1	1	Острова	1988	приключения
2	1	2	Затерянные сокровища	1988	приключения
3	2	4	За океаном	2008	приключения
4	2	4	Колье Анны	1998	детектив
5	1	2	Аэропорт	1988	роман
7	1	3	Екатерина	1892	роман
8	1	6	Перекресток	2004	приключения
9	1	7	Замкнутый круг	2004	детектив
10	1	7	К звездам	1998	приключения
11	2	8	Доктор Спенс	1971	детектив
12	2	8	Карен и Джек	1988	роман
13	1	6	Пересекая экватор	2008	приключения
14	1	1	Калейдоскоп	2004	детектив
15	1	2	Торнадо	1963	детектив
16	2	5	Полночь	1971	детектив

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.

Поле:				
Имя таблицы:				
Вывод на экран:				
Сортировка:				
Условие отбора:				
или:				

Запрос 1.1.

SELECT Фамилия, [Название книги], [Год написания], Жанр
FROM Авторы **INNER JOIN** Книги **ON** (Авторы.[Номер автора] = Книги.[Номер автора]) **AND** (Авторы.[Код страны] = Книги.[Код страны])
WHERE ([Год написания] Between 1965 And 2000) **AND** (Жанр = "детектив");

Запрос 1.2.

SELECT Фамилия, [Дата рождения] **INTO** Поздравление
FROM Авторы
WHERE (Month([Дата рождения]) = [Номер месяца]);

Запрос 1.3.

SELECT Фамилия, [Название книги], Жанр
FROM Авторы **INNER JOIN** Книги **ON** (Авторы.[Номер автора] = Книги.[Номер автора]) **AND** (Авторы.[Код страны] = Книги.[Код страны])
WHERE (Фамилия = [Фамилия автора]);

Запрос 1.4.

SELECT Жанр, Count([Код книги]) **AS** [Count-Код книги]
FROM Книги
GROUP BY Жанр;

Запрос 1.5.

TRANSFORM Count(Книги.[Код книги]) **AS** [Count-Код книги]
SELECT [Код страны]
FROM Книги
GROUP BY [Код страны]
PIVOT [Год написания];

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит фамилия и названия книг авторов с 1950 по 1980 год рождения, имеющих награды.

Запрос 2.2. Перекрестный запрос, вычисляющий количество книг каждого жанра по каждому году написания.

Запрос 2.3. Запрос на удаление книг, написанных до 1900 года.

3. Для запросов **1.1**, **1.4**, **1.5**, **2.1** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\pi_{2,3,5}$ (Книги) или Книги [Код страны, Жанр].

4.2. Предприятие состоит из двух филиалов. В таблице КФ1 хранится информация о клиентах первого филиала. В таблице КФ2 хранится информация о клиентах второго филиала. Найти КФ1 – КФ2. Описать смысловое значение полученного отношения.

КФ1

Название	*Город*	*Телефон*
Альвена	Минск	222-77-88
Сотис	Гродно	111-22-33
Гедеон	Брест	444-33-66
Октан	Минск	223-55-44
Белпак	Минск	223-11-22

КФ2

Название	*Город*	*Телефон*
Белпак	Минск	223-11-22
C&D	Витебск	555-88-33
Сотис	Гродно	111-22-33
Винтек	Минск	222-99-11

Вариант 2. База данных «Дома» имеет схему данных, представленную на рис. 5.4.

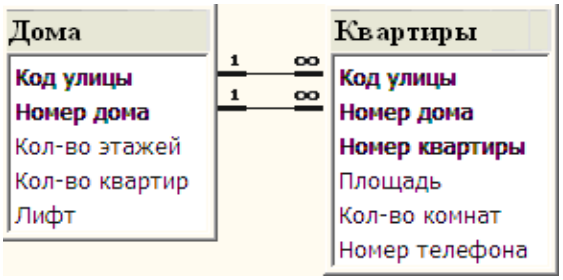


Рис. 5.4. Схема базы данных «Дома»

Содержание таблицы **Дома**

Код улицы	Номер дома	Кол-во этажей	Кол-во квартир	Лифт
1	1	9	1	☑
1	2	12	1	☑
1	3	9	1	☐
1	4	7	1	☑
2	1	12	1	☑
2	2	14	1	☑
2	3	12	1	☑
2	4	7	1	☐
2	5	3	1	☐
2	6	6	1	☑

Содержание таблицы **Квартиры**

Код улицы	Номер дома	Номер квартиры	Площадь	Кол-во комнат	Номер телефона
1	1	1	80	3	229-22-11
1	1	2	70	2	229-22-33
1	1	3	90	3	227-22-29
1	2	1	80	2	227-22-44
1	2	2	40	1	227-22-55
1	2	3	50	1	229-22-66
1	2	4	60	2	227-22-77
2	3	1	60	2	225-33-66
2	3	2	100	3	225-33-77
2	5	1	90	3	225-33-88
2	5	2	80	3	223-42-25
2	5	3	70	2	223-42-29
2	5	4	30	1	223-22-90
2	6	1	70	2	322-92-25
2	6	2	55	2	322-52-29

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.

Запрос 1.1.

SELECT [Код улицы], [Номер дома], [Номер квартиры], [Номер телефона]
FROM Квартиры
WHERE ([Номер телефона] Like "225*") **OR** ([Номер телефона] Like "229*");

Запрос 1.2.

SELECT [Код улицы], [Номер дома], [Номер квартиры], Площадь, [Кол-во комнат]
FROM Квартиры
WHERE (Площадь > 70) **AND** ([Кол-во комнат] = [Число комнат]);

Запрос 1.3.

SELECT [Код улицы], [Номер дома], Sum(Площадь) **AS** [Sum-Площадь]
FROM Квартиры
GROUP BY [Код улицы], [Номер дома];

Запрос 1.4.

SELECT [Код улицы], [Номер дома], Count([Номер квартиры]) **AS** [Count-Номер квартиры] **INTO** [Кол-во квартир]
FROM Квартиры
GROUP BY [Код улицы], [Номер дома];

Запрос 1.5.

TRANSFORM Avg(Квартиры.Площадь) **AS** [Avg-Площадь]
SELECT [Код улицы]
FROM Квартиры
GROUP BY [Код улицы]
PIVOT [Кол-во комнат];

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит коды улиц и номера домов, имеющих более 4 этажей и не имеющих лифта.

Запрос 2.2. Перекрестный запрос, вычисляющий количество домов каждой этажности на каждой улице.

Запрос 2.3. Запрос на обновление поля **Кол-во квартир** с использованием данных, получаемых с помощью запроса **1.4**.

3. Для запросов **1.1**, **1.3**, **1.5**, **2.1** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\delta_{5=3\ \text{And}\ 4<90}$ (Квартиры) или Квартиры(Кол-во комнат = Номер дома и Площадь<90)

4.2. Предприятие состоит из двух филиалов. В таблице КФ1 хранится информация о клиентах первого филиала. В таблице КФ2 хранится информация о клиентах второго филиала. Найти КФ1 ∩ КФ2. Описать смысловое значение полученного отношения.

КФ1

Название	*Город*	*Телефон*
Альвена	Минск	222-77-88
Сотис	Гродно	111-22-33
Гедеон	Брест	444-33-66
Октан	Минск	223-55-44
Белпак	Минск	223-11-22

КФ2

Название	*Город*	*Телефон*
Белпак	Минск	223-11-22
C&D	Витебск	555-88-33
Сотис	Гродно	111-22-33
Винтек	Минск	222-99-11

Вариант 3. База данных «Занятия» имеет схему данных, представленную на рис. 5.5.

Рис. 5.5. Схема базы данных «Занятия»

Содержание таблицы **Препод**

КодПрепода	ФИО
1	Разорёнова Т.Р,
2	Альшевская О.В.
3	Галай Т.А.
4	Ковалькова И.А.
5	Лабкович О.Н.
6	Моисеенко Е.Г.
7	Бровка Г.М.

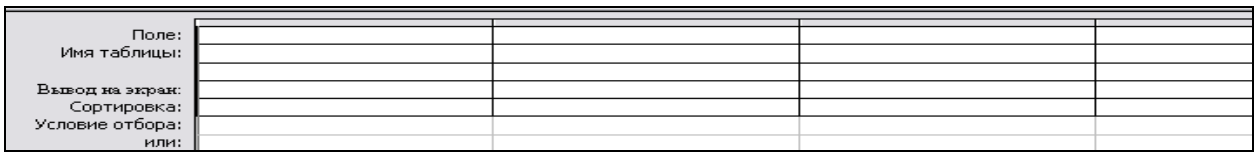
Содержание таблицы **Предмет**

КодПредмета	Предмет
1	КИТ
2	ТОХОД
3	Таможенные Информационные Технологии
4	Национальная Безопасность
5	Компьютерная Безопасность
6	Статистика
7	Таможенная Статистика

Содержание таблицы **Занятие**

КодПрепод	КодПредмет	День	Аудитория	КоличествоСтудентов	КоличествоПар
1	2	Вторник	301	15	1
1	2	Среда	301	16	2
1	2	Пятница	Акт Зал	60	1
2	6	Пятница	301	16	2
3	1	Вторник	301	14	1
3	1	Среда	301	12	2
4	1	Понедельник	205	14	2
4	1	Среда	205	14	2
4	5	Среда	301	16	1
5	1	Среда	212	15	2
5	1	Среда	301	15	1
6	4	Четверг	Акт Зал	60	1
7	4	Суббота	Акт Зал	60	1

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.



Запрос 1.1.

SELECT День, Аудитория
FROM Занятие
ORDER BY Аудитория

Запрос 1.2.

SELECT Препод.ФИО, Занятие.День, Занятие.Аудитория
FROM Препод **INNER JOIN**
(Предмет **INNER JOIN** Занятие
ON Предмет.КодПредмета = Занятие.КодПредмет**)**
ON Препод.КодПрепода = Занятие.КодПрепод
WHERE ((Занятие.День="Вторник"**) AND (**Занятие.Аудитория="301"**));**

Запрос 1.3.

SELECT Препод.ФИО, Занятие.Аудитория
FROM Препод **INNER JOIN** Занятие **ON** Препод.КодПрепода = Занятие.КодПрепод
WHERE (Занятие.Аудитория=[Задайте аудиторию])**;**

Запрос 1.4.

SELECT Аудитория, День, Sum(КоличествоПар) **AS** [Sum-КоличествоПар]
FROM Занятие
GROUP BY Аудитория, День;

Запрос 1.5.

TRANSFORM Count(КоличествоПар) **AS** [Count-КоличествоПар]
SELECT День, Count(КоличествоПар) **AS** [Итоговое значение КоличествоПар]
FROM Занятие
GROUP BY День
PIVOT Аудитория;

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит фамилия преподавателей, которые не ведут занятия в заданный день недели.

Запрос 2.2. Перекрестный запрос, вычисляющий количество студентов в каждой аудитории по каждому дню недели.

Запрос 2.3. Запрос на удаление занятий, проводимых в Актовом зале.

Запрос 2.4. Вывести дни недели, в которых занято заданное количество пар занятий.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.1** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\pi_{2,4}$ (Занятие) или Занятие[Код предмета, Аудитория].

4.2. Предприятие состоит из двух филиалов. В таблице КФ1 хранится информация о клиентах первого филиала. В таблице КФ2 хранится информация о клиентах второго филиала. Найти КФ1 ∪ КФ2. Описать смысловое значение полученного отношения.

КФ1

Название	*Город*	*Телефон*
Альвена	Минск	222-77-88
Сотис	Гродно	111-22-33
Гедеон	Брест	444-33-66
Октан	Минск	223-55-44
Белпак	Минск	223-11-22

КФ2

Название	*Город*	*Телефон*
Белпак	Минск	223-11-22
C&D	Витебск	555-88-33
Сотис	Гродно	111-22-33
Винтек	Минск	222-99-11

Вариант 4. База данных «Сувениры» имеет схему данных, представленную на рис. 5.6.

Рис. 5.6. Схема базы данных «Сувениры»

Содержание таблицы **Страна**

КодСтраны	Страна
1	Беларусь
2	Россия
3	Франция
4	Япония
5	США

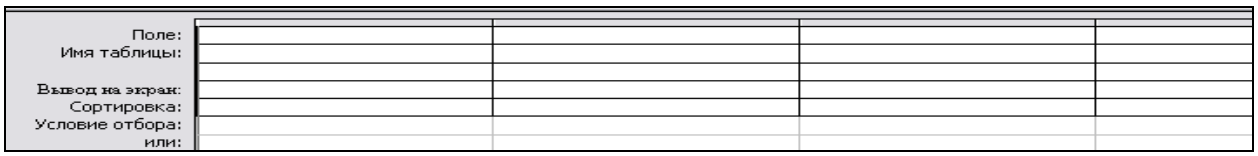
Содержание таблицы **Производитель**

КодПр	Производитель	Страна
1	Мир	Россия
2	Сябры	Беларусь
3	Светоч	Беларусь
4	Чио-Чио-Сан	Япония
5	DIOR	Франция

Содержание таблицы **Сувенир**

КодСувенира	Сувенир	ДатаВыпуска	Цена	КодПр
1	Матрешка	11.12.2009	80,00р.	1
2	Веер	22.11.2009	30,00р.	4
3	Книга	01.12.2009	20,00р.	3
4	Книга	06.10.2009	70,00р.	3
5	Календарь	12.10.2009	50,00р.	3
6	Часы	12.12.2009	120,00р.	2
7	Духи	01.12.2009	200,00р.	5
8	Елка	12.12.2009	20,00р.	2

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.



Запрос 1.1.

SELECT ДатаВыпуска, Сувенир
FROM Сувенир
ORDER BY ДатаВыпуска

Запрос 1.2.

SELECT Страна.Страна, Сувенир.Сувенир
FROM Страна **INNER JOIN**
(Производитель **INNER JOIN** Сувенир
ON Производитель.КодПр = Сувенир.КодПр**)**
ON Страна.КодСтраны = Производитель.Страна
WHERE ((Страна.Страна ="Беларусь"**) OR (**Страна.Страна ="Франция"**));**

Запрос 1.3.

SELECT Производитель.Производитель, Сувенир.Сувенир
FROM Производитель **INNER JOIN** Сувенир **ON** Производитель.КодПр = Сувенир.КодПр
WHERE (Сувенир.Цена<[Задайте цену])**;**

Запрос 1.4.

SELECT ДатаВыпуска, Sum(Цена) **AS** [Sum-Цена]
FROM Сувенир
GROUP BY ДатаВыпуска;

Запрос 1.5.

TRANSFORM Count(Сувенир) **AS** [Количество Сувениров]
SELECT ДатаВыпуска, Count(Сувенир) **AS** [Всего Сувениров]
FROM Сувенир
GROUP BY ДатаВыпуска
PIVOT Сувенир;

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит страны, которые не поставляли сувениры.

Запрос 2.2. Перекрестный запрос, вычисляющий суммы за сувениры в каждый день выпуска по каждому производителю.

Запрос 2.3. Запрос на удаление сувениров, выпущенных в ноябре.

Запрос 2.4. Вывести дни недели, в которых цена на сувениры была минимальной.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.4** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\pi_{2,4}$ (Сувенир) или Сувенир[Сувенир, Цена].

4.2. В таблице КФ1 хранится информация о производителях. В таблице КФ2 хранится информация о товарах. Найти КФ1 ⊗ КФ2. Описать смысловое значение полученного отношения.

КФ1

Название	*Город*
Альвена	Минск
Сотис	Гродно
Гедеон	Брест

КФ2

Название	*Цена*
шторы	200 у.е.
люстры	400 у.е.
плитка	50 у.е.

Вариант 5. База данных «Планеты» имеет схему данных, представленную на рис. 5.7.

Рис. 5.7. Схема базы данных «Планеты»

Содержание таблицы Небесное тело

Тело
Астероид
Комета
Планета

Содержание таблицы **Планета**

Тело	Вид	ОтСолнца	Диаметр	НаличиеАтмосферы	НаличиеЖизни
Венера	Планета	0,40		отсутствует	☐
Галлея	Комета	0,00	0		☐
Земля	Планета	1,00		наполовину облачность	☑
Марс	Планета	1,50		разреженная прозрачная	☐
Меркурий	Планета	0,70		сплошь облачность	☐
Нептун	Планета	30,10		сплошь облачность	☐
Паллада	Астероид	0,00	490		☐
Плутон	Планета	39,60		?	☐
Сатурн	Планета	9,50		сплошь облачность	☐
Уран	Планета	19,20		сплошь облачность	☐
Фая	Комета	0,00	0		☐
Цецера	Астероид	0,00	770		☐
Энке	Комета	0,00	0		☐
Юнона	Астероид	0,00	190		☐
Юпитер	Планета	5,20		сплошь облачность	☐

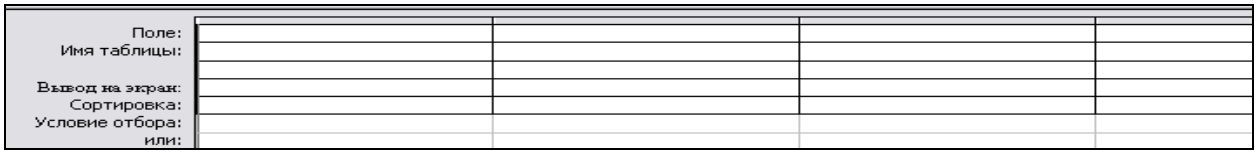
Содержание таблицы **Спутник**

Спутник	Планета	Диаметр	ГодОткрытия
Ариэль	Уран	950,00	1851
Деймос	Марс	8,00	1877
Европа	Юпитер	3000,00	1610
Ио	Юпитер	3700,00	1610
Луна	Земля	3476,00	
Мимас	Сатурн	650,00	1789
Рея	Сатурн	1750,00	1672
Фобос	Марс	15,00	1877

Содержание таблицы **Небесное тело**

Тело
Астероид
Комета
Планета

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.



Запрос 1.1.

SELECT Тело **AS** Планета
FROM Планета
WHERE Вид="Планета"
ORDER BY Тело**;**

Запрос 1.2.

SELECT Спутник.Спутник, Планета.Тело
FROM Планета **INNER JOIN** Спутник **ON** Планета.Тело = Спутник.Планета
WHERE ((Планета.Тело ="Юпитер") **OR** (Планета.Тело ="Сатурн"));

Запрос 1.3.

SELECT Спутник.Спутник, Планета.Тело, Планета.Вид
FROM (НебесноеТело **INNER JOIN** Планета **ON** НебесноеТело.Тело = Планета.Вид) **LEFT JOIN** Спутник **ON** Планета.Тело = Спутник.Планета
WHERE (((Спутник.Диаметр)<=3000)) **OR** (((Планета.Вид)="Астероид"));

Запрос 1.4.

SELECT Планета, MIN(Диаметр) **AS** [Расстояние до ближайшего спутника]
FROM Спутник
GROUP BY Планета;

Запрос 1.5.

SELECT Планета.Вид, Планета.Тело, Count(Спутник.Спутник) **AS** [Count-Спутник]
FROM (НебесноеТело **INNER JOIN** Планета **ON** НебесноеТело.Тело = Планета.Вид) **INNER JOIN** Спутник **ON** Планета.Тело = Спутник.Планета
GROUP BY Планета.Вид, Планета.Тело;

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит планеты и их спутники, открытые в 17 веке н.э.

Запрос 2.2. Найти планету, наиболее удаленную от Солнца..

Запрос 2.3. Найти планеты с признаками облачности.

Запрос 2.4. Вывести информацию о небесных телах, где нет признаков жизни.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.4** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\delta_{6=\text{Нет And } 2=\text{Планета}}$(Планета) или Планета(НаличиеЖизни = Нет и Вид=Планета)

4.2. В таблице КФ1 хранится информация о производителях. В таблице КФ2 хранится информация о товарах, в таблице КФ1_2 хранится информация о поставках. Найти КФ1 Devide By КФ2. Описать смысловое значение полученного отношения.

КФ1

Название	*Город*
Альвена	Минск
Сотис	Гродно
Гедеон	Брест

КФ2

Товар	*Цена*
шторы	200 у.е.
люстры	400 у.е.
плитка	50 у.е.

КФ1_2

Название	*Товар*
Альвена	шторы
Сотис	шторы
Альвена	люстры
Альвена	плитка
Сотис	плитка

Вариант 6. База данных «Военные корабли» имеет схему данных, представленную на рис. 5.8.

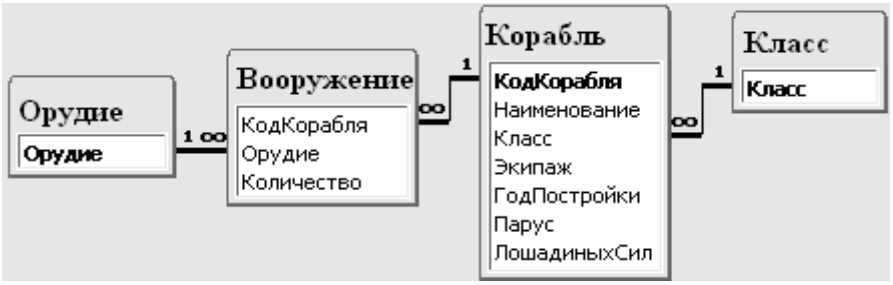


Рис. 5.8. Схема базы данных «Военные корабли»

Содержание таблицы **Класс**

Класс
Крейсер
Линкор
Миноносец
Подводная лодка

Содержание таблицы **Орудие**

Орудие
Мины
Орудие
Пулемет
Пушка
Торпедные аппараты
Автомат

Содержание таблицы **Корабль**

КодКорабля	Наименование	Класс	Экипаж	ГодПостройки	Парус	ЛошадиныеСилы
1	Полтава	Линкор	350	1712	☑	0
2	Св. Павел	Линкор	700	1794	☑	0
3	Петр Великий	Линкор	440	1872	☐	8258
4	Севастополь	Линкор	1126	1911	☐	42000
5	Бородино	Крейсер	800	1901	☐	18000
6	Буйный	Миноносец	74	1902	☐	5700
7	Сторожевой	Миноносец	250	1939	☐	60000
8	Аврора	Крейсер	570	1900	☐	11610

Содержание таблицы **Вооружение**

КодКорабля	Орудие	Количество
1	Пушка	54
2	Пушка	90
3	Орудие	12
4	Орудие	32
4	Торпедные аппараты	4
5	Орудие	16
6	Пушка	6
7	Мины	0
7	Орудие	4
7	Пушка	2
7	Пулемет	4
8	Торпедные аппараты	3
8	Пушка	32
8	Орудие	8
6	Торпедные аппараты	3

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.

Поле:				
Имя таблицы:				
Вывод на экран:				
Сортировка:				
Условие отбора:				
или:				

Запрос 1.1.

SELECT Наименование **AS** Корабль, Орудие, Количество
FROM Корабль **INNER JOIN** Вооружение **ON**
Корабль.КодКорабля=Вооружение.КодКорабля
WHERE Класс="Линкор"
ORDER BY Количество**;**

Запрос 1.2.

SELECT Класс.Класс, Sum(Вооружение.Количество) **AS** [Sum-Количество]
FROM Класс **INNER JOIN** (Корабль **INNER JOIN** Вооружение **ON** Корабль.КодКорабля = Вооружение.КодКорабля) **ON** Класс.Класс = Корабль.Класс
WHERE ((Корабль.Парус)=No)
GROUP BY Класс.Класс;

Запрос 1.3.

```
DELETE Корабль.*
FROM Класс INNER JOIN Корабль ON Класс.Класс = Корабль.Класс
WHERE (((Класс.Класс)="Миноносец"));
```

Запрос 1.4.

```
SELECT  Корабль.Наименование
FROM Корабль
WHERE Экипаж=(SELECT Min(Корабль.Экипаж) FROM Корабль);
```

Запрос 1.5.

```
TRANSFORM Sum(Вооружение.Количество) AS [Sum-Количество]
SELECT Корабль.Класс, Sum(Вооружение.Количество) AS [Итого Вооружения]
FROM Корабль INNER JOIN Вооружение ON
        Корабль.КодКорабля = Вооружение.КодКорабля
GROUP BY Корабль.Класс
PIVOT Вооружение.Орудие;
```

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который выводит корабли, выпущенные в 20 веке н.э.

Запрос 2.2. Вывести информацию о вооружении миноносцев.

Запрос 2.3. Найти самый мощный корабль.

Запрос 2.4. Найти корабли, имеющие в вооружении торпедные аппараты.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.4** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\delta_{3=\text{Крейсер And } 5=1900}$(Корабль) или Корабль(Класс = Крейсер и ГодПостройки=1900).

4.2. В таблице К хранится информация о размещении сотрудников по кабинетам. В таблице Схранится информация о сотрудниках. Какой оператор соединения используется для результата, который получен в таблице К __ JOIN С. Описать смысловое значение полученного отношения.

К

Кабинет	*Сотрудник*
1	1
2	4
1	2

С

Сотрудник	*Фамилия*
1	Иванов
2	Петров
3	Сидоров

К__JOIN С

Кабинет	*Фамилия*
1	Иванов
1	Петров
	Сидоров

Вариант 7. База данных «Файловая система» имеет схему данных, представленную на рис. 5.9.

Рис. 5.9. Схема базы данных «Файловая система»

Содержание таблицы **Файл**

Код	Имя	Тип	Длина
1	IMG_0855	jpeg	1043675
2	IMG_0865	jpeg	1017453
3	лекция	doc	2176582
4	конспект по СУБД	Adobe Acrobat	88314806
5	metod_TOXOD	hlp	23444
6	Lab_Excel	xls	18666

Содержание таблицы **Дерево**

Файл	Папка
IMG_0855	Фото
IMG_0865	Фото
лекция	Documents
конспект по СУБД	Documents
metod_TOXOD	Tanya
Lab_Excel	Tanya

Содержание таблицы **Каталог**

КодКаталога	Папка	Родитель	Диск
1	Program Files		d:
2	Documents		e:
3	Nick		e:
4	Tanya		flash
5	DrWeb	Program Files	d:
6	MSOffice	Program Files	d:
7	Фото	Nick	e:

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.

Поле:				
Имя таблицы:				
Вывод на экран:				
Сортировка:				
Условие отбора:				
или:				

Запрос 1.1.

SELECT Каталог.Диск, Каталог.Родитель, Каталог.Папка, Файл.Имя, Файл.Тип, Файл.Длина

FROM Файл **INNER JOIN** (Каталог **INNER JOIN** Дерево
ON Каталог.КодКаталога = Дерево.Папка)
ON Файл.Код = Дерево.Файл;

Запрос 1.2.

SELECT Дерево.Папка, Count(Файл.Имя) **AS** [Count-Имя]
FROM Файл **INNER JOIN** Дерево **ON** Файл.Код = Дерево.Файл
GROUP BY Дерево.Папка;

Запрос 1.3.

SELECT [Каталог]![Диск] & "\" & iif([Каталог]![Родитель]<>null, [Каталог]![Родитель] & "\") & [Каталог]![Папка] & "\" & [Файл]![Имя] **AS** Путь
FROM Файл **INNER JOIN** (Каталог **INNER JOIN** Дерево **ON** Каталог.КодКаталога = Дерево.Папка) **ON** Файл.Код = Дерево.Файл**;**

Запрос 1.4.

SELECT Папка **INTO** Подкаталоги
FROM Каталог
WHERE Родитель**<>**null**;**

Запрос 1.5.

TRANSFORM Sum(Файл.Длина) **AS** [Sum-Длина]
SELECT Каталог.Диск
FROM Каталог **INNER JOIN** (Файл **INNER JOIN** Дерево **ON** Файл.Код = Дерево.Файл) **ON** Каталог.КодКаталога = Дерево.Папка
GROUP BY Каталог.Диск
PIVOT Каталог.Папка**;**

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который подсчитает, сколько файлов на диске (имя диска задать как параметр).

Запрос 2.2. Найти самый длинный файл.

Запрос 2.3. Найти папки, не имеющие вложенных каталогов.

Запрос 2.4. Подсчитать место, занимаемое на каждом диске содержимым файлов.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.4** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\delta_{4=flash}$(Каталог) или Каталог(Диск = flash)

4.2. В таблице **К** хранится информация о размещении сотрудников по кабинетам. В таблице **С** хранится информация о сотрудниках. Какой оператор соединения используется для результата, который получен в таблице **К __ JOIN С**. Описать смысловое значение полученного отношения.

К

Кабинет	*Сотрудник*
1	1
2	4
1	2

С

Сотрудник	*Фамилия*
1	Иванов
2	Петров
3	Сидоров

К__JOIN С

Кабинет	*Фамилия*
1	Иванов
1	Петров
2	

Вариант 8. База данных «Дети» имеет схему данных, представленную на рис. 5.10.

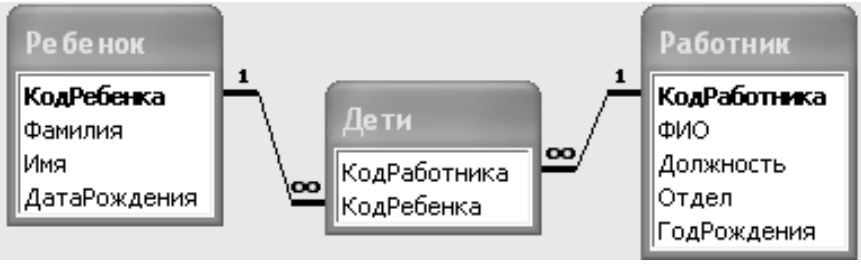


Рис. 5.10. Схема базы данных «Дети»

Содержание таблицы **Работник**

КодРаботника	ФИО	Должность	Отдел	ГодРождения
1	Разоренова Т.Р.	ст. преподаватель	Таможенное дело	1955
2	Галай Т.А.	ст. преподаватель	Таможенное дело	1965
3	Пищик Т.В.	доцент	Таможенное дело	1970
4	Копко Ю.А.	преподавтель	Таможенное дело	1977
5	Бухвалова И.А.	ст. преподаватель	ПОВТиАС	1957
6	Кучерявенко Л.И.	ст. преподаватель	ПОВТиАС	1955
7	Разоренов Н.А.	доцент	ПОВТиАС	1948
8	Гурский Н.Н.	доцент	ПОВтиАС	1954
9	Гурская Л.Б.	инженер	ПОВТиАС	1950

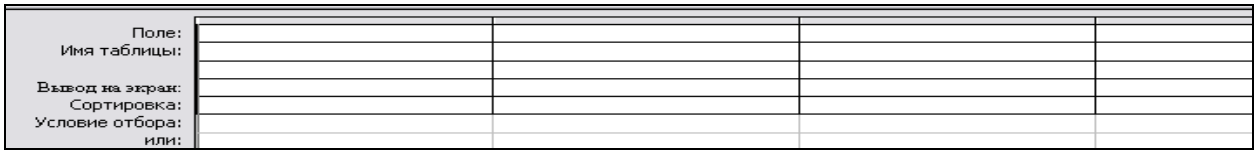
Содержание таблицы **Ребенок**

КодРебенка	Фамилия	Имя	ДатаРождения
1	Разоренов	Дмитрий	05.03.1983
2	Галай	Виталий	19.06.1985
3	Пищик	Иван	18.08.1992
4	Пищик	Алина	21.12.1995
5	Дежурко	Мария	11.11.1989
6	Дежурко	Юля	03.05.1985
7	Голубева	Александра	12.12.1988
8	Кучерявенко	Полина	05.06.1990
9	Смитт	Анна	13.01.1982

Содержание таблицы **Дети**

КодРаботника	КодРебенка
5	5
5	6
2	2
9	7
8	7
6	8
6	9
3	4
3	3
7	1
1	1

1. По записи на языке SQL нарисовать бланки запросов в режиме *Конструктора* и **описать действия**, выполняемые данными запросами.



Запрос 1.1.

SELECT Работник.ФИО, Ребенок.Фамилия, Ребенок.Имя
FROM Ребенок **INNER JOIN** (Работник **INNER JOIN** Дети
ON Работник.КодРаботника = Дети.КодРаботника)
ON Ребенок.КодРебенка = Дети.КодРебенка;

Запрос 1.2.

```
SELECT Работник.ФИО, Count(Дети.КодРебенка) AS [Сколько детей]
FROM Работник INNER JOIN Дети ON Работник.КодРаботника = Дети.КодРаботника
GROUP BY Работник.ФИО;
```

Запрос 1.3.

```
SELECT (Left([ФИО], InStr([ФИО]," ")-1)) AS фамилия
FROM Работник
WHERE КодРаботника IN (SELECT КодРаботника FROM Дети);
```

Запрос 1.4.

```
SELECT Ребенок.Фамилия, Ребенок.Имя,
        MonthName(Month([ДатаРождения]),1) AS [Месяц Рождения]
FROM Ребенок;
```

Запрос 1.5.

```
TRANSFORM Count(Фамилия) AS [Количество Детей]
SELECT ФИО, Count(Фамилия) AS [Всего Детей]
FROM Ребенок INNER JOIN (Работник INNER JOIN Дети
ON Работник.КодРаботника = Дети.КодРаботника)
ON Ребенок.КодРебенка = Дети.КодРебенка
GROUP BY ФИО
PIVOT Отдел;
```

2. Разработать запросы **на языке SQL** и нарисовать бланки этих запросов в режиме *Конструктора*.

Запрос 2.1. Запрос, который подсчитает, сколько детей в каждом отделе.

Запрос 2.2. Найти самого взрослого ребенка

Запрос 2.3. Найти тех, кто не имеет детей.

Запрос 2.4. Кому надо сделать новогодний подарок, если их дают детям до 10 лет.

3. Для запросов **1.1**, **1.3**, **1.4**, **2.4** нарисовать таблицы результатов.

4. Операции реляционной алгебры:

4.1. Нарисовать таблицу результатов для операции $\delta_{З=Дмитрий\ Or\ З=Юля}$(Ребенок) или Ребенок(Имя = Дмитрий или Имя=Юля)

4.2. В таблице К хранится информация о размещении сотрудников по кабинетам. В таблице С хранится информация о сотрудниках. Какой оператор соединения используется для результата, который получен в таблице К __ JOIN С. Описать смысловое значение полученного отношения.

К

Кабинет	*Сотрудник*
1	1
2	4
1	2

С

Сотрудник	*Фамилия*
1	Иванов
2	Петров
3	Сидоров

К__JOIN С

Кабинет	*Фамилия*
1	Иванов
1	Петров

Тема 6. Элементы автоматизации приложения

Теоретические сведения

При разработке приложений используется множество объектов, таких как формы, отчеты, таблицы, элементы управления, наборы записей и т.д. Управление этими объектами основывается на установке их свойств и вызове методов, то есть функций, определяющих поведение объектов. Событийная модель программирования предполагает создавать процедуры, вызываемые при обработке событий, происходящих в процессе работы пользователя с приложением.

Стандартные модули содержат общие процедуры, которые могут использоваться и для вычислительных процессов, и при обработке событий. При разработке *функций* на VBA тело функции заключается между операторами ***Function*** и ***End Function***. Аналогично *программа* VBA является фрагментом программного кода, заключенного между операторами ***Sub*** и ***End Sub***. Модули форм и отчетов связаны с соответствующими формами и отчетами.

Для хранения значений, выполнения вычислений и передачи параметров в процедуры и функции используются *переменные*. Тип данных в объявлении переменной указывает компилятору, какое количество памяти необходимо резервировать для этой переменной и какие операции являются для нее допустимыми. Как и в других языках программирования, в VBA определены операции с числами, строками, датами и логическими значениями. Для управления в программах используются конструкции ветвления ***If…Then…Else*** и оператор выбора ***Select Case***. При простом ветвлении удобно использование функции ***IIf,*** которая имеет формат:

IIf(<выражение>,<значениеЕслиДа>,<значение ЕслиНет>).

Операторы цикла в VBA можно разделить на три группы:

– циклы с условием (***Do…Loop***);

– циклы со счетчиком (***For…Next***);

– циклы по структуре данных (***For Each…Next***).

Для вызова подпрограмм и функций можно использовать следующие формы:
<имяПроцедуры> <список ФактическихПараметров> или
Call *<имяПроцедуры>* ***(****<список ФактическихПараметров>****)*** или
<имяПеременной>=<имяФункции> ***(****<список ФактическихПараметров>****)***.

Подробнее основы языка VBA рассматриваются в соответствующих разделах курсов “Компьютерные информационные технологии” и “Основы программирования”, здесь же приобретаются навыки использования ранее написанных функций и адаптация предлагаемых кодов к применению в разрабатываемом приложении, поэтому строки кода VBA комментируются описанием тех действий, которые они выполняют при автоматизации работы приложения.

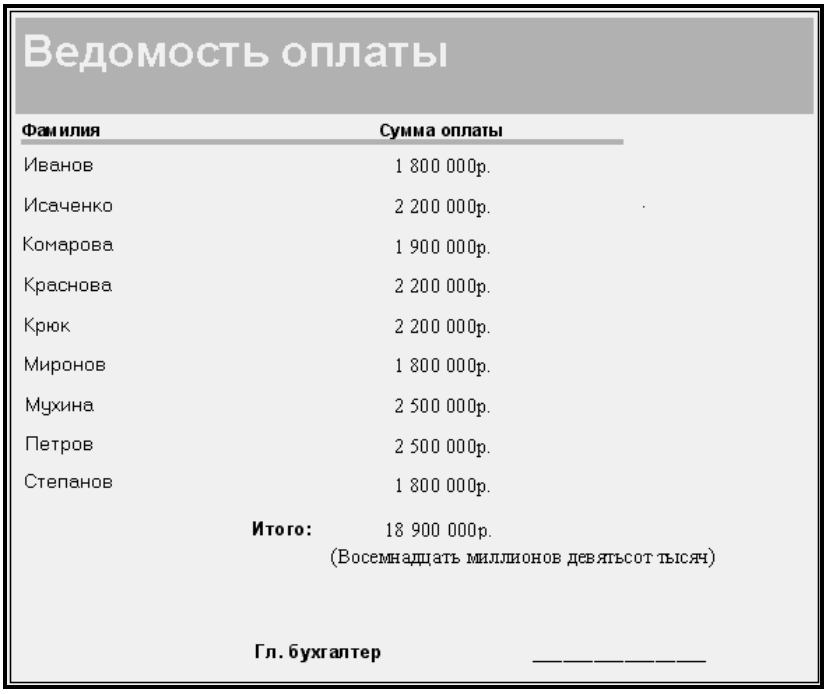
Объектная модель Microsoft Access 2002 представлена такими объектами, как ***Application*** (Приложение Access), ***Forms*** (Формы), ***Reports*** (Отчеты), ***Controls*** (Элементы управления), ***DoCmd*** (Выполнить макрокоманду), ***CurrentData*** (Объекты базы данных: таблицы – ***AllTables***, запросы – ***AllQueries***, и.т.д.). Доступ к объектам, входящим в семейство, возможен через упоминание имени семейства и различных операций с семействами, установке или получении конкретных свойств объекта и вызове их методов.

Для быстроты разработки желательно использовать электронный вариант лабораторной работы, поскольку можно не набирать достаточно объемный код, а скопировать его и адаптировать в своем приложении. При выполнении заданий следует придерживаться той последовательности выполнения, которая позволит создать проект приложения и внести корректировки в модули форм и отчетов, предлагаемых в данном пособии. Каждое задание ставит определенную цель автоматизации работы с приложением и пошагово позволяет достичь желаемого результата.

Практические задания

Задание 6.1. Разработка пользовательской функции.

Требуется получить ведомость оплаты за обучение с подсчетом итоговой суммы и прописыванием ее текстовой прописью, по примеру, представленному на рисунке 6.1.



Ведомость оплаты

Фамилия	Сумма оплаты
Иванов	1 800 000р.
Исаченко	2 200 000р.
Комарова	1 900 000р.
Краснова	2 200 000р.
Крюк	2 200 000р.
Миронов	1 800 000р.
Мухина	2 500 000р.
Петров	2 500 000р.
Степанов	1 800 000р.
Итого:	18 900 000р.

(Восемнадцать миллионов девятьсот тысяч)

Гл. бухгалтер _____

*Рис. 6.1. Отчет **Ведомость оплаты** в режиме просмотра*

Рекомендации по выполнению:

1. Создать запрос *Платники*, в котором выбрать всех студентов, обучающихся на внебюджете и которые вносят плату за образование.

2. На основе этого запроса разработать отчет по примеру, представленному на рис. 6.2.

3. В режиме *Конструктора* в области примечания отчета следует вставить три элемента управления: два поля **a** и **b** и надпись **c**.

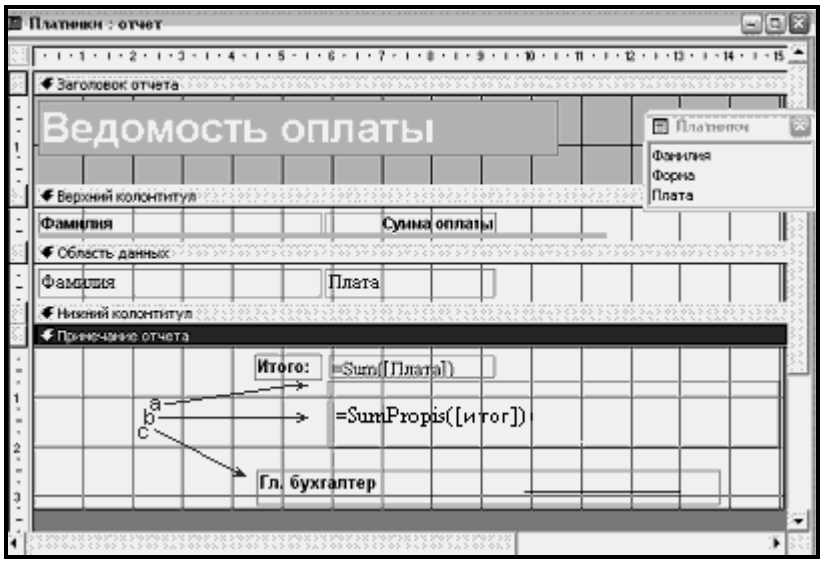


Рис. 6.2. Вид отчета в Конструкторе

3.1. В поле **a** с помощью встроенной функции подсчитывается общая сумма по полю **Плата**. В окне *свойств* надо присвоить полю *имя* значение **итог**, а в качестве *данных* использовать функцию **Sum** (рис. 6.3). Подпись надписи поля изменить на **Итого**.

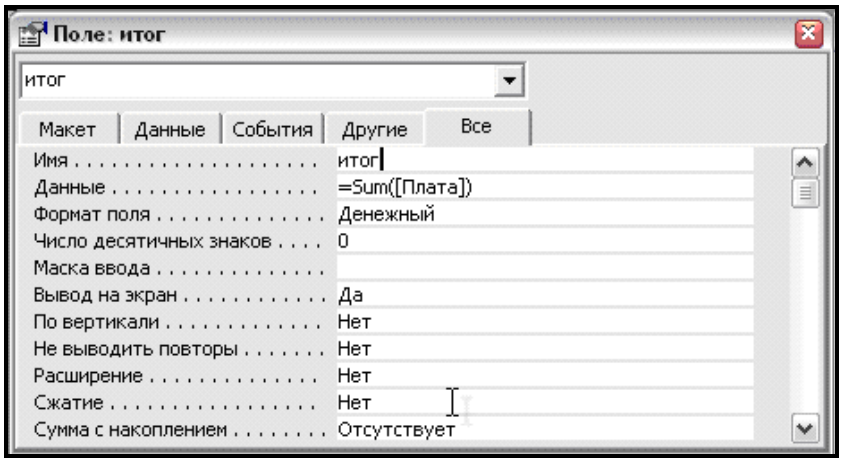


Рис. 6.3. Свойства поля ***a***

3.2. В поле **b** надпись поля следует выделить и удалить, а для самого поля присвоить *имя* ***пропись*** (рис. 6.4). В качестве данных будем использовать пользовательскую функцию **SumPropis**, которая осуществляет прописывание прописью числового данного, заданного в качестве ее аргумента. Поскольку поле **итог** может содержать достаточно большое число, для текстовой записи может не хватить предусмотренного размера поля, который можно расширить, если установить параметр свойства *Расширение* в значение **Да**.

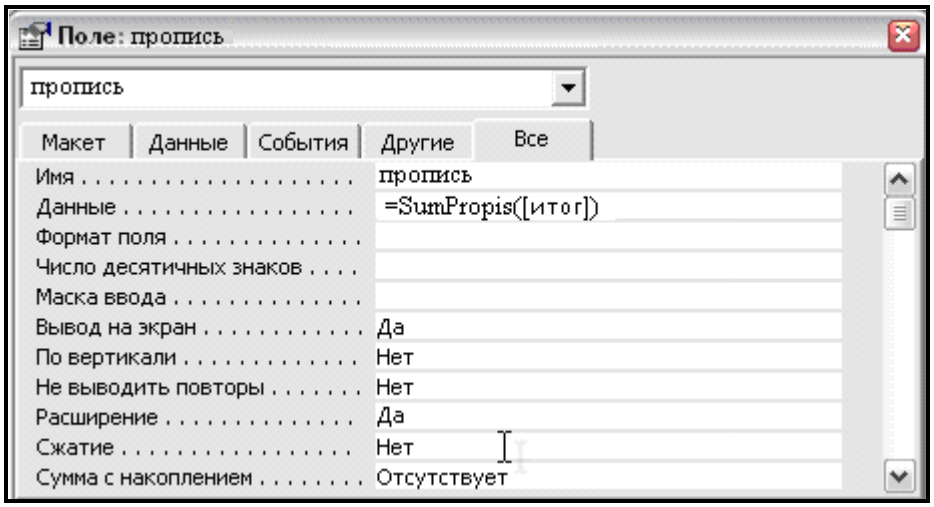


Рис. 6.4. Свойства поля ***b***

В стандартный модуль проекта VBA следует поместить программный код функции *SumPropis* и ряд вспомогательных функций, выделяющих из исходного числа единицы, десятки, сотни, тысячи и другие разряды, и определяющих род для корректного прописывания текстом цифр в числовом значении.

Исходный код можно позаимствовать из файла *Functions_6.txt.* Скопируйте в *Буфер Обмена* приведенный ниже на рис. 6.5 код, перейдите на вкладку **Модули** в окне объектов Access, нажмите кнопку **Создать** (в Access_2007 раскрыть кнопку **Макрос** на закладке **Создание**). Откроется окно модуля VBA, в который и вставьте скопированный фрагмент.

```
  Dim Сумма As Currency
  Dim Остаток As Currency

  Function Единицы(Разряд As Currency, Род
As String) As String
      Select Case Разряд
          Case 1
              If Род = "Мужской" Then
                  Единицы = "один "
              Else
                  Единицы = "одна "
              End If
          Case 2
              If Род = "Мужской" Then
                  Единицы = "два "
              Else
                  Единицы = "две "
              End If
          Case 3
              Единицы = "три "
          Case 4
              Единицы = "четыре "
          Case 5
              Единицы = "пять "
          Case 6
              Единицы = "шесть "
          Case 7
              Единицы = "семь "
          Case 8
              Единицы = "восемь "
          Case 9
              Единицы = "девять "
          Case 10
              Единицы = "десять "
          Case 11
              Единицы = "одиннадцать "
          Case 12
              Единицы = "двенадцать "
          Case 13
              Единицы = "тринадцать "
          Case 14
              Единицы = "четырнадцать "
          Case 15
              Единицы = "пятнадцать "
          Case 16
              Единицы = "шестнадцать "
          Case 17
              Единицы = "семнадцать "
          Case 18
              Единицы = "восемнадцать "
          Case 19
              Единицы = "девятнадцать "
      End Select
  End Function

  Function Десятки(Разряд As Currency) As
String
      Select Case Разряд
          Case 2
              Десятки = "двадцать "
          Case 3
              Десятки = "тридцать "
          Case 4
              Десятки = "сорок "
          Case 5
              Десятки = "пятьдесят "
          Case 6
              Десятки = "шестьдесят "
          Case 7
              Десятки = "семьдесят "
          Case 8
              Десятки = "восемьдесят "
          Case 9
              Десятки = "девяносто "
      End Select
  End Function


  Function Сотни(Разряд As Currency) As
String
      Select Case Разряд
          Case 1
              Сотни = "сто "
          Case 2
              Сотни = "двести "
          Case 3
              Сотни = "триста "
          Case 4
              Сотни = "четыреста "
          Case 5
              Сотни = "пятьсот "
          Case 6
              Сотни = "шестьсот "
          Case 7
              Сотни = "семьсот "
          Case 8
              Сотни = "восемьсот "
          Case 9
              Сотни = "девятьсот "
      End Select
  End Function

  Function Тысячи(Разряд As Currency) As
String
      If Разряд = 1 Then
          Тысячи = "тысяча "
      ElseIf Разряд > 1 And Разряд < 5
Then
          Тысячи = "тысячи "
      Else
          Тысячи = "тысяч "
      End If
  End Function


  Function Миллионы(Разряд As Currency) As
String
      If Разряд = 1 Then
          Миллионы = "миллион "
      ElseIf Разряд > 1 And Разряд < 5
Then
          Миллионы = "миллиона "
      Else
          Миллионы = "миллионов "
      End If
  End Function

  Function Миллиарды(Разряд As Currency)
As String
      If Разряд = 1 Then
          Миллиарды = "миллиард "
      ElseIf Разряд > 1 And Разряд < 5
Then
          Миллиарды = "миллиарда "
      Else
          Миллиарды = "миллиардов "
      End If
  End Function
```

Рис. 6.5. Код стандартного модуля (файл Functions_6.txt)

```
Public Function SumPropis(СуммаЦифрами As Currency) As String
Dim Группа As Currency, Разряд As Currency, Длина As Currency
Dim Пропись As String
    Сумма = СуммаЦифрами
    Остаток = Сумма
    Группа = Int(Остаток / 1000000000)
    If Группа <> 0 Then
        Разряд = Группа \ 100
        Пропись = Пропись & Сотни(Разряд)
        Остаток = Остаток - Разряд * 100 * 1000000000
        Группа = Группа - Разряд * 100
        If Группа > 19 Then
            Разряд = Группа \ 10
            Пропись = Пропись & Десятки(Разряд)
            Остаток = Остаток - Разряд * 10 * 1000000000
            Группа = Группа - Разряд * 10
        End If
        Разряд = Группа
        Пропись = Пропись & Единицы(Разряд, "Мужской")
        Остаток = Остаток - Разряд * 1000000000
        Пропись = Пропись & Миллиарды(Разряд)
    End If
    Группа = Int(Остаток / 1000000)
    If Группа <> 0 Then
        Разряд = Группа \ 100
        Пропись = Пропись & Сотни(Разряд)
        Остаток = Остаток - Разряд * 100 * 1000000
        Группа = Группа - Разряд * 100
        If Группа > 19 Then
            Разряд = Группа \ 10
            Пропись = Пропись & Десятки(Разряд)
            Остаток = Остаток - Разряд * 10 * 1000000
            Группа = Группа - Разряд * 10
        End If
        Разряд = Группа
        Пропись = Пропись & Единицы(Разряд, "Мужской")
        Остаток = Остаток - Разряд * 1000000
        Пропись = Пропись & Миллионы(Разряд)
    End If
    Группа = Int(Остаток / 1000)
    If Группа <> 0 Then
        Разряд = Группа \ 100
        Пропись = Пропись & Сотни(Разряд)
        Остаток = Остаток - Разряд * 100 * 1000
        Группа = Группа - Разряд * 100
        If Группа > 19 Then
            Разряд = Группа \ 10
            Пропись = Пропись & Десятки(Разряд)
            Остаток = Остаток - Разряд * 10 * 1000
            Группа = Группа - Разряд * 10
        End If
        Разряд = Группа
        Пропись = Пропись & Единицы(Разряд, "Женский")
        Остаток = Остаток - Разряд * 1000
        Пропись = Пропись & Тысячи(Разряд)
    End If
    Группа = Остаток
    If Группа <> 0 Then
        Разряд = Группа \ 100
        Пропись = Пропись & Сотни(Разряд)
        Остаток = Остаток - Разряд * 100
        Группа = Группа - Разряд * 100
        If Группа > 19 Then
            Разряд = Группа \ 10
            Пропись = Пропись & Десятки(Разряд)
            Остаток = Остаток - Разряд * 10
            Группа = Группа - Разряд * 10
        End If
```

Рис. 6.5. (продолжение)

```
            Разряд = Группа
            Пропись = Пропись & Единицы(Разряд, "Мужской")
            Остаток = Остаток - Разряд
        End If
        Длина = Len(Пропись)
        If IsNull(Длина) Then
        Exit Function
        End If
        Пропись = UCase(Mid(Пропись, 1, 1)) & (Mid(Пропись, 2, Длина - 2))
        SumPropis = "(" & Пропись & ") "
    End Function
```

Рис. 6.5. (окончание)

3.3. Перейдите в режим *Конструктора*, создайте надпись **с** и введите текст для подписи бухгалтера.

4. Просмотрите отчет в режиме предварительного просмотра перед печатью и убедитесь в полученном результате.

Задание 6.2. Использование VBA для связи приложений.

При работе в офисных приложениях часто приходится формировать документы в текстовом редакторе Word, куда необходимо переносить информацию, хранящуюся в базе данных. Технология *автоматизации* позволяет управлять этим процессом на программном уровне.

Рассмотрим *бизнес-ситуацию*. Студентам в университете можно получать справку о том, где они учатся (с указанием факультета, специальности, курса, группы и фамилии и др. информации). Вид справки представлен ниже на рис. 6.6.

СПРАВКА

Студент _ факультета _ обучается по специальности _ в группе _ на _ курсе.

Декан факультета /_____/

(фамилия, и. о.)

Рис. 6.6. Вид справки из деканата

В места подчеркивания вставляется информация, которая хранится в базе данных «Студенты». Перенос информации вручную неэффективен, поэтому необходимо автоматизировать этот процесс.

Рекомендации по выполнению:

Первое, что надо сделать – это создать в Word документ подобного содержания и вида (рис. 6.7), только в места подчеркивания потребуется вставить ***закладки*** (отмечены серой чертой) с понятными именами (добавляются в диалоговом окне *Закладка*), а сам документ сохранить в папке на диске (предположим, в головной папке диска **D:**) под именем **Справка.doc**.

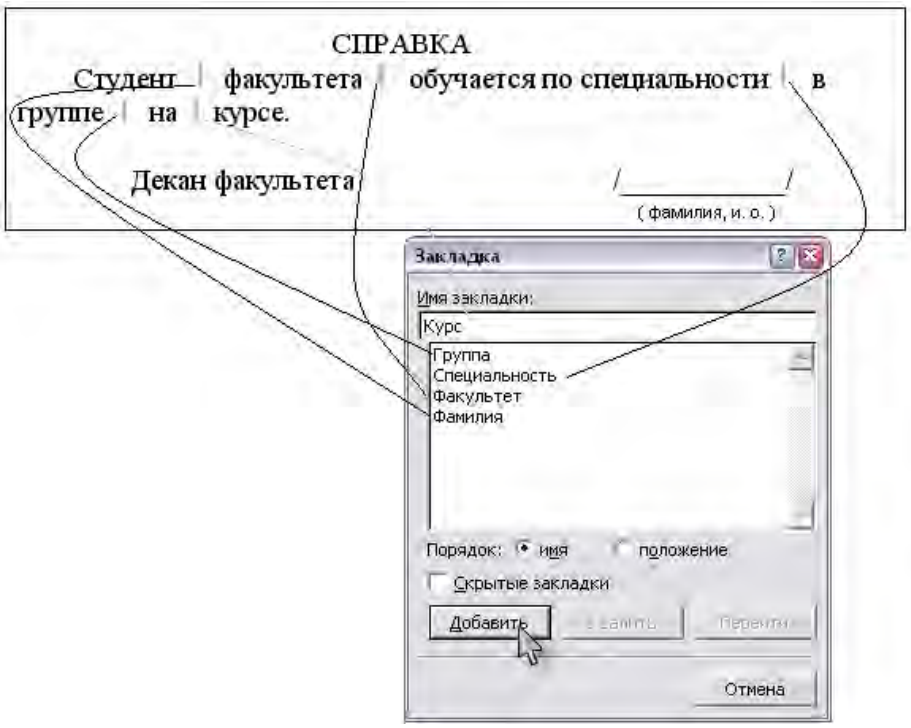


Рис. 6.7. Документ справки с закладками

Второе, что потребуется сделать – это разработать ***сложную форму***, представленную на рис. 6.8, с подчиненными ***подформами***, которые обеспечат удобный просмотр структурированной информации из базы данных.

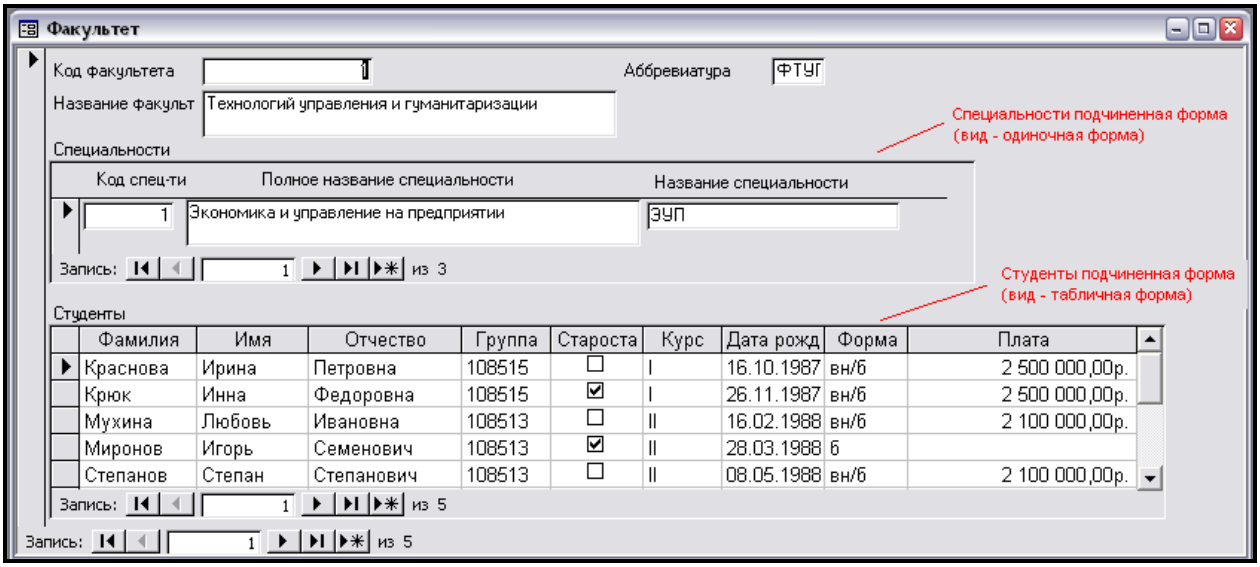


Рис. 6.8. Сложная форма

Работа с такой формой позволяет выбирать факультет, листать список специальностей и для каждой просматривать списки студентов. Форма разрабатывается ***Мастером***, в котором вначале загружаются все поля из таблицы **Факультет**, затем все поля, кроме **Код факультета**, из таблицы **Специальности**, и, наконец, все поля, кроме **Код специальности** и **Фото**, из таблицы **Студенты**.

На рис. 6.9–6.12 продемонстрированы шаги Мастера по выбору полей для создания сложной формы, вида представления подчиненных форм и стиля главной формы.

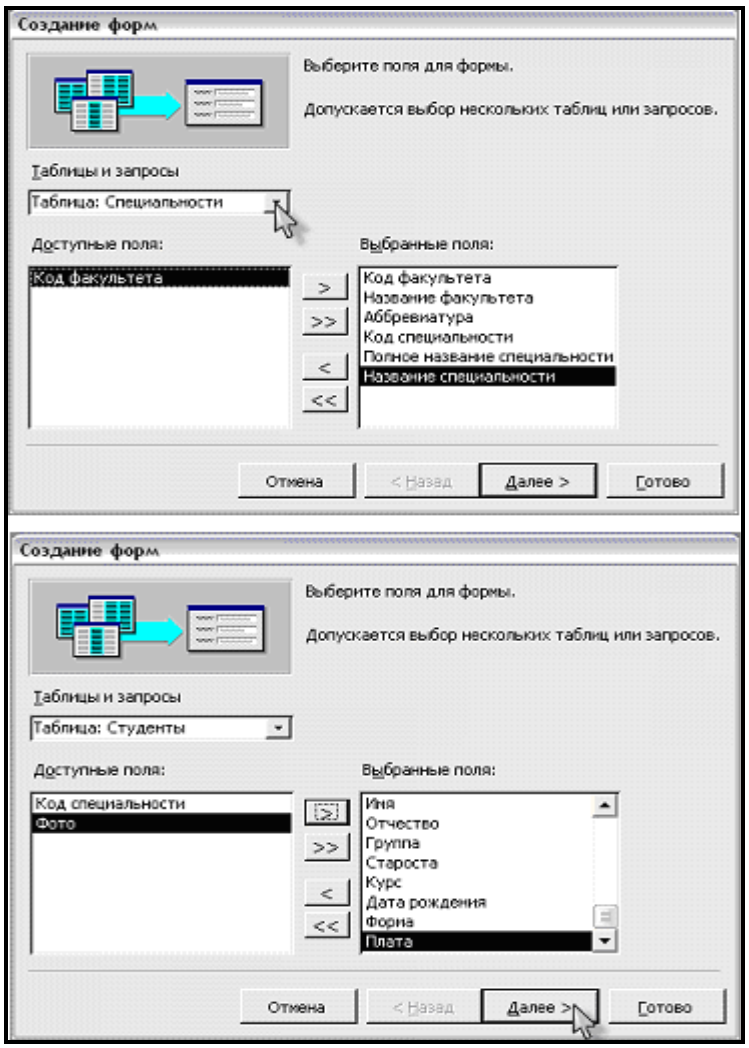


Рис. 6.9. Выбор полей из таблицы Специальности и из таблицы Студент

Выбираем подчиненный вид вложенных форм, внешний вид (ленточный для Специальностей и табличный для Студентов) и стиль Стандартный.

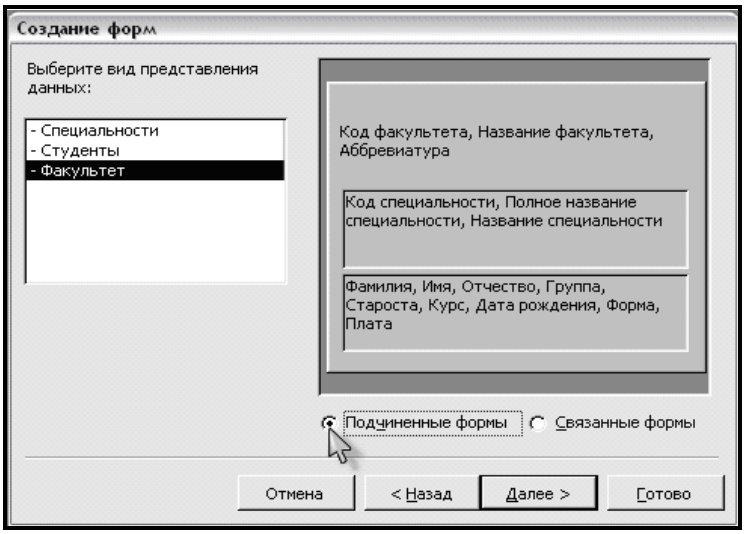


Рис. 6.10. Выбор вида представления данных

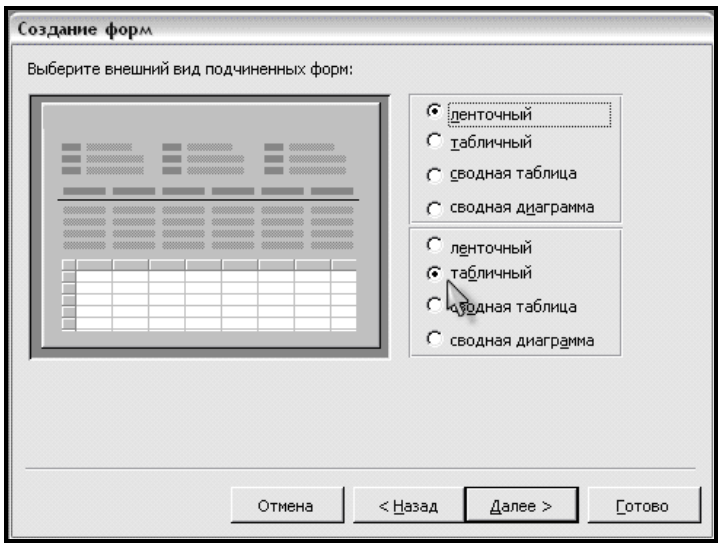


Рис. 6.11. Выбор вида подчиненных форм

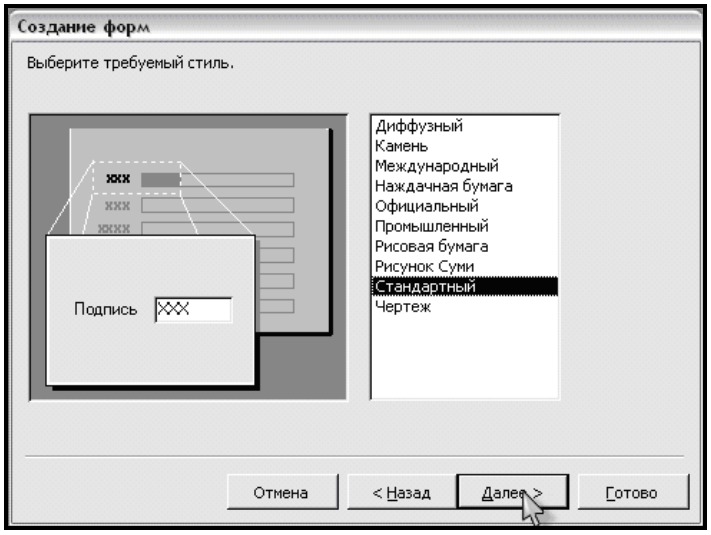


Рис. 6.12. Выбор стиля формы

Соглашаемся с именами форм, задаваемыми по умолчанию и открываем форму для просмотра и ввода данных.

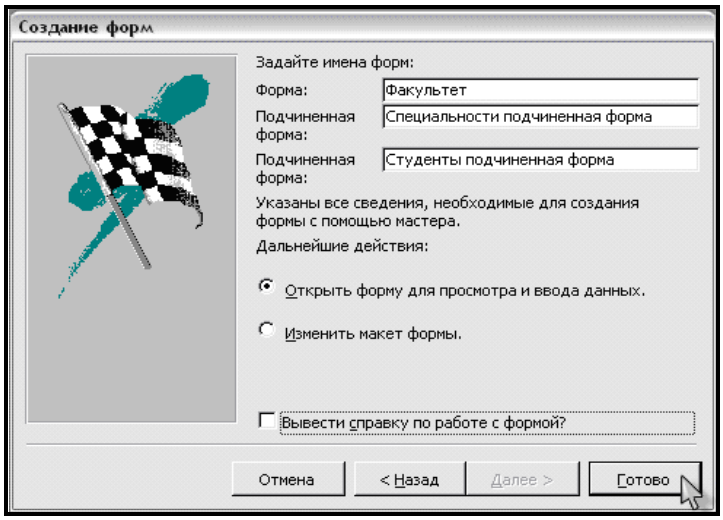


Рис. 6.13. Задание имен формам

В режиме **Конструктора** можно видоизменить ширину полей ленточной формы **Специальности** и сменить вид на *одиночную форму* – тогда в подчиненной форме будет отображаться информация об одной специальности из выбранных в строке прокрутки списка специальностей. Для табличной формы **Студенты** можно уменьшить ширину полей прямо в режиме работы с формой.

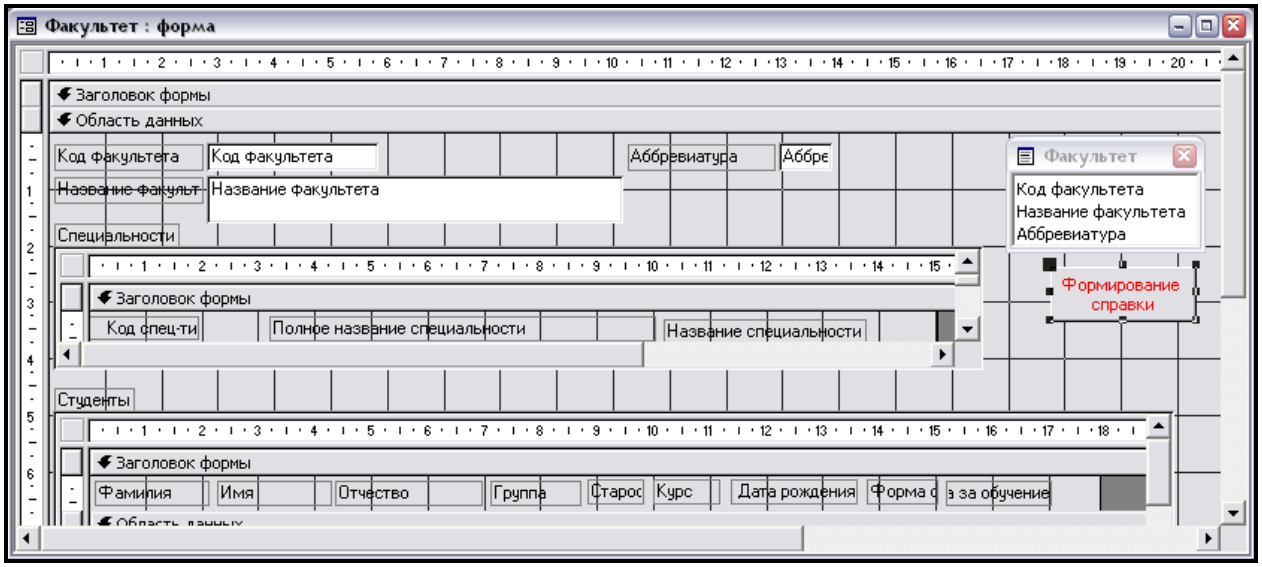


Рис. 6.14. Форма в режиме Конструктора

Создадим на форме кнопку, нажатие на которую обеспечит доступ к файлу **D:\Справка.doc**, вставив при этом поля текущей записи из формы в места закладок.

Перейдем в режим **Конструктора** для главной формы **Факультет**, выберем инструмент **Кнопка** (отключим Мастер!!!) и начертим кнопку в свободной области формы. Сменим название кнопки: оно должно соответствовать действию – **Формирование справки**.

Двойным щелчком по созданной кнопке перейдем в окно выбора обработчика событий, выберем **Нажатие кнопки**, вызовем в **Построителе** режим создания **Программы** и перейдем в редактор VBA, где появится «заглушка» программы – обработчика события нажатия на эту кнопку (**Click**), внутрь которой вставьте приведенный ниже код (в нем даны комментарии). Текст кода можно скопировать из файла *ButtonClick_6.txt*. Обратите внимание, какие длинные конструкции определяют элементы управления главной и подчиненных форм.

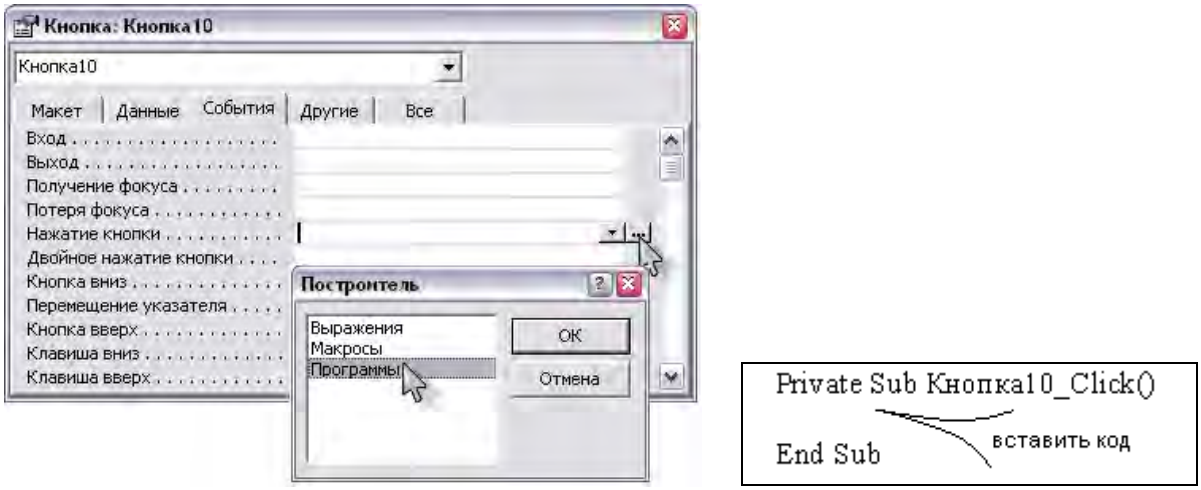


*Рис. 6.15. Построение программы для обработки события нажатия на кнопку (**Click**)*

В приведенном ниже на рис. 6.16 коде организован доступ к файлу с закладками, которые по очереди выделяются и вместо них из полей сложной формы методом *TypeText* вводятся значения активной записи, на которую указывает маркер. Используются коллекции для работы с объектами *Bookmarks* – закладки и *Forms* – формы.

```
    '  переменная wda ссылается на объект – абстрактное приложение Word
Dim wda As Word.Application
    '  переменная wdd ссылается на объект – документ приложения Word
Dim wdd As Word.Document

    '  если возникнет ошибка – перейдем
    '  на ее обработку по метке ErrStartWord
On Error GoTo ErrStartWord
    'открываем документ
    '  устанавливаем связь переменной wdd с нашим документом,
    '   доступ к которому получаем командой GetObject
Set wdd = GetObject("D:\Справка.doc")
    '  устанавливаем связь переменной wda с предком (Parent),
    '  который создал наш документ (т.е. с приложением Word)
Set wda = wdd.Parent
    'делаем приложение Word видимым
wda.Visible = True
    '  ищем в коллекции Bookmarks закладку Факультет
    '  и выделяем ее методом Select
wdd.Bookmarks("Факультет").Select
    '  по месту выделения Selection методом TypeText вводим данные,
    '  полученные из элемента управления Аббревиатура формы Факультет
wda.Selection.TypeText Text:=Forms!Факультет![Аббревиатура]
    '  выделяем следующие закладки, такие как,
    '  например, Специальность
wdd.Bookmarks("Специальность").Select
    '   используем конструкцию With для более короткого доступа к объекту
With wda
    '   по месту выделения вставляем данные из подформы
.Selection.TypeText  Text:=Forms!Факультет![Специальности  подчиненная
форма].Form![Название специальности]
    '  относительно выделения смещаемся методом GoTo к закладке Фамилия,
    '  указывая ее имя параметром Name
.Selection.GoTo Name:="Фамилия"
    '   и вновь имеем объект Selection, который
    '   уже указывает на новую закладку Фамилия
.Selection.TypeText     Text:=Forms!Факультет![Студенты     подчиненная
форма].Form![Фамилия]
    '   аналогично производим заполнение остальных полей
    '   активизируем документ методом Activate – теперь с ним можно работать,
    '   например, отпечатать справку на принтере…
.Activate
End With     '   закрываем конструкцию With
    '   освобождаем переменные, чтобы они ни на что не ссылались,
    '   т.е. указывали на Nothing
Set wdd = Nothing
Set wda = Nothing
Exit Sub
    '  это метка обработчика ошибок – он будет формировать сообщение командой
    '  MsgBox с указанием номера, описания и информационной картинки в окне
ErrStartWord:
MsgBox Err.Description & "   " & Err.Number, vbInformation
Exit Sub     '   выход из программы при возникновении ошибки
```

Рис. 6.16. Код программы с комментариями (файл ButtonClick_6.txt)

В программе используется также метод *Select*, который позволяет получить новый объект выделения – *Selection*, к которому применяются следующие методы:

– *GoTo* – переход к новому выделению, задаваемому параметром *Name*;

– *TypeText* с параметром *Text*, которому присваиваются цепочки, указывающие на вставляемый элемент из формы или подформы.

ВНИМАНИЕ!!!

Во-первых, не забудьте, что в коде наш файл справки расположен в корневой папке диска **D:**. Внесите необходимые корректировки, если вы используете другое место размещения документа.

Во-вторых, в редакторе VBA необходимо подключить ссылку на библиотеку объектов MS Word 10.0 или более позднюю (например, 12.0). Для этого в меню нажать пункт **Tools** – **References…** . В списке найти и установить флажок для строки **Microsoft Word 10.0 Object Library**. Результат подключения представлен на рисунке ниже.

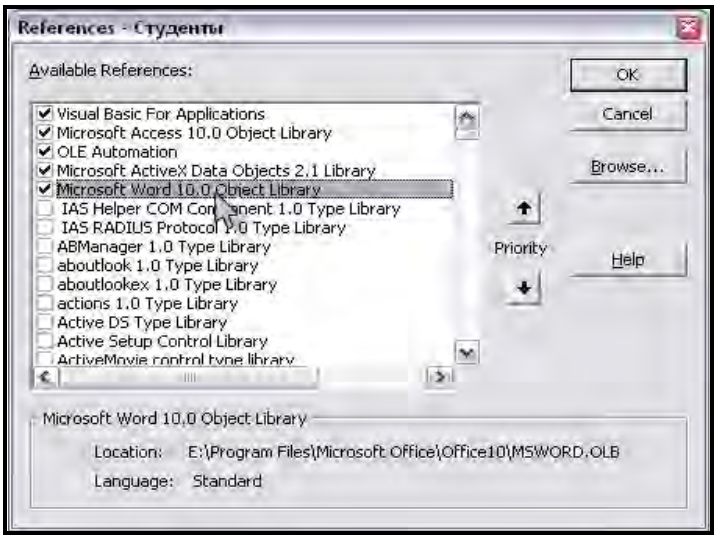


Рис. 6.17. Подключение библиотеки объектов Microsoft Word

Теперь вернемся в **Access**, перейдем в режим работы с формой и протестируем результаты автоматизации: выберем факультет, специальность и студента, для которого требуется выдать справку. Нажимаем на кнопку **Формирование справки**.

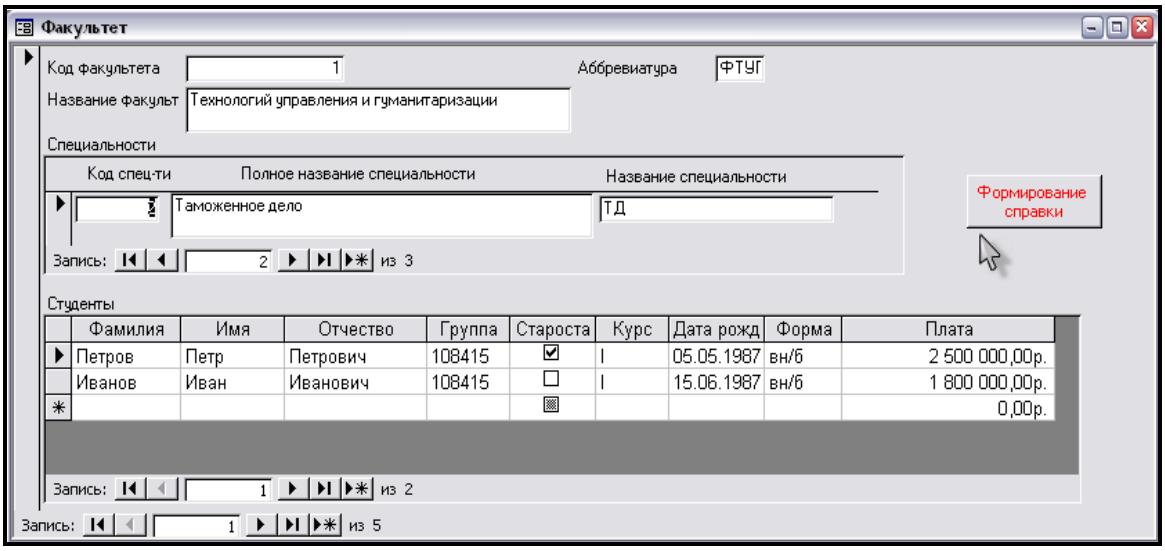


Рис. 6.18. Формирование справки для выбранного студента

Результат представлен ниже:

СПРАВКА

Студент Петров факультета ФТУГ обучается по специальности ТД в группе на курсе.

Декан факультета /_____/

Замечание. Файл **Справка.doc** можно распечатать и закрыть, не сохраняя внесенных изменений. Он будет использоваться для формирования аналогичных справок.

Самостоятельная работа

Доработать код программы-обработчика события нажатия на кнопку так, чтобы в справке отображалась информация о группе и номере курса, на котором учится студент.

Результат продемонстрировать преподавателю.

Контрольные вопросы

1. В приведенном на рис. 6.5 коде функции **SumPropis** имеются функции для работы с текстом: Len, UCase, Mid. Используя Help-справочник, изучите и опишите их назначение.
2. Какой оператор в языке VBA используется для сцепки текстовых данных и как он применяется в функции **SumPropis**?
3. Какие управляющие конструкции используются в коде стандартного модуля на рис. 6.5?
4. Какие изменения потребуется сделать, чтобы в справке печаталось полное название специальности?
5. Какое приложение в задании 6.2 выступает в роли сервера автоматизации, а какое в роли клиента автоматизации?

Тема 7. Программирование элементов управления формы

Теоретические сведения

Программирование элементов управления представляет собой процесс создания процедур обработки событий, происходящих с конкретным элементом формы или самой формой. Список событий, доступных для обработки, можно увидеть на закладке *События* в окне свойств элемента управления (рис. 7.1). Список событий зависит от вида элемента управления. Когда в процессе работы пользователя с формой происходит событие, для которого была создана процедура, то тем самым инициируется выполнение этой процедуры.

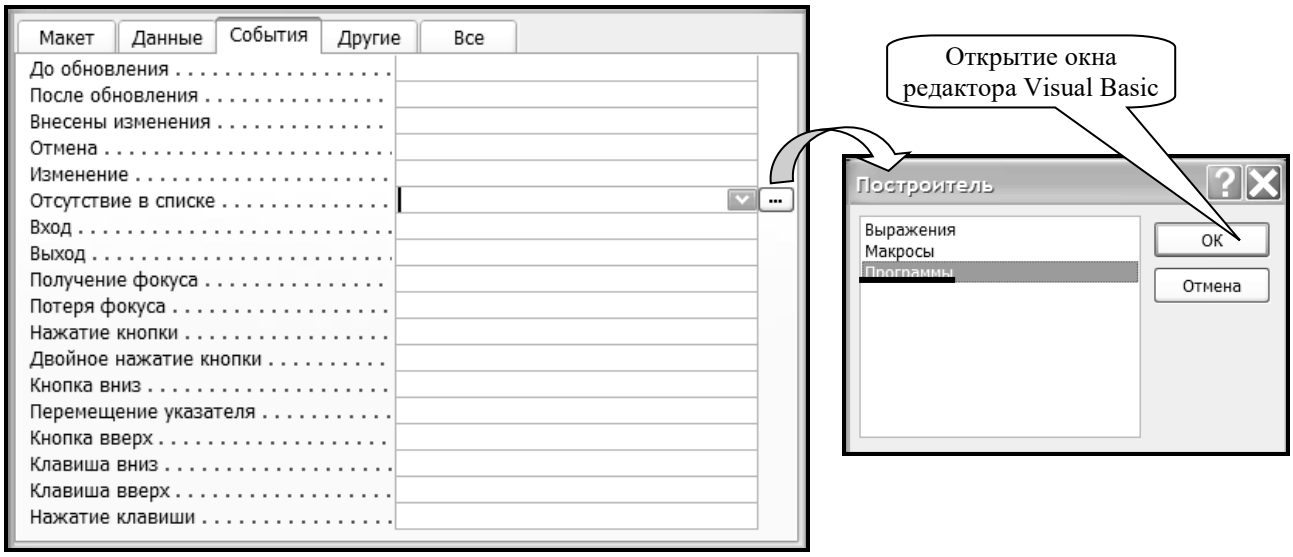


Рис. 7.1. События элемента управления ***Поле со списком***

Для создания процедуры обработки события следует поставить курсор в нужное событие, нажать многоточие, в окне построителя выбрать **Программы** и нажать **ОК** (рис. 7.1). После этого будет открыто окно редактора Visual Basic с шаблоном процедуры: первая строка `Private Sub ИмяЭлемента_ИмяСобытия()` и последняя строка `End Sub`. Между этими строками непосредственно вводится текст самой процедуры. При выполнении заданий объемные тексты процедур можно копировать из указанных файлов.

Практические задания

Задание 7.1. Запрограммировать поле со списком **Группа** в форме *Студенты* так, чтобы иметь возможность ввода номера группы при отсутствии его в списке.

Рекомендации по выполнению:

1. Создать запрос с группировкой по полю **Группа** на основе таблицы *Студенты* и преобразовать его в запрос на создание таблицы с именем *Группа*. Запрос сохранить под именем *Создание таблицы Группа* и выполнить его. В результате появится новая таблица. В режиме *Конструктора* в таблице *Группа* сделать ключевым поле **Группа**.
2. Открыть форму *Студенты* в режиме *Конструктора*. С помощью контекстного меню преобразовать поле **Группа** в поле со списком. Затем открыть его свойства и на закладке *Данные* задать:

   Источник строк → **Группа**

   Ограничится списком → **Да**
3. Перейти в режим *Формы* и проверить работоспособность сделанных изменений: для студента Андреева выбрать группу 107514.

4. Чтобы иметь возможность пополнять список групп в таблице *Группа* непосредственно в форме *Студенты*, необходимо проделать следующие действия:
   - открыть свойства поля со списком **Группа** на закладке *События*;
   - нажать кнопку **...** для события *Отсутствие в списке* (рис. 7.1) и в появившемся окне выбрать **Программы**;
   - в окне редактора Visual Basic между строк **`Private Sub`** … и **`End Sub`** (там где будет стоять курсор) ввести код (выделен жирным шрифтом).

```
Private Sub Группа_NotInList(NewData As String, Response As Integer)
 |
End Sub
```

Одинарная кавычка

CurrentDb.Execute "INSERT INTO Группа (Группа) VALUES ('" & NewData & "')"
Response = acDataErrAdded

5. Перейти в режим *Формы* и проверить работоспособность сделанных изменений.

Задание 7.2. Создать в форме *Студенты* кнопку закрытия формы без использования *Мастера*.

Рекомендации по выполнению:
1. Открыть форму *Студенты* в режиме *Конструктора*.
2. Отключить кнопку **Мастера** (если она включена) на панели элементов, выбрать элемент **Кнопка** и разместить его на форме.
3. Открыть свойства **Кнопки** на закладке *События*.
4. Щелкнуть по многоточию **...** для события *Нажатие кнопки* и в появившемся окне выбрать **Программы**.
5. В окне редактора Visual Basic между строк **`Private Sub`** … и **`End Sub`** (там где будет стоять курсор) ввести код: **`DoCmd.Close`**
6. Перейти в режим *Формы* и проверить работоспособность сделанных изменений.

Задание 7.3. Создать в форме *Студенты* элементы управления, позволяющие добавлять фотографию студента, отображать, если она уже имеется, или удалять.

Рекомендации по выполнению:
1. Для работы с фотографиями нужно скопировать несколько графических файлов **.bmp* или **.jpg* в папку, где расположена ваша база данных. Можно использовать свои фотографии или запустить поиск графических файлов на компьютере. В дальнейшем при копировании базы данных в другие папки или на съемные носители эти файлы нужно будет копировать вместе с базой данных.
2. Открыть форму *Студенты* в режиме *Конструктора* и перетащить поле **Фото** из списка полей в область данных, подпись поля удалить.
3. Открыть свойства поля **Фото** и изменить его имя на **ImagePath**, в свойстве *Вывод на экран* выбрать **Нет**.
4. Щелкнуть по многоточию **...** для события *После обновления* и в появившемся окне выбрать **Программы**.
5. В окне редактора Visual Basic между строк (там где будет стоять курсор) **`Private Sub ImagePath_AfterUpdate()`** и **`End Sub`** ввести код:

```
On Error Resume Next
   showErrorMessage
   showImageFrame
   Me![ImageFrame].Picture = Me![ImagePath]
```

Этот код позволит отобразить в элементе **ImageFrame** фотографию, имя файла которой записано в поле **ImagePath**.

6. Переключиться в режим конструктора и создать элемент **Рисунок** (рис. 7.1). Выбрать для него любой графический файл. Затем в окне свойств изменить его имя на **ImageFrame**, свойство *Рисунок* очистить, в свойстве *Установка размеров* выбрать **По размеру рамки**, в свойстве *Вывод на экран* выбрать **Нет**.
7. Создать элемент **Надпись** с поясняющим текстом: *Для добавления или удаления фотографии нажмите кнопку "Добавить фото" или "Удалить фото"*. Изменить его имя на **ErrorMsg**, а свойстве *Выравнивание текста* выбрать **По центру**.
8. Создать две кнопки: 1) с именем **AddPicture** и подписью **Добавить фото**; 2) с именем **RemovePicture** и подписью **Удалить фото**.

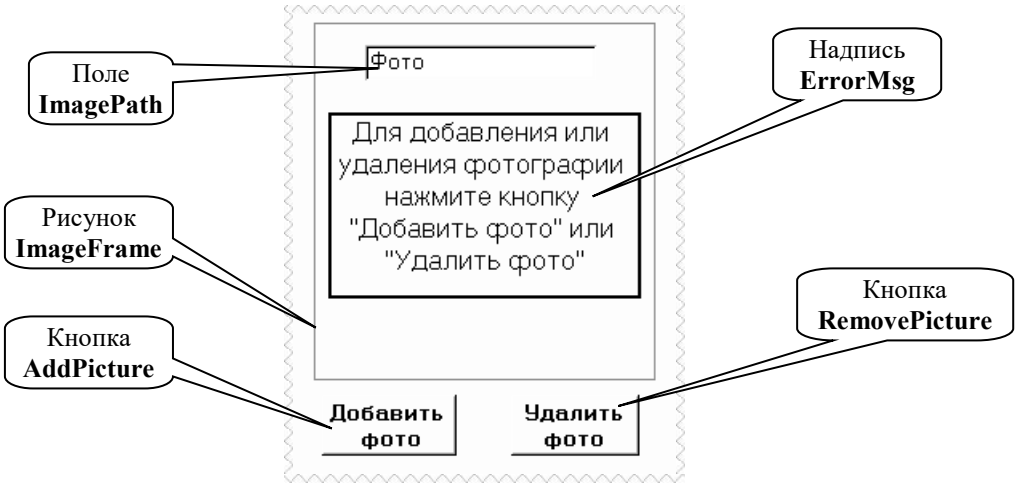


*Рис. 7.1. Фрагмент формы **Студенты** с программируемыми элементами управления*

9. Для кнопки **AddPicture** в процедуру обработки события *Нажатие кнопки* ввести код: **`getFileName`** (см. пп. 3–5 задания 7.2). Этот код является именем процедуры, текст и пояснение которой будут приведены ниже (см. п. 14).
10. Аналогично для кнопки **RemovePicture** ввести:

```
    Me![ImagePath] = ""
    hideImageFrame
    ErrorMsg.Visible = True
```

Таким образом по нажатию кнопки будет очищаться поле **ImagePath**, скрываться рамка фотографии и выводиться сообщение в надписи **ErrorMsg**.

11. Открыть окно свойств для формы *Студенты* на закладке *События*. Щелкнуть по многоточию **...** для события *После обновления* и в появившемся окне выбрать **Программы**. В окне редактора Visual Basic между строк (там где будет стоять курсор) **`Private Sub Form_AfterUpdate()`** и **`End Sub`** ввести тот же код (можно скопировать) что и для **ImagePath** в п. 5.
12. Перейти в режим конструктора, щелкнуть по многоточию **...** для события *Текущая запись* и в появившемся окне выбрать **Программы**. В окне редактора Visual Basic между строк (там где будет стоять курсор) **`Private Sub Form_Current()`** и **`End Sub`** ввести код (можно скопировать из файла **FormCurrent_7.txt**):

```
Dim path As String
  path = CurrentProject.path
  On Error Resume Next
    ErrorMsg.Visible = False
    If Not IsNull(Me![Фото]) Then
      Me![ImageFrame].Picture = path & "\" & Me![ImagePath]
```

```
      showImageFrame
      Me.PaintPalette = Me![ImageFrame].ObjectPalette
      If (Me![ImageFrame].Picture <> path & "\" & Me![ImagePath]) Then
         hideImageFrame
         ErrorMsg.Caption = "Фотография не найдена"
         ErrorMsg.Visible = True
       End If
    Else
      hideImageFrame
      ErrorMsg.Caption = "Для добавления или удаления фотографии _
          нажмите кнопку ""Добавить фото"" или ""Удалить фото"""
      ErrorMsg.Visible = True
    End If
```

Данная процедура позволяет отображать в форме фотографию, если она имеется для записи текущего сотрудника, Если указанный файл не существует, либо если для текущего сотрудника поле **Фото** пусто, надпись **ErrorMsg** выводит соответствующее сообщение.

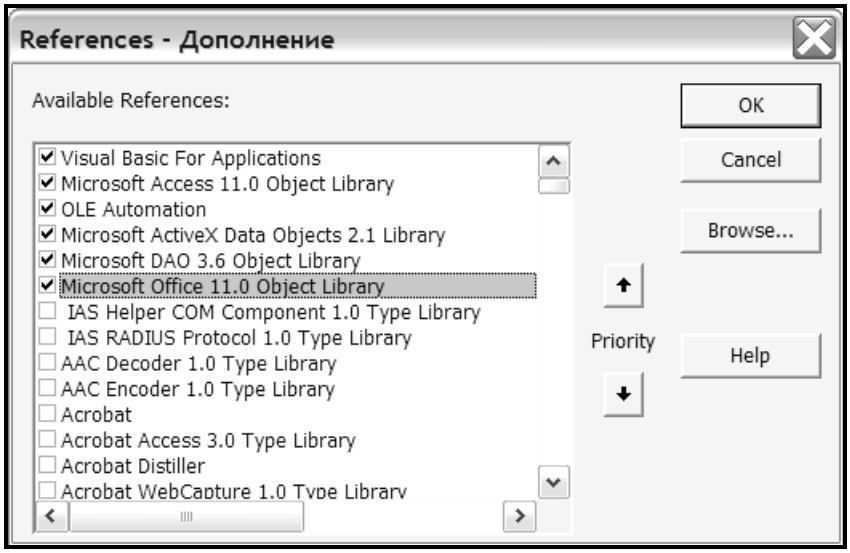


Рис. 7.2. Диалог подключения дополнительных библиотек

13. В окне редактора Visual Basic выполнить команду **Tools—References** и подключить дополнительные библиотеки, как показано на рис. 7.2.
14. Скопировать тексты вспомогательных процедур из файла **Procedurs_7.txt** и вставить их после всех процедур, созданных для формы *Студенты*.

Содержимое файла **Procedurs_7.txt**:

```
Sub getFileName()
  ' Процедура выбора имени файла с фотографией текущего студента
  ' с помощью стандартного окна открытия файла Office.
  ' Если пользователь указывает файл, его содержимое
  ' отображается в элементе управления ImageFrame.
  Dim fileName As String
  Dim result As Integer
  With Application.FileDialog(msoFileDialogFilePicker)
    .Title = "Выбор фотографии студента"
    .Filters.Add "Все файлы", "*.*"
    .Filters.Add "JPEG", "*.jpg"
    .Filters.Add "Рисунки", "*.bmp"
    .FilterIndex = 1
    .AllowMultiSelect = False
```

```
    .InitialFileName = CurrentProject.path
    result = .Show
    If (result <> 0) Then
       fileName = Trim(.SelectedItems.Item(1))
       Me![ImagePath].Visible = True
       Me![ImagePath].SetFocus
       Me![ImagePath].Text = StrReverse(Left(StrReverse(fileName), _
       InStr(1, StrReverse(fileName), "\") - 1))
       Me![Имя].SetFocus
       Me![ImagePath].Visible = False
    End If
  End With
End Sub

Sub showErrorMessage()
    ' Выводит сообщение errormsg, если файл фотографии недоступен.
    If Not IsNull(Me![Фото]) Then
        ErrorMsg.Visible = False
    Else
        ErrorMsg.Visible = True
    End If
End Sub

Sub hideImageFrame()
    ' Скрывает элемент управления с фотографией.
    Me![ImageFrame].Visible = False
End Sub

Sub showImageFrame()
    ' Выводит элемент управления с фотографией.
    Me![ImageFrame].Visible = True
End Sub
```

15. Форму сохранить, перейти в режим *Формы* и проверить работоспособность сделанных изменений.

Задание 7.4. Создать форму для работы с таблицей *Успеваемость* и возможностью ввода названия предмета при отсутствии его в списке предметов.

Рекомендации по выполнению:

1. Создать форму для ввода и просмотра данных таблицы *Успеваемость*:
   - на закладке *Формы* выбрать строку *Создание формы с помощью мастера*;
   - выбрать поля **Фамилия**, **Имя**, **Отчество**, **Группа** из таблицы *Студенты* и поля **Предмет**, **Оценка** из таблицы *Успеваемость*;
   - выбрать вид представления данных по таблице *Студенты*;
   - выбрать внешний вид подчиненной формы — **Ленточный**;
   - выбрать стиль — **Стандартный**;
   - задать имена форм:

     Форма: *Оценки студентов*

     Подчиненная форма: *Успеваемость подчиненная форма*
   - открыть форму *Успеваемость подчиненная форма* в режиме Конструктора;
   - открыть свойства формы на закладке **Данные**;
   - нажать кнопку ... для строки **Источник записей**;

- выбрать в бланке запроса поля в следующем порядке: **Оценка, Предмет, Фамилия, Имя, Отчество**;
- для поля **Оценка** задать сортировку **по возрастанию**;
- закрыть построитель запросов и форму, сохранив изменения.

2. Преобразовать поле **Предмет** в форме *Успеваемость подчиненная форма* в поле со списком.

Самостоятельная работа

Запрограммировать поле со списком **Предмет** в форме *Успеваемость подчиненная форма* по аналогии с полем **Группа** в задании 7.1.

Контрольные вопросы

1. Каким образом осуществляется программирование элементов управления формы?
2. Когда происходит выполнение процедуры **`AddPicture_Click( )`**, созданной в п. 9 задания 7.3? Что делает эта процедура?
3. Когда происходит выполнение процедуры **`Form_Current( )`**, созданной в п. 12 задания 7.3? Что делает эта процедура?
4. Как подключаются дополнительные библиотеки стандартных процедур?

Тема 8. Конструирование макросов

Теоретические сведения

Макрос — это последовательность макрокоманд, т.е. инструкций, ориентированных на выполнение определенных действий. Макросы являются надстройкой над языком Visual Basic и обеспечивают пользователя средствами для решения нестандартных задач без знаний детального программирования. В Access есть возможность обеспечить взаимодействие макросов с объектами на основе **событий**. Таким образом, пользователь, выполняя определенные действия в формах, инициирует запуск макросов, автоматизирующих выполнение различных подзадач. Например, при закрытии формы возникает событие *Закрытие*, при щелчке мышью по кнопке, имеющейся в форме, для нее возникает событие *Нажатие кнопки*.

Формирование макроса осуществляется в режиме *Конструктора* путем последовательного выбора макрокоманд, в том порядке, в котором они должны выполняться. Для каждой макрокоманды задаются значения аргументов (рис. 8.1).

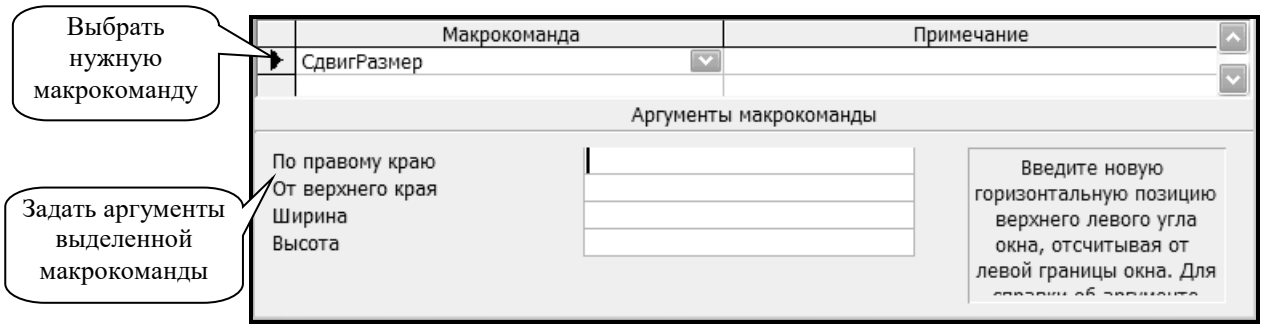


Рис. 8.1. Конструирование макроса

Макрокоманды делятся на классы, сформированные по функциональному принципу:

- открытие/закрытие таблиц, запросов, форм и отчетов, печать данных,
- проверка истинности условий и управление выполнением макрокоманд,
- установка значений,
- поиск данных,
- построение пользовательского меню и выполнение команд меню,
- управление выводом на экран и фокусом,
- сообщение пользователю о выполняемых действиях,
- переименование, копирование, удаление, импорт и экспорт объектов,
- запуск приложений.

Для выполнения макроса используется кнопка ***Запуск.*** Чтобы связать созданный макрос с каким-либо событием, нужно открыть свойства объекта или элемента управления на закладке *События* и затем выбрать имя макроса в строке этого события. Набор событий с которыми можно связать макрос или какую-либо процедуру обработки на языке VBA зависит от конкретного объекта.

Группа макросов создаётся как один макрос, в котором содержатся несколько макросов, например, связанных с решением одной задачи или используемых при работе с одной формой. Работать с группой удобнее, чем с несколькими отдельными макросами.

Для создания группы макросов нужно включить столбец *Имя макроса* с помощью кнопки  на панели инструментов или команды **Вид** — **Имена макросов**. В ячейку столбца *Имя макроса* надо ввести имя первого макроса, входящего в группу. Затем записать макрокоманды, выполняемые в первом макросе. Аналогичным образом надо ввести имена других макросов и их макрокоманды. Все макросы, созданные в одном

окне, будут составлять одну группу. Имя, указанное при сохранении такой группы макросов, будет именем группы, которое выводится в списке макросов в окне БД.

Для ссылок на макросы, которые вошли в группу макросов, используется следующий синтаксис: *Имя_Группы_Макросов. Имя_Макроса*

Использование условий в макросе позволяет задать порядок передачи управления между макрокомандами и обеспечивает выполнение определенных ветвей алгоритма. Например, если в макросе проверяется значение поля в форме на соответствие заданным условиям, то для одних значений может потребоваться вывести сообщение, а для других значений сформировать отчет.

Условие вводится в столбец *Условие* в строку макрокоманды, которая должна выполняться если это условие истино. Столбец *Условие* включается нажатием кнопки  на панели инструментов или командой **Вид** — **Условия**. Условие задается с помощью логического выражения.

Для организации ветвлений в программе нужно наряду с условиями использовать макрокоманды *Остановить Макрос* и *Запуск Макроса*. Возможно также создание циклов.

Пример 8.1. Создать макрос, который будет подсчитывать количество студентов в каждой группе и сохранять эти данные в таблице *Количество_студентов*.

Выполнение:

1. Создать новый макрос.
2. Выбрать макрокоманду **УстановитьСообщения** и в аргументе *Включить сообщения* выбрать **Нет**, для того чтобы отключить системные сообщения и избежать остановок в работе макроса.
3. Выбрать макрокоманду **ЗапускЗапросаSQL** и в аргументе *Инструкция SQL* ввести следующую инструкцию:

```
SELECT Группа, Count(Фамилия) AS Кол_студ INTO Количество_студентов
FROM Студенты GROUP BY Группа;
```

Правила написания инструкций SQL в аргументе макрокоманды такие же, как и в режиме SQL при создании запросов (см. тему 5), но в этом случае вся инструкция вводится в одну строчку.

4. Выбрать макрокоманду **ОткрытьТаблицу** и в аргументе *Имя таблицы* ввести **Количество_студентов**.
5. Макрос сохранить и выполнить.

Практические задания

Задание 8.1. Создать макрос «*Число студентов*», который будет: 1) создавать таблицу *Количество_студентов*, содержащую данные о количестве студентов в каждой группе; 2) добавлять поле **Кол_во** в таблицу *Группа*; 3) обновлять поле **Кол_во** данными таблицы *Количество_студентов*; 4) открывать таблицу *Группа*; 5) удалять таблицу *Количество_студентов*.

Рекомендации по выполнению:

- Создать новый макрос.
- С помощью макрокоманды **УстановитьСообщения** отключить предупреждающие сообщения.
- Выбрать макрокоманду **ЗапускЗапросаSQL** и в аргументе *Инструкция SQL* ввести инструкцию, приведенную в *п. 3 примера 8.1*.
- Выбрать макрокоманду **ЗапускЗапросаSQL** и в аргументе *Инструкция SQL* ввести инструкцию, которая позволит добавить поле **Кол_во** в таблицу *Группа* (см. тему 5).

- Выбрать макрокоманду **ЗапускЗапросаSQL** и в аргументе *Инструкция SQL* ввести инструкцию, которая позволит обновить поле **Кол_во** в таблице *Группа* данными поля **Кол_студ** таблицы *Количество_студентов*.
- С помощью макрокоманды **ОткрытьТаблицу** открыть таблицу *Группа*.
- С помощью макрокоманды **УдалитьОбъект** удалить таблицу *Количество_студентов*.
- Макрос сохранить под именем *Число студентов* и выполнить.

Задание 8.2. Создать макросы для выполнения нескольких запросов. Для размещения запросов на экране так, чтобы они не перекрывали друг друга, используйте макрокоманду **СдвигРазмер**.

Макрос	*Выполняемые запросы*
Факультеты и форма обучения	2-2-1 и 2-2-2
Сумма оплаты	2-2-3 и 2-3-1
Задолженности	2-3-3 и 5-3-1
Итоги сессии	2-3-2, 5-8-3, 5-15, 5-16
Количество по курсам	2-2-4 и 5-8-2
Доля внебюджетников	2-2-6, 2-2-7, 3-4

Задание 8.3. Создать макрос для вывода в форме *Успеваемость подчиненная форма* текста **Двоечник!** рядом с надписью *Оценка*, если в текущей записи в поле ОЦЕНКА стоит 2 или 3.

Рекомендации по выполнению:

- Создать новый макрос и в окне конструктора включить столбец **Условие**.
- Задать порядок и аргументы макрокоманд, руководствуясь таблицей:

Условие	**Макрокоманда**	**Примечание**	**Имя аргумента**	**Значение аргумента**
[ОЦЕНКА]<=3	ЗадатьЗначение	Вывести на экран надпись «Двоечник!»	Элемент	[Надпись30].[Visible]
			Выражение	Да
[ОЦЕНКА]>3	ЗадатьЗначение	Убрать с экрана надпись «Двоечник!»	Элемент	[Надпись30].[Visible]
			Выражение	Нет

- Сохранить и закрыть макрос.
- Открыть форму *Успеваемость подчиненная форма* в режиме конструктора. В области заголовка формы, рядом с надписью «Оценка» создать надпись «Двоечник!» (цвет шрифта — красный, размер шрифта — 14, имя надписи — **Надпись30**).
- Открыть **Свойства формы**. На закладке **События** в строке **Текущая запись** выбрать имя созданного макроса. Сохранить и закрыть форму.
- Открыть форму *Оценки студентов* и проверить работу макроса.

Контрольные вопросы

1. Что такое макрос, группа макросов?
2. Что представляет собой макрокоманда? Как задаются аргументы макрокоманды?
3. На какие классы делятся макрокоманды?
4. Что такое событие? Как ассоциировать макрос с событием?
5. Каким образом и для какой цели в макрос добавляются условия?
6. Каков порядок создания макроса в Access?

Тема 9. Разработка кнопочной формы

Теоретические сведения

Кнопочная форма представляет собой форму, с помощью которой можно открывать другие формы или отчеты базы данных, а также выполнять макросы. Кнопочная форма создается с целью сгруппировать объекты базы данных по функциональному назначению, обеспечить удобный графический интерфейс и возможность ориентироваться среди множества разрабатываемых объектов.

Создание и корректировка кнопочной формы производится с помощью надстройки **Диспетчер кнопочных форм**, которая открывается через меню **Сервис—Служебные программы** (в Access_2007 кнопка **Диспетчер кнопочных форм** на закладке **Работа с базами данных**). Если в базе данных еще нет кнопочной формы, то при выполнении этой команды появится сообщение «*Не удается найти кнопочную форму в этой базе данных. Создать кнопочную форму?*», на которое нужно ответить **Да**. После этого будет открыт диалог (рис. 9.1).

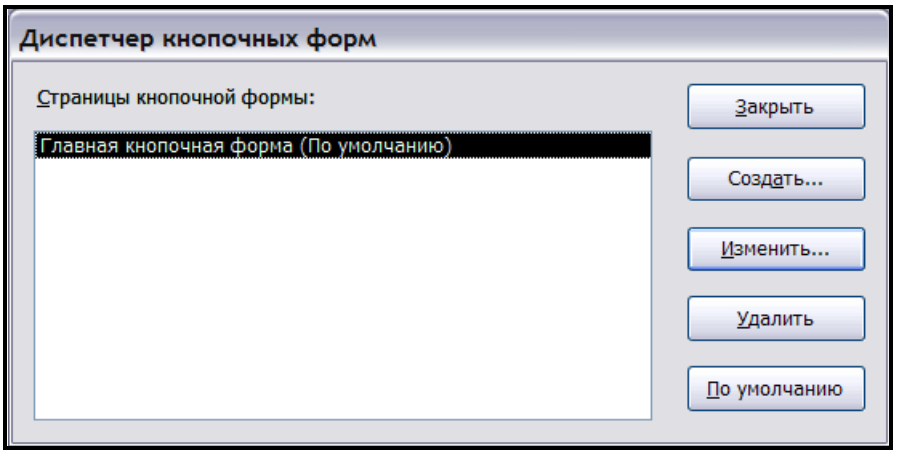


Рис. 9.1. Диспетчер кнопочных форм

В диалоге **Диспетчер кнопочных форм** (рис. 9.1) формируются страницы кнопочной формы разных уровней, одна из которых является главной и открывается по умолчанию при открытии кнопочной формы. Изменение страницы, открываемой по умолчанию, производится кнопкой **По умолчанию** данного диалога. Кнопка **Создать** позволяет создавать новые страницы кнопочной формы и задавать им имена. Кнопка **Изменить** вызывает диалог **Изменение страницы кнопочной формы** (рис. 9.2), который позволяет создавать элементы выделенной кнопочной формы и корректировать ее название.

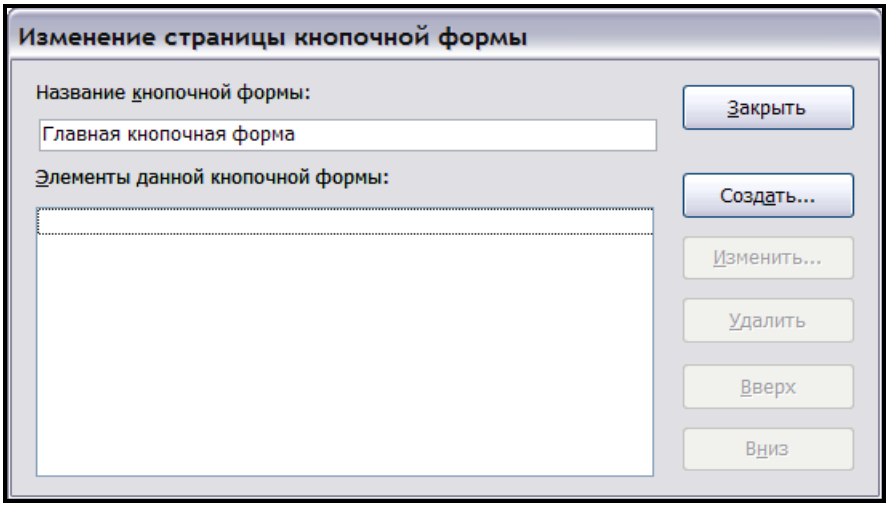


Рис. 9.2. Изменение страницы кнопочной формы

Для этого в диалоге **Изменение элемента кнопочной формы** (рис. 9.3), который открывается кнопкой **Создать** или **Изменить**, задается текст подписи к элементу, команда, выполняемая по нажатию кнопки, связанной с этим элементом, и выбирается имя объекта, который должен быть открыт. На каждой странице может быть создано не более восьми элементов.

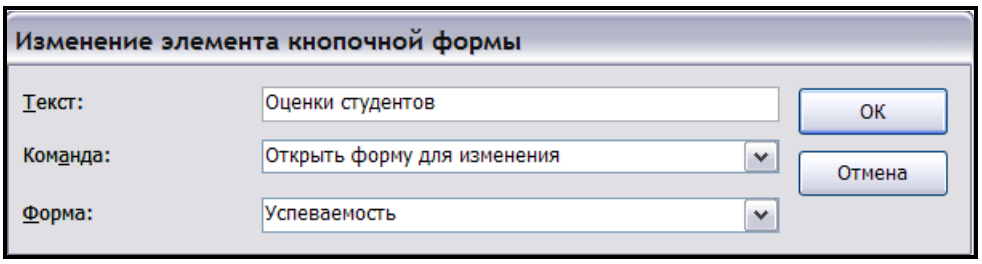


Рис. 9.3. Изменение элемента кнопочной формы

После создания всех страниц кнопочной формы и элементов на каждой странице диспетчер закрывают, а в базе данных будет создана кнопочная форма и таблица Switchboard Items, которая описывает иерархические уровни страниц формы, а также подписи и действия кнопок формы. Кроме того в процессе работы диспетчера кнопочных форм Access создает необходимые процедуры обработки событий, которые обеспечивают работу кнопочной формы.

Кнопочную форму можно открыть в режиме конструктора и изменить фон, добавить рисунки, подписи. Ни в коем случае нельзя изменять или удалять кнопки и надписи рядом с ними, так как после этого форма не будет работать.

Для работы с базой данных удобно, когда кнопочная форма автоматически открывается при открытии базы данных. Чтобы включить эту возможность, нужно в диалоге **Сервис—Параметры запуска** в списке *Вывод формы/страницы* выбрать **Кнопочная форма**. В Access 2007 нужно открыть **Параметры Access**, выбрать категорию **Текущая база данных**, раскрыть список *Форма просмотра* и выбрать **Кнопочная форма**.

Практические задания

Задание 9.1. Открыть диспетчер кнопочных форм и создать четыре страницы кнопочной формы:

БД «Студенты» (По умолчанию)
Ввод и просмотр данных
Отчетность
Справочная информация

Задание 9.2. В созданных страницах создать и настроить следующие элементы:

Кнопочная форма	Текст элемента	Команда	Объект (**выбрать имя объекта из списка**)
БД «Студенты» (По умолчанию)	Ввод и просмотр данных	Перейти к кнопочной форме	Ввод и просмотр данных
	Отчетность	Перейти к кнопочной форме	Отчетность
	Справочная информация	Перейти к кнопочной форме	Справочная информация
	Выход	Выйти из приложения	—
Ввод и просмотр данных	Студенты	Открыть форму для изменения	Студенты
	Факультет	Открыть форму для изменения	Факультет

Окончание таблицы

Кнопочная форма	Текст элемента	Команда	Объект (**выбрать имя объекта из списка**)
	Оценки студентов	Открыть форму для изменения	Оценки студентов
	Назад	Перейти к кнопочной форме	БД «Студенты»
Отчетность	Студенты	Открыть отчет	Студенты
	Внебюджетники	Открыть отчет	Отчет из п. 4.1
	Стипендия	Открыть отчет	Отчет из п. 4.2
	Итоги сессии	Открыть отчет	Отчет из п. 4.3
	Назад	Перейти к кнопочной форме	БД «Студенты»
Справочная информация	Факультеты и форма обучения	Выполнить макрос	Факультеты и форма обучения
	Сумма оплаты	Выполнить макрос	Сумма оплаты
	Задолженности	Выполнить макрос	Задолженности
	Итоги сессии	Выполнить макрос	Итоги сессии
	Количество по курсам	Выполнить макрос	Количество по курсам
	Доля внебюджетников	Выполнить макрос	Доля внебюджетников
	Число студентов	Выполнить макрос	Число студентов
	Назад	Перейти к кнопочной форме	БД «Студенты»

Задание 9.3. Закрыть **Диспетчер кнопочных форм**, перейти на закладку **Формы**, открыть кнопочную форму и проверить ее работу.

Задание 9.4. В режиме конструктора добавить в кнопочную форму рисунок, надписи, изменить фон, шрифт по своему усмотрению.

Задание 9.5. Установить кнопочную форму в качестве автоматически запускаемой при открытии базы данных.

Контрольные вопросы

1. Каково назначение кнопочной формы?
2. Перечислите основные этапы создания кнопочной формы.
3. Какие действия могут выполняться с помощью кнопок элементов кнопочной формы?
4. Как настроить автоматический запуск кнопочной формы при открытии базы данных?
5. Какие действия допускаются и какие не допускаются в режиме конструктора кнопочной формы?
6. Для чего нужна таблица Switchboard Items и как она создается?

Тема 10. Проектирование базы данных

Теоретические сведения

Проектирование базы данных представляет собой процесс отображения исследуемых явлений реального мира, называемых **предметной областью**, в виде данных в памяти компьютера.

Основная цель проектирования базы данных — это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте.

Основные этапы процесса проектирования:

1. **Концептуальное (инфологическое) проектирование** включает сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:

- обследование предметной области, изучение ее информационной структуры;
- выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами, связями между ними, действиями над информационными объектами;
- моделирование и интеграция всех представлений.

По окончании данного этапа формируется семантическая модель предметной области, не зависящая от каких-либо физических условий реализации. Часто она представляется в виде ER-диаграммы. В ER-диаграммах информационные объекты (сущности) изображаются прямоугольниками, ассоциации (связи) — ромбами или шестиугольниками, атрибуты — овалами. Сущности соединяются линиями, над которыми могут проставляться степени связи (1 или буква M) и необходимые пояснения.

2. **Логическое проектирование** представляет собой этап, в результате которого концептуальная модель предметной области претерпевает изменения с учетом выбранной модели данных. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

Модель данных — это некоторая абстракция, в которой отражаются самые важные аспекты функционирования выделенной предметной области, а второстепенные — игнорируются. Модель данных включает в себя набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные. В модели данных различают три главные составляющие:

- структурную часть, определяющую правила порождения допустимых для данной СУБД видов структур данных;
- управляющую часть, определяющую возможные операции над такими структурами;
- классы ограничений целостности данных, которые могут быть реализованы средствами этой системы.

В настоящее время описано много разнообразных моделей, построение которых преследует разные цели. В большинстве коммерческих СУБД используются ставшие классическими три вида моделей, которые различаются способами представления взаимосвязей между объектами:

- сетевая модель;
- иерархическая модель;
- реляционная модель.

3. **Физическое проектирование** рассматривает вопросы физической реализации полученной логической модели посредством выбранной СУБД. На этом этапе происходит определение структур хранения данных, методов доступа к данным, обеспечивающих оптимальную производительность, разработка средств защиты базы данных.

Реляционная модель данных

Реляционные базы данных на сегодняшний момент занимают доминирующее положение в вопросах обработки данных в информационных системах.

Основной принцип реляционной модели — получение из таблицы необходимых отношений и формирование новых. На основе первичной таблицы при помощи логических операций формируется новая таблица соответствующей структуры. Каждая таблица соответствует какому-нибудь понятию из предметной области. При проектировании реляционных баз данных используется определенная терминология (табл. 10.1).

Таблица 10.1 – Терминология реляционных баз данных

Сущность	Реальный или абстрактный объект информационной системы.
Отношение (реляция)	Некоторая регулярная структура, представленная в виде двумерной таблицы и состоящая из конечного набора однотипных строк. Содержит сведения о множестве экземпляров одной сущности.
Атрибут	Поименованный столбец отношения (поле), характеризующий отдельное свойство (реквизит) объекта.
Домен	Множество допустимых значений атрибута.
Степень отношения (арность, ранг)	Количество атрибутов реляции.
Заголовок (схема) отношения	Конечное множество имен атрибутов.
Кортеж	Строка реляции (запись). Содержит описание отдельного экземпляра данной сущности.
Кардинальное число отношения (мощность)	Количество кортежей.
Ключ	Минимальный набор атрибутов, однозначно определяющий каждый кортеж реляции.
Тело отношения	Множество кортежей. Отражает состояние сущности, поэтому постоянно меняется во времени.

Выделяют два подхода к проектированию реляционных баз данных. Первый является более традиционным и предполагает создание непосредственно реляционной схемы БД уже на этапе концептуального проектирования. В этом случае проектирование заключается в нормализации определений реляционных отношений. Нормализация представляет собой вариант восходящего подхода к проектированию, который заключается в постепенном выявлении зависимостей между атрибутами и устранении нежелательных из них. Этот подход используется, когда требуется простая схема базы данных. Второй подход основан на создании концептуальной модели данных, которая затем механически преобразуется в реляционную. Процесс преобразования автоматически гарантирует получение нормализованной реляционной модели. Подход относится к категории нисходящих, так как начинается с выявления важных для предметной области объектов и связей. К полученной схеме в упрощенном виде также применяется процесс нормализации. Это необходимо для проверки ее корректности, но этот процесс уже не приводит к перестройке отношений. Данный подход применяется при проектировании сложных корпоративных информационных систем.

Исходными данными для проектирования реляционной базы данных могут служить бланки и другие виды документов, с которыми работают конечные пользователи, а также описание информационных объектов и их атрибутов, полученное в ходе изучения предметной области.

Рассмотрим процесс проектирования на примере.

Пример 10.1. При записи в библиотеку на каждого читателя заводится формуляр (рис. 10.1), в который заносятся сведения о читателе, а затем регистрируются взятые им книги. В свою очередь каждое издание в библиотеке имеет свою учетную карточку (рис. 10.2), в которую заносится информация о данной книге.

№ читательского билета _____ *Сведения о читателе:* Фамилия _____ Имя _____ Отчество _____ Адрес _____ Телефон _____ Дата записи в библиотеку _____	*Дата получения книги*	*Дата возврата*	*Номер книги*	*Автор*	*Название книги*

Рис. 10.1. Читательский формуляр

УДК 681.3.06(075.8)

ББК 32.973.26я7

М20

Малыхина М.П.

Базы данных: основы, проектирование, использование. – СПб.: БХВ-Петербург, 2004. – 512 с.: ил.

Рис. 10.2. Учетная карточка книги

Спроектировать базу данных «*Библиотека*», основанную на записях в читательских формулярах и учетных карточках книг.

Выполнение:

1. Для проектирования базы данных «*Библиотека*» кроме ознакомления с представленными документами было бы полезно дополнительно изучить предметную область, в частности назначение первых трех строк учетной карточки книги (рис. 10.2):

Шифр хранения конкретного издания в библиотеке состоит из трех классификационных индексов: УДК, ББК и авторского знака. УДК (Универсальный Десятичный Классификатор) разработан для индексирования научно-технических и научных публикаций. ББК (Библиотечно-Библиографический Классификатор) используется в традиционных библиотеках для систематизации фондов. Оба классификатора организованы как системы "деревьев" (от общего к частному). Их структура

соответствует иерархии, принятой для упорядочивания наук и их понятийного аппарата. Каждая рубрика ББК и УДК состоит из цифрового индекса и названия рубрики. К рубрикам "привязаны" издания, имеющиеся в библиотеке. Авторский знак используется для кодирования имени автора книги (первого автора книги, созданной не более тремя авторами) или первого слова заглавия книги (если авторов больше трех или издание опубликовано под заглавием). С помощью авторского знака в библиотеках осуществляется расстановка изданий по алфавиту имен авторов или заглавий книг.

Пользователи разработанной базы данных должны иметь возможность поиска книг не только по названию или автору, но и с помощью стандартных классификаторов ББК и УДК. Кроме того, книги в библиотеке имеются, как правило, в нескольких экземплярах, причем каждый экземпляр книги будет иметь свой уникальный номер, но одну и ту же учетную карточку. Эту ситуацию следует учесть при проектировании.

2. После ознакомления с представленными документами и более детального изучения предметной области можно выделить два информационных объекта *ЧИТАТЕЛЬ* и *КНИГА* (рис. 10.3, а), которые будут находиться в отношении *многие-ко-многим*, т.е. читатель может взять несколько книг, а одна и та же книга может быть взята читателями неоднократно. Таким образом, между этими объектами можно определить связь *БЕРЁТ*, составленную из множества пар, в каждой из которых имеется читатель и взятая им книга. Полученная структура также является информационным объектом, позволяющим вести учет взятых книг (рис. 10.3, б). При этом читатель берет конкретный экземпляр книги, имеющий свой уникальный номер, поэтому нет необходимости вносить полную информацию о книге каждый раз, когда ее взяли, будет достаточно указать ее номер. В то же время полная информация о книге имеется в учетной карточке, и нет необходимости повторять эти сведения для каждого экземпляра книги. Исходя из этих рассуждений, из объекта *КНИГА* можно выделить еще один объект — ЭКЗЕМПЛЯР. Окончательный вариант ER-диаграммы представлен на рис. 10, в.

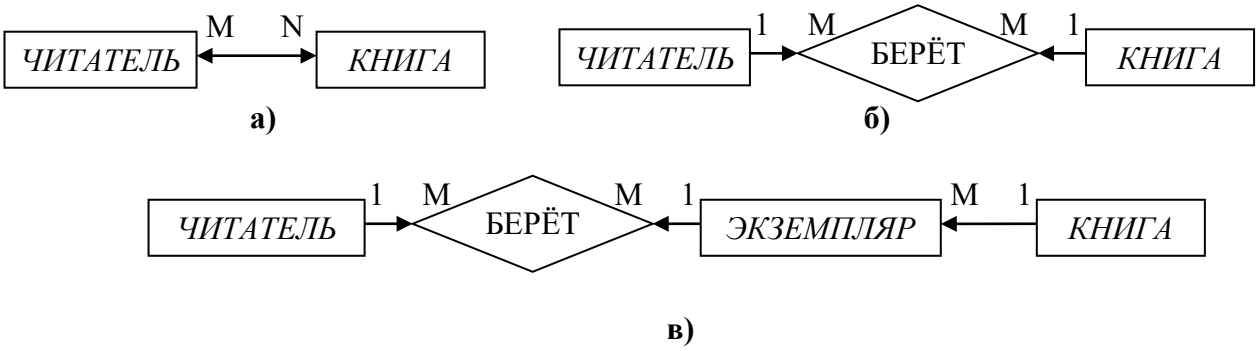


Рис. 10.3. Построение ER-диаграммы

3. Каждый информационный объект характеризуется набором атрибутов, которые можно определить с помощью документов (рис. 10.1 и 10.2). Один или несколько атрибутов будут являться первичным ключом, однозначно характеризующим конкретный экземпляр объекта. Для связи объектов должны быть предусмотрены поля связи. При этом разработанный проект должен обеспечивать поддержку целостности данных (см. тему 1). В таблице 10.2 приведены атрибуты информационных объектов проектируемой базы данных с указанием типа данных. При реализации проекта в СУБД Access каждому объекту будет соответствовать отдельная таблица, а каждому атрибуту — поле.

Таблица 10.2 – Информационные объекты и их атрибуты

Имя объекта	Описание объекта	Имя атрибута	Признак ключа	Тип данных
Читатель	Сведение о читателях, взятые из формуляра	**Номер_билета**	Ключ	Числовой
		Фамилия		Текстовый
		Имя		Текстовый
		Отчество		Текстовый
		Адрес		Текстовый
		Телефон		Текстовый
		Дата_записи		Дата/Время
Книга	Сведения о книгах, взятые из учетной карточки (дополнены ключевым атрибутом *Номер_книги* и атрибутами *Вид_издания* и *Цена*)	**Номер_книги**	Ключ	Числовой
		УДК		Текстовый
		ББК		Текстовый
		Авторский_знак		Текстовый
		Автор		Текстовый
		Название		Текстовый
		Вид_издания		Текстовый
		Год_ издания		Числовой
		Город_ издания		Текстовый
		Издательство		Текстовый
		Кол-во_стр		Числовой
		Цена		Числовой
Экземпляр	Сведения об экземплярах книги и месте их хранения в библиотеке	**Номер_книги**	Поле связи, входит в составной ключ	Числовой
		Код_экземпляра	Входит в составной ключ	Числовой
		Место_хранения		Текстовый
Выданные книги	Учет книг, выданных читателю (соответствует объекту *БЕРЁТ* ER-диаграммы)	**Номер_билета**	Поле связи, входит в составной ключ	Числовой
		Номер_книги	Поле связи, входит в составной ключ	Числовой
		Код_экземпляра	Поле связи, входит в составной ключ	Числовой
		Дата_получения	Входит в составной ключ	Дата/Время
		Дата_возврата		Дата/Время

Более наглядно взаимосвязь объектов можно представить в виде информационно-логической модели (рис. 10.4). При реализации проекта в СУБД Access схема данных должна соответствовать информационно-логической модели проекта.

Замечание. Проектирование баз данных, хотя и подчиняется определенным правилам и требованиям, но представляет собой творческий процесс, не имеющий какого-то заранее известного единственного решения. Следует также отметить, что данный пример демонстрирует проектирование по упрощенной схеме, без учета всех нюансов, которые могут возникнуть в реальной ситуации.

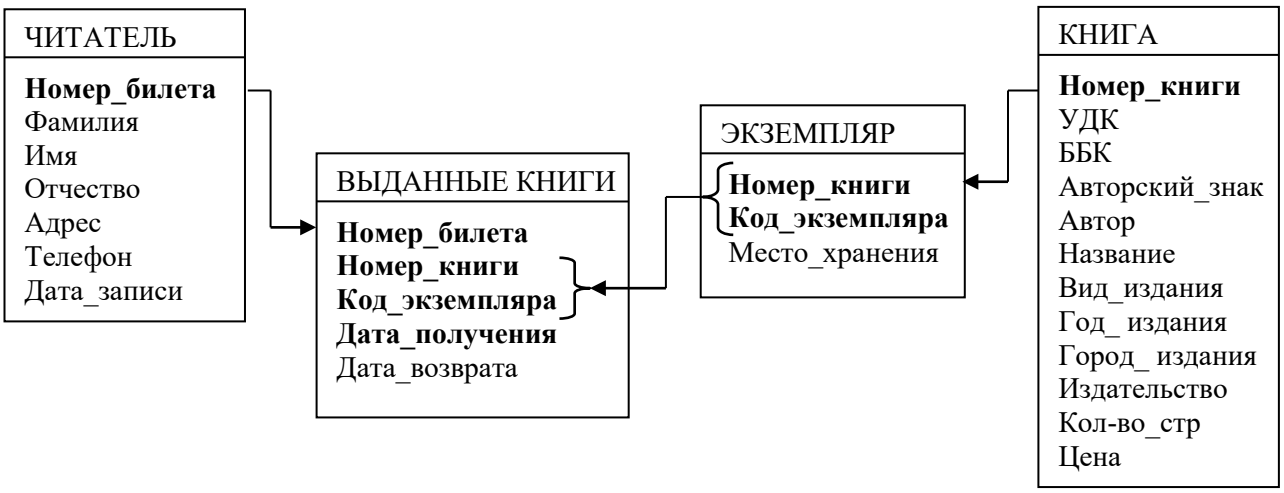


Рис. 10.4. Информационно-логическая модель (ИЛМ) базы данных «Библиотека»

Практические задания

При выполнении практических заданий по данной теме рекомендуется ознакомиться заданиями соответствующего варианта темы 11, чтобы получить более полное представление о задачах, которые должна решать проектируемая база данных.

Вариант 1

Каждый месяц владелец телефона оплачивает услуги связи. В квитанции, которую он получает при оплате, указывается абонентская плата за месяц, количество минут и сумма за звонки по межгороду, по городу, на мобильные телефоны.

Квитанция на оплату телефона _____

Абонент _____
(ФИО, адрес)

Период _____ Дата оплаты _____
(месяц, год)

Абонентская плата _____ Итого начислено _____

Вид разговора	**Тариф за минуту**	**Кол-во минут**	**Начислено**
по городу	20	…	…
по межгороду	190	…	…
на мобильные тел.	250	…	…

Спроектировать базу данных «*Телефонная компания*», основанную на данных квитанций за два месяца:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 2

Материальные ценности (столы, стулья, шкафы, компьютеры и т.д.), приобретенные организацией, передаются в конкретное подразделение под ответственность материально-ответственного лица. При этом оформляется *Акт передачи материальных ценностей*. По окончании отчетного периода (например квартала) проводится инвентаризация материальных ценностей, определяется процент износа и формируется *Ведомость инвентаризации*.

Акт передачи материальных ценностей № _____

от «____»_____20___ г.

Подразделение _____

Материально-ответственное лицо _____
(ФИО, должность)

Инв. №	Наименование	Цена	Кол-во	Балансовая стоимость

Ведомость инвентаризации материальных ценностей № _____

от «____»_____20___ г.

Подразделение _____

Материально-ответственное лицо _____
(ФИО, должность)

Инв. №	Наименование	Кол-во	Износ		Остаточная стоимость
			%	сумма	

Спроектировать базу данных «*Инвентаризация*», основанную на этих документах и содержащую данные о нескольких инвентаризациях:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 3

Расходный или приходный кассовый ордер выписывается для выдачи подотчетному лицу или получении от лица суммы по следующим основаниям: командировка, оплата за обучение, хозяйственные нужды, госпошлина, ликвидация разницы и т.д.

_____ кассовый ордер № _____
(Приходный или Расходный)

от «____» _____ 20 ___ г.

Принято / Получено от _____
(ФИО)

Основание _____
(наименование операции)

Сумма _____

Спроектировать базу данных «*Приходно-расходные ордера*», основанную на данных ордеров и содержащую данные за два месяца:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 4

Владельцы квартир ежемесячно оплачивают коммунальные услуги (оплата за газ, за электроэнергию, за воду, за отопление и т.д.). Для оплаты ему выдается квитанция, в которой указывается тариф, единица измерения, количество и сумма по каждой услуге.

Счет-квитанция

Лицевой счет _____

Плательщик _____
(ФИО)

Период _____
(месяц, год)

Адрес _____

Итого к оплате _____

Дата оплаты _____

Вид услуги	Ед. изм.	Тариф	Количество	Начислено
Газ	куб. м	460	…	…
Вода	куб. м	630	…	…
Отопление	гКал	43500	…	…
Электроэнергия	кВт-ч	175	…	…

Спроектировать базу данных «*Коммунальные платежи*», основанную на данных квитанций за два месяца:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 5

Фирма по прокату товаров заключает договора с клиентами на прокат различных видов товаров. Товары разделены на категории: бытовая техника, спорт и отдых, мебель. Имеется прейскурант на предоставляемые услуги.

Прейскурант		
Бытовая техника		
Наименование	Срок проката	Стоимость за день
Телевизор (50см)	до месяца	1000
Телевизор (50см)	от 1 до 3 мес	800
…	…	…
Мебель		
…	…	…

Договор № _____ от «____» _____20 ___ г.

на прокат «_____» инв. № _____
(наименование товара)

Срок возврата _____ Стоимость проката _____

Предоплата _____

Дата возврата _____ Пеня за каждый день просрочки — 1%

Стоимость с учетом пени _____

Реквизиты клиента:

Фамилия _____ № паспорта _____

Адрес и телефон _____

Спроектировать базу данных «*Прокат товаров*», основанную на этих документах и содержащую данные за несколько месяцев:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 6

Официанты ресторана при расчете с клиентом выдают ему чек с расчетом суммы заказа в соответствии с заказанными по меню блюдами.

Меню

Холодные закуски			
Наименование блюда	Стоимость	*Вторые блюда*	
…	…	Наименование блюда	Стоимость
		…	…
Первые блюда			
Наименование блюда	Стоимость		
…	…		

Чек № _____ от «____» _____20 ___ г.

Официант _____
(Фамилия, Имя)

Столик № _____

Время заказа _____

Наименование блюда	Стоимость порции	Количество порций
…	…	…
…	…	…

Стоимость заказа _____

Скидка _____ %

Обслуживание (2%) _____

Итого к оплате _____

Спроектировать базу данных «*Ресторан*», основанную на этих документах и содержащую данные за два месяца:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 7

Предприятие поставляет топливо (бензин марок АИ95, АИ92, АИ80 и дизельное топливо) на три заправки. На каждую поставку топлива оформляется накладная.

Накладная на поставку № _____ от «____» _____20 ___ г.

Название заправки _____

Адрес _____

Наименование топлива	**Цена за литр**	**Количество**	**Стоимость**
Итого по накладной _____			

Спроектировать базу данных «*Поставки топлива*», основанную на этом документе и содержащую данные за два месяца:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Вариант 8

Коммерческий банк выдает предприятиям кредиты на различных условиях, заключая с ними договора.

Условия кредитования			
Название кредита	Сумма кредита, $	Ставка	Срок погашения
Развитие бизнеса	от 10 000 до 30 000	10%	7 лет
Стабилизационнный	от 5 000 до 12 000	12%	3 года
…	…	…	…

Договор № _____ от «____» _____20 ___ г.

на предоставление кредита «_____» в размере _____
(название кредита)

Реквизиты клиента:

Название предприятия _____ Расчетный счет _____

Адрес _____

Спроектировать базу данных «*Банковские кредиты*», основанную на этих документах и содержащую данные за несколько лет:

1. Описать предметную область разрабатываемой базы данных (см. п. 1 примера 10.1).

2. Выделить информационные объекты предметной области и построить ER-диаграмму (см. п. 2 примера 10.1 и рис. 10.3).

3. Определить структуру таблиц (поля, типы данных, ключи) и установить связи между таблицами (см. п. 3 примера 10.1, табл. 10.2 и рис. 10.4). Предусмотреть дополнительные поля для идентификации объектов (коды или номера). Список полей дополнить с целью расширения возможностей базы данных. Для каких-либо полей (по своему усмотрению) задать условие на значение и сообщение об ошибке.

Контрольные вопросы

1. Опишите этапы процесса проектирования базы данных.
2. Что такое модель данных? Назовите основные виды моделей.
3. Назовите основные термины реляционных баз данных и поясните их.
4. Какие существуют подходы к проектированию реляционных баз данных?
5. Опишите процесс проектирования на примере своего варианта.

Тема 11. Реализация проекта и управление базой данных

Практические задания

Целью выполнения заданий по данной теме является реализация базы данных, спроектированной в процессе выполнения практических заданий темы 10, в СУБД Access. Задания предназначены для углубления теоретических знаний и закрепления практических навыков, полученных при изучении предыдущих тем. Задания 11.7 и 11.8 являются дополнительными, они позволяют расширить возможности разрабатываемой базы данных по усмотрению разработчика. Задание 11.9 (повышенной сложности) выполняется после изучения лекционной темы «*Защита базы данных*» и соответствующих разделов справочной системы СУБД Access.

Задание 11.1. В СУБД Access создать новую базу данных. Создать таблицы, руководствуясь разработанной структурой (см. п. 3 соответствующего варианта темы 10).

Задание 11.2. В соответствии с проектом создать связи между таблицами и установить для них параметры обеспечения целостности данных.

Задание 11.3. Создать формы с помощью **Мастера форм** и доработать их в режиме **Конструктора**:

1) простую форму для одного из справочников;
2) составную форму для одновременной загрузки или просмотра данных второго справочника и учетной информации;
3) для полей с датами или номерами телефона задать маску ввода;
4) выполнить подстановку данных первого справочника в составную форму, используя элемент управления *Поле со списком*;
5) создать кнопки для управления составной формой.

Задание 11.4. Заполнить таблицы данными (не менее 5 записей для справочников и не менее 20 для учетной информации) с помощью созданных форм.

Задание 11.5. Разработать запросы:

а) в режиме **Конструктора**

Вариант 1

1) список всех уплативших за телефон с 1 по 25 число каждого месяца, отсортированный по дате;
2) список квитанций, оплаченных абонентом, фамилия которого является параметром запроса;
3) список всех абонентов, уплативших за телефон после 25 числа, с расчетом суммы пени за просрочку платежа (пеня составляет 1% от абонементной платы);
4) стоимость оплаты для каждого абонента;
5) сумма каждого вида платежа за каждый месяц и общая сумма платежей за каждый месяц;

Вариант 2

1) список инвентарных объектов, которые числятся за материально-ответственным лицом, фамилия которого является параметром запроса;
2) список инвентарных объектов с процентом износа более 50%;
3) инвентарные объекты с рассчитанной балансовой стоимостью, суммой износа и остаточной стоимостью;
4) общая стоимость материальных ценностей, числящихся за каждым материально-ответственным лицом;
5) количество материальных ценностей каждого наименования, которые числятся за каждым материально-ответственным лицом;

Вариант 3

1) список всех подотчетных лиц, получивших командировочные;
2) список всех подотчетных лиц, получивших деньги по расходным кассовым ордерам, с полной информацией по каждому из них;
3) список приходных ордеров с суммой от А до Б, где А и Б являются параметрами запроса;
4) количество подотчетных лиц, получивших деньги по каждому основанию за каждый месяц;
5) разность между суммами по приходным ордерам и расходным ордерам за каждый месяц;

Вариант 4

1) список всех уплативших за воду, отсортированный по фамилии;
2) список квитанций, оплаченных с 26 по 30 число каждого месяца;
3) сумма платежей за газ за каждый месяц;
4) общая сумма платежей по каждому квартиросъемщику за каждый месяц;
5) средний расход воды по каждой квартире за каждый месяц;

Вариант 5

1) информация о договорах о прокате товара, наименование которого является параметром запроса;
2) предоплата за взятый товар за каждый месяц;
3) стоимость проката возвращенных товаров по каждому наименованию;
4) новая таблица со списком клиентов, возвративших товар позже установленного срока и расчетом пени;
5) стоимость пени, полученной за каждый месяц;

Вариант 6

1) список закусок, которые были заказаны в количестве большем двух порций, упорядоченный по дате;
2) количество заказанных порций по каждому блюду за каждый месяц;
3) стоимость заказов за каждый месяц;
4) сумма заказа по каждому чеку;
5) предоставление клиенту скидки 5% на оплату чека, если стоимость заказа превышает 100 000, и 10%, если стоимость заказа превышает 200 000;

Вариант 7

1) список накладных, по которым поставлялся бензин марки АИ95, упорядоченный по дате поставки;
2) стоимость поставки по каждой накладной;
3) количество каждого вида топлива, поставленного на каждую заправку;
4) стоимость поставок по каждому виду топлива за каждый месяц;
5) увеличение цены на АИ92 на 10% (обновление);

Вариант 8

1) список договоров, упорядоченный по дате, заключенных в году, который является параметром запроса;
2) список договоров по кредитам с годовой ставкой более 10%;
3) сумма по каждому виду кредита за каждый год;
4) вычисление ежемесячной выплаты (при выплате равными долями) по договору, который является параметром запроса;
5) количество договоров по каждому виду кредита с суммами более 20 000 $;

б) в режиме **SQL**

Вариант 1

1) увеличение на 2% абонентской платы (обновление);
2) оплата каждого абонента за каждый месяц;
3) список абонентов говоривших по межгороду более 40 минут и оплативших после 25 числа.

Вариант 2

1) список инвентарных объектов балансовой стоимости от 80 000р. до 170 000р.;
2) остаточная стоимость материальных ценностей каждого наименования по каждой инвентаризации;
3) передача материальных ценностей одного материально-ответственного лица другому, фамилии лиц являются параметрами запроса (обновление);

Вариант 3

1) все данные по подотчетным лицам, первая буква фамилий которых является параметром запроса;
2) сумма поступившая по приходным ордерам по каждому основанию за каждый месяц;
3) удаление информации о расходных ордерах на хознужды с суммой менее 10000 (запрос не выполнять).

Вариант 4

1) список всех плательщиков, уплативших коммунальные платежи за месяц, который является параметром запроса;
2) общая сумма платежей по каждой услуге за каждый месяц;
3) увеличение тарифа на газ на 7% (обновление).

Вариант 5

1) список клиентов, которые воспользовались услугами проката более двух раз;
2) количество договоров, заключенных каждый месяц;
3) удаление информации о договорах, по которым товар возвращен более трех лет назад.

Вариант 6

1) увеличение стоимости блюда, наименование которого является параметром запроса, на 10% (обновление);
2) количество чеков, оформленных каждым официантом за каждый месяц;
3) сумма за обслуживание по каждому официанту.

Вариант 7

1) список накладных со стоимостью от А до Б, где А и Б параметры запроса;
2) стоимость поставок на каждую заправку за каждый месяц;
3) удаление информации о поставках АИ80 за месяц, который является параметром запроса (запрос не выполнять).

Вариант 8

1) договора по долгосрочным кредитам (более 5 лет);
2) договора с суммой кредита от А до Б, где А и Б являются параметрами запроса;
3) предприятие, получившее больше всего кредитов.

Задание 11.6. Разработать отчеты с группировкой и итоговыми вычислениями.

Задание 11.7.* Разработать запросы различных типов для дальнейшей обработки БД.

Задание 11.8.* Разработать макросы для автоматизации работы и кнопочную форму для управления базой данных.

Задание 11.9.** Создать несколько пользователей БД с различными правами доступа.

Литература

Основная

1. Разоренова, Т.Р. Технологии организации, хранения и обработки данных: методическое пособие и лабораторный практикум для студентов специальностей 1-26 02 02 «Менеджмент», 1-25 01 08 «Бухгалтерский учет, анализ и аудит», 1-96 01 01 «Таможенное дело», 1-25 01 07 «Экономика и управление на предприятии», «Финансовое обеспечение и экономика боевой и хозяйственной деятельности войск (сил)» / Т.Р. Разоренова, И.М. Желакович. – Минск: БНТУ, 2006. – 94 с.

2. Астахова, И.Ф. SQL в примерах и задачах: учебное пособие / И.Ф. Астахова, А.П. Толстобров, В.М. Мельников. – Минск: Новое знание, 2002. – 176 с.

3. Харитонова, И. Программирование в Access 2002: учебный курс / И. Харитонова, Н. Вольман. – СПб.: Питер, 2002. – 480 с.

4. Хернандес, Дж. SQL запросы для простых смертных: практическое руководство по манипулированию данными в SQL / Майкл Дж. Хернандес, Джон Л. Вьескас. – М.: Лори, 2003. – 473 с.

5. Электронные курсы по СУБД SQL Server 2000 [Электронный ресурс] / Интернет-Университет Информационных Технологий – дистанционное образование. – Россия, 2003–2010. – Режим доступа: http://www.INTUIT.ru.

6. Малыхина, М.П. Базы данных: основы, проектирование, использование / М.П. Малыхина. – СПб.: БХВ-Петербург, 2004. – 512 с.

Дополнительная

1. Дейт, К.Дж. Введение в системы баз данных, 7-е издание / К.Дж. Дейт; пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1072 с.

2. Гарнаев, А.Ю. Excel, VBA, Internet в экономике и финансах / А.Ю. Гарнаев. – СПб.: БХВ-Петербург, 2003. – 816 с.

3. Мамаев, Е.В. Microsoft SQL Server 2000 / Е.В. Мамаев. – СПб.: БХВ-Петербург, 2004.– 1262 с.

4. Харитонова, И. Microsoft Access 2000: разработка приложений / И. Харитонова, В. Михеева. – СПб.: БХВ-Петербург, 2000. – 832 с.

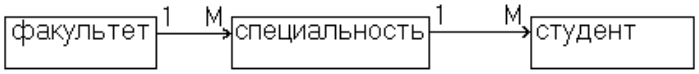
5. Федоров, А. Microsoft SQL Server 2008. Краткий обзор ключевых новинок / А. Федоров. – Киев: Изд. группа BHV, 2008. – 128 с.

6. Разоренова, Т.Р. Лабораторный практикум по информатике «Система управления базами данных MS Access» для студентов II курса дневного отделения специальностей Э 01.09.00 «Менеджмент» и Э 01.03.00 «Экономика и управление на предприятии» / Т.Р. Разоренова, О.В. Альшевская, И.М. Желакович. – Минск: Технопринт, 2000. – 59 с.

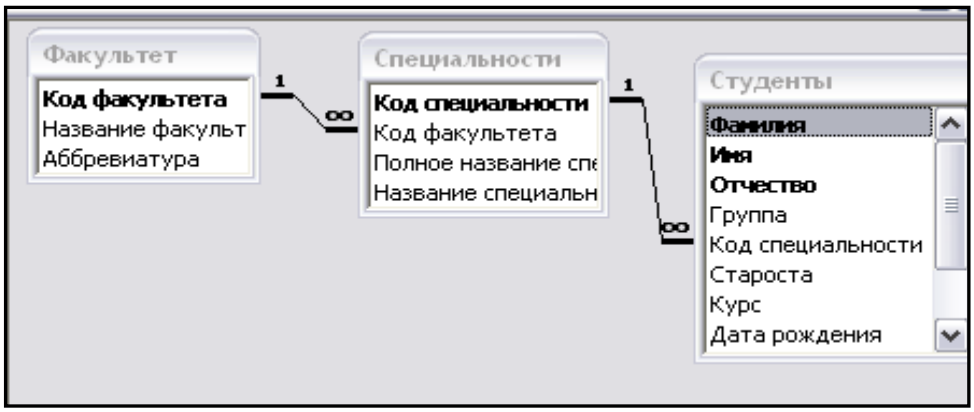
7. Савицкий, Н.И. Технологии организации, хранения и обработки данных: учебное пособие / Н.И. Савицкий. – М.: ИНФРА-М, 2001. – 232 с.

Приложение

Представленное учебное издание является логическим продолжением и расширением ранее разработанного методического пособия «Технологии организации, хранения и обработки данных» [1], в котором студенты по предложенному сценарию создают прототип учебной базы «Студенты», состоящей из трех связанных таблиц, описывающих факультеты, специальности и студентов. Информационная модель исходной базы данных приведена на рисунке.



Информационные объекты – **Факультет, Специальности и Студенты** описаны своими реквизитами и связаны логической моделью следующим образом:



Структура таблицы **Факультет**

Поле	Тип данных	Размер поля
Код факультета	счетчик	длинное целое
Название факультета	текстовый	100
Аббревиатура	текстовый	5

Исходные данные таблицы **Факультет**

Код факультета	*Название факультета*	*Аббревиатура*
1	Технологий управления и гуманитаризации	ФТУГ
2	Информационных технологий и робототехники	ФИТР
3	Менеджмента, маркетинга и предпринимательства	ФММП
4	Автомобили и тракторы	АТФ
5	Машиностроительный	МСФ

Структура таблицы **Специальности**

Поле	Тип данных	Размер
Код специальности	Счетчик	Длинное целое
Код факультета	Числовой (Мастер подстановок...)[1]	
Полное название специальности	Текстовый	100
Название специальности	Текстовый	10

[1] Поле Код факультета и аббревиатура из таблицы Факультет

Исходные данные таблицы **Специальности**

Код специальности	*Код факультета*	*Полное название специальности*	*Название специальности*
1	1	Экономика и управление на предприятии	ЭУП
2	1	Таможенное дело	ТД
3	2	Программное обеспечение информационных	ПОИТ
4	1	Бухгалтерский учет, анализ и аудит	БУАиА
5	3	Управление в социально-экономических сферах	Туризм
6	2	Автоматизация финансовых операций	АФО
7	3	Коммерческая деятельность	КД

Структура таблицы **Студенты**

Поле	Тип данных	Размер поля	Обязательное поле	Индексированное поле
Фамилия	Текстовый	50	Да	Да (допускаются совпадения)
Имя	Текстовый	50	Да	Да (допускаются совпадения)
Отчество	Текстовый	50	Да	Да (допускаются совпадения)
Группа	Текстовый	10	Нет	Нет
Код специальности	Числовой (подстановка из таблицы Специальности)[1]			
Староста	Логический	Формат поля: Вкл/Выкл	Нет	Нет
Курс	Текстовый (подстановка из набора фиксированных значений)[2]	3	Нет	Нет
Дата рождения	Дата/время	Краткий формат	Нет	Нет

Исходные данные таблицы **Студенты**

Студенты : таблица

Фамилия	Имя	Отчество	Группа	Код специальности	Староста	Курс	Дата рождения
Андреев	Андрей	Андреевич	107414	3	☑	II	26.07.1987
Иванов	Иван	Иванович	108415	2	☐	I	15.06.1987
Исаченко	Елена	Сергеевна	107514	3	☑	I	06.01.1988
Комарова	Ольга	Сергеевна	107414	3	☐	II	05.09.1987
Краснова	Ирина	Петровна	108515	1	☐	I	16.10.1987
Крюк	Инна	Федоровна	108515	1	☑	I	26.11.1987
Миронов	Игорь	Семенович	108513	1	☑	II	28.03.1988
Мухина	Любовь	Ивановна	108513	1	☐	II	16.02.1988
Петров	Петр	Петрович	108415	2	☑	I	05.05.1987
Степанов	Степан	Степанович	108513	1	☐	II	08.05.1988

[1] Поле Код специальности и Название специальности из таблицы Специальности.

[2] Список фиксированных значений: I, II, III, IV, V (набирать большими латинскими буквами).

Содержание

Учебное издание

Разорёнова Т.Р., Альшевская О.В.

УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ

Учебно-методическое пособие для студентов специальностей

1–96 01 01 “Таможенное дело”
1–26 02 02 “Менеджмент”
1–25 01 08 “Бухгалтерский учет, анализ и аудит”
1–25 01 07 “Экономика и управление на предприятии”,
”Финансовое обеспечение и экономика боевой
и хозяйственной деятельности войск (сил)”