МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«УЛЬЯНОВСКОЕ ВЫСШЕЕ АВИАЦИОННОЕ УЧИЛИЩЕ ГРАЖДАНСКОЙ АВИАЦИИ (ИНСТИТУТ)»

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ, БАЗЫ ДАННЫХ

РАБОТА С СУБД ACCESS

Учебное пособие

Рекомендовано
редакционно-издательским советом института

Ульяновск 2015

УДК 004.65 (075.8)
ББК 3973.26-018.2я7
И74

Информационное обеспечение, базы данных. Работа с СУБД Access : учеб. пособие / сост. К. А. Толстов, А. М. Лебедев. – Ульяновск : УВАУ ГА(И), 2015. – 103 с.

ISBN 978-5-7514-0230-3

Содержит необходимые сведения о порядке проектирования и создания баз данных с использованием СУБД Access, требования и задачи, решаемые с использованием баз данных в гражданской авиации, необходимый иллюстративный материал и примеры.

Подготовлено в соответствии с Федеральным государственным образовательным стандартом и рабочей программой дисциплины «Информационное обеспечение, базы данных».

Предназначено для курсантов и студентов заочной формы обучения направления подготовки «Управление качеством» профиль 1 – Управление качеством в производственно-технологических системах, изучающих дисциплину «Информационное обеспечение, базы данных», а также может быть использовано курсантами и студентами технических специальностей и профилей подготовки при изучении дисциплин «Информатика» и «Современные информационные технологии».

УДК 004.65 (075.8)
ББК 3973.26-018.2я7

ISBN 978-5-7514-0230-3

© ФГБОУ ВПО «Ульяновское высшее авиационное училище гражданской авиации (институт)», 2015

Оглавление

ВВЕДЕНИЕ

Развитие средств вычислительной техники обеспечило возможности для создания и широкого использования информационных систем различного назначения, в том числе и в гражданской авиации (ГА). Определяющим фактором при создании эффективных систем обработки данных стала концепция баз данных.

В предлагаемом пособии рассматриваются возможности СУБД Access, описана технология работы пользователя от проектирования и создания базы данных до реализации задач и создания приложений. Одним из важных вопросов является разработка и создание целостной базы данных. При всех возможностях средств СУБД Access в значительной степени от разработчика зависит, будет ли работа с базой данных легкой и понятной при вводе данных и интерпретации выходных данных. База данных должна иметь законченный и полноценный облик. Необходимо помнить, что качество базы данных проявляется во взаимосвязи с входными и выходными документами. Особое значение имеет возможность создания экранных форм и отчетов, совпадающих по структуре с реальными документами.

С построением корректной базы данных тесно связана разработка и реализация задач пользователя. При решении многих задач достаточно использования объектов Access, которые легко создаются в диалоговом режиме.

Для реализации практических задач возникает необходимость в создании макросов и модулей на языке Visual Basic for Application (VBA), которые позволяют объединить отдельные объекты, обеспечивают обработку событий, возникающих в процессе диалога. В пособии приведена объектная модель СУБД Access, последовательность обработки событий, приведены необходимые примеры.

Сведения, приведенные в разделах 1–6, позволят научиться разрабатывать и эксплуатировать созданные базы данных. Следует отметить, что при создании баз данных важным моментом является разработка интерфейса пользователя. Удачно спроектированные формы и отчеты значительно облегчают работу пользователя, способствуют быстрому освоению работы с базой данных.

Сведения, приведенные в разделе 7, позволяют создавать базы данных на ином качественном уровне. Рассматриваются вопросы разработки приложений пользователя, изложены основы создания макросов, приведено описание встроенного

языка программирования. Модули, разработанные с использованием языка программирования VBA, позволяют объединить в одном приложении пользователя отдельные запросы, формы и отчеты, а также обеспечивают обработку событий, возникающих в процессе диалога пользователя.

Раздел 8 посвящен применению баз данных в ГА. В разделе приведены требования стандартов к базам данных систем менеджмента безопасности авиационной деятельности поставщиков обслуживания, задачи, решаемые с использованием баз данных, рассмотрена система управления безопасностью полетов.

При проектировании баз данных необходимо провести тщательный анализ предметной области, построить инфологическую модель, провести нормализацию, реализовать ее с помощью СУБД, разработать приложения пользователя, что позволит осуществлять эффективную обработку данных.

1. ОСНОВНЫЕ СВЕДЕНИЯ И НАЧАЛО РАБОТЫ В СУБД ACCESS

База данных организуется на машинном носителе и содержит сведения о различных сущностях определенной предметной области – реальных объектах или процессах.

СУБД Access использует реляционную модель данных. Реляционная база данных представляет собой множество взаимосвязанных двумерных таблиц – реляционных таблиц, в каждой из которых содержатся сведения об одной сущности.

Структура реляционной таблицы определяется составом и последовательностью полей, соответствующих ее столбцам, с указанием типа данных, размещаемых в поле. Каждое поле отражает определенную характеристику сущности, а соответствующий столбец содержит данные одного типа.

Содержание таблицы заключено в ее строках. Каждая строка таблицы содержит данные о конкретном экземпляре сущности и называется записью. Для однозначного определения каждой записи таблица должна иметь уникальный ключ. Ключ может состоять из одного или нескольких полей. По значению ключа определяется единственная запись.

Связи между таблицами дают возможность совместно использовать данные из разных таблиц, связь каждой пары таблиц обеспечивается одинаковыми

полями в них – ключом связи (внешним ключом). Внешним ключом в связанной таблице является уникальный ключ главной таблицы. Связи между таблицами в реляционной базе данных могут быть один-к-одному (1:1) или один ко многим (1:M).

На рис. 1.1 показаны таблица с перечнем курсантов и таблица учета результатов сессии, которые связаны по полю IDkursant.

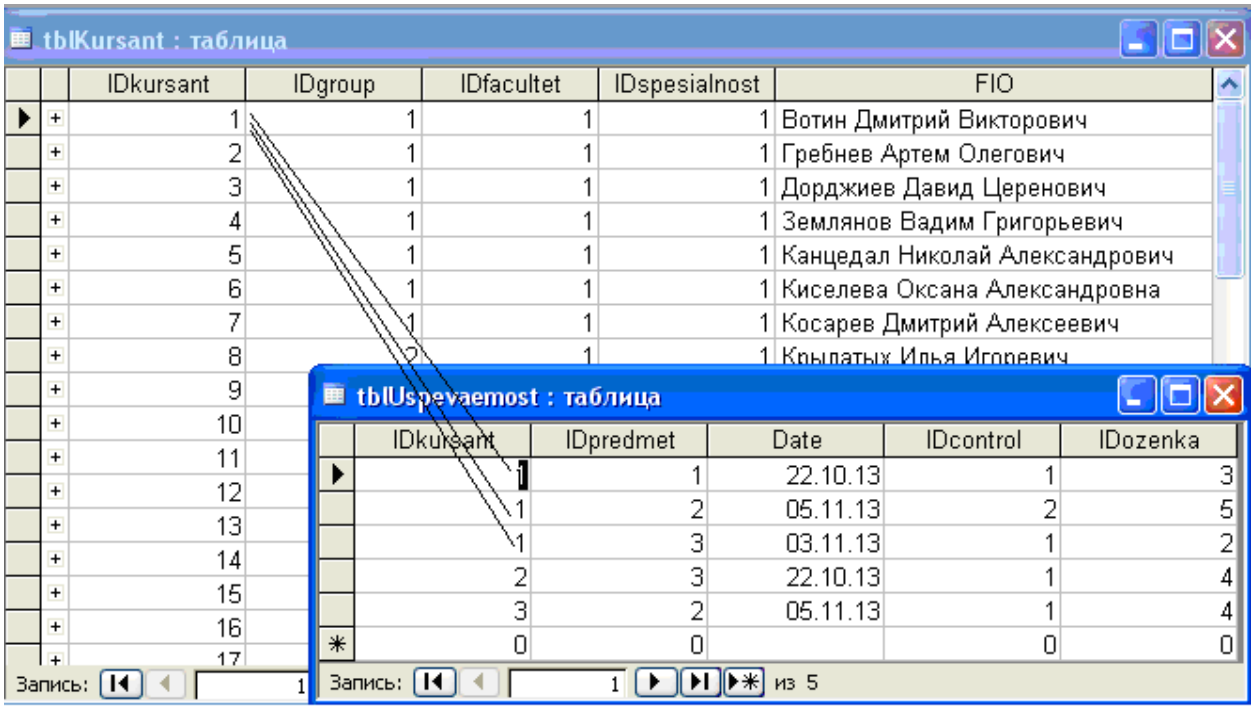

Рис. 1.1. Связанные таблицы **Курсант** и **Успеваемость**

Размещение сведений о каждой сущности в отдельной таблице и связывание таблиц позволяет избежать повторения данных в разных таблицах, обеспечивает целостность данных и упрощает процесс обновления и поиска. При этом обеспечивается однократный ввод данных при загрузке и корректировке базы данных. Если данные двух таблиц в приведенном примере разместить в одной таблице, то каждая запись должна соответствовать одному курсанту (студенту) во всех записях о результатах сессии. Причем данные о курсанте (студенте) – фамилия, имя, отчество и др. – будут повторяться во всех записях о курсантах (студентах) каждой группы.

В СУБД Access процесс создания базы данных включает создание схемы данных. Схема данных наглядно отображает таблицы и связи между ними, а

также обеспечивает использование связей при обработке данных и целостность базы данных.

СУБД Access ориентирована на работу с объектами базы данных, к которым относятся *таблицы*, *формы*, *запросы*, *отчеты*, *макросы* и *модули*.

Для типовых процессов обработки данных – просмотра, обновления, поиска, получения отчетов – имеются средства конструирования. Формы и отчеты состоят из графических элементов, называемых элементами управления, которые служат для связи объектов с записями таблиц, которые являются источниками данных.

При создании приложений пользователя ими могут использоваться макросы и модули на языке программирования VBA.

Таблицы создаются пользователем для хранения данных. Таблица состоит из полей (столбцов) и записей (строк). Каждое поле содержит одну характеристику (атрибут) объекта предметной области. В записи отражаются сведения об одном экземпляре этого объекта.

Запросы создаются пользователем для выборки необходимых данных из одной или нескольких таблиц.

Формы предназначены для ввода и просмотра данных в удобном виде. Форма может соответствовать привычному для пользователя документу.

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

Макросы содержат описание действий, которые должны быть выполнены в ответ на некоторое событие. Каждое действие реализуется макрокомандой. Макрос позволяет объединить различные операции обработки данных в одном приложении.

Модули содержат программы на языке VBA, которые разрабатываются пользователем для реализации различных процедур при создании приложения.

Все объекты базы данных могут размещаться в одном файле на диске, что позволяет упростить технологию ведения базы данных и ее приложений и обеспечить высокую компактность и эффективность обработки данных.

СУБД ACCESS располагает разнообразными диалоговыми средствами, позволяющими автоматизировать процесс создания различных объектов базы данных.

После запуска отображается окно СУБД ACCESS (рис. 1.2).

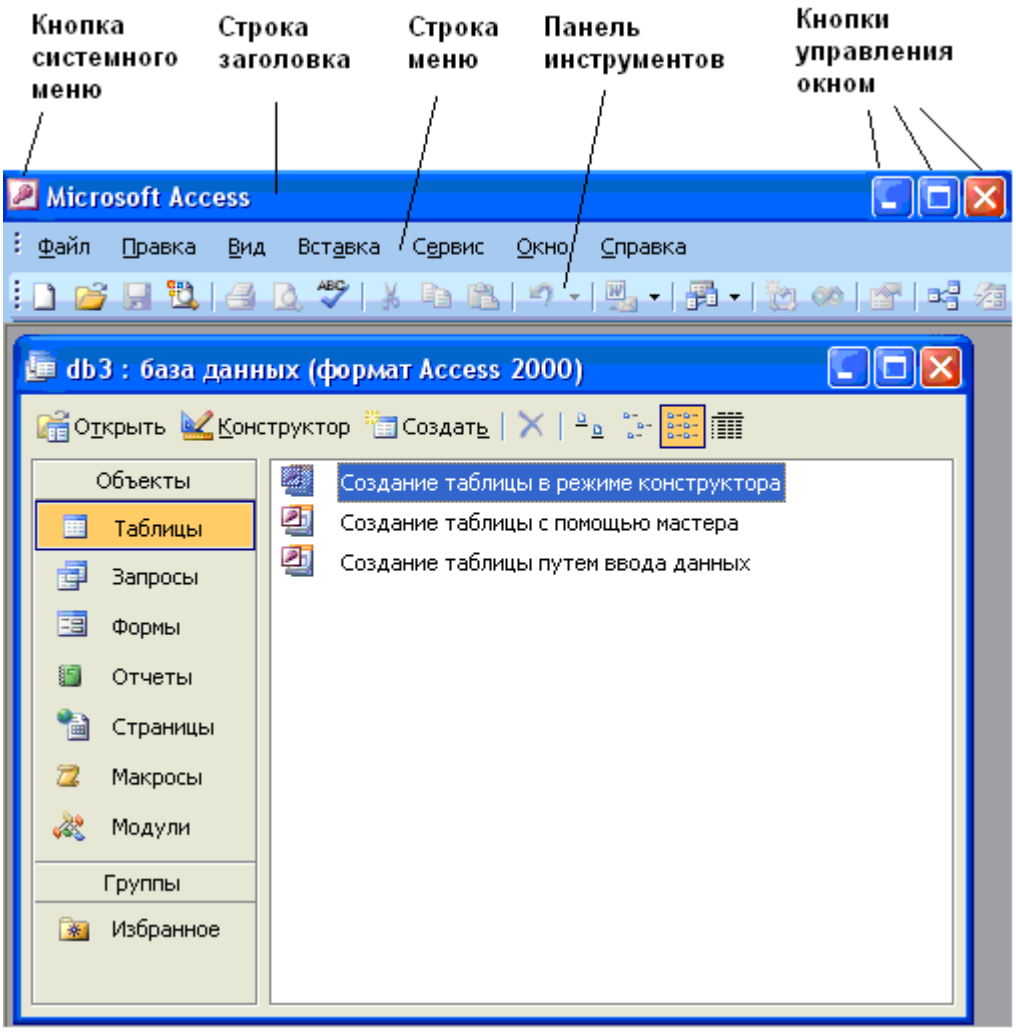

Рис. 1.2. Окно СУБД Access

Строка заголовка содержит кнопку системного меню, название приложения – Microsoft Access – и три кнопки, управляющие представлением окна на экране.

Строка меню содержит несколько пунктов, каждый из которых имеет собственное ниспадающее меню (список команд), открываемое щелчком мыши на этом пункте.

Панель инструментов состоит из значков кнопок, каждая из которых реализует наиболее часто используемую команду меню более быстрым и удобным способом. Пользователь может отобразить необходимые для работы панели инструментов, а также настроить собственные панели инструментов путем выбора нужных кнопок.

После запуска СУБД Access одновременно с его окном открывается диалоговое окно, позволяющее начать создание новой базы данных или открыть существующую (рис. 1.3).

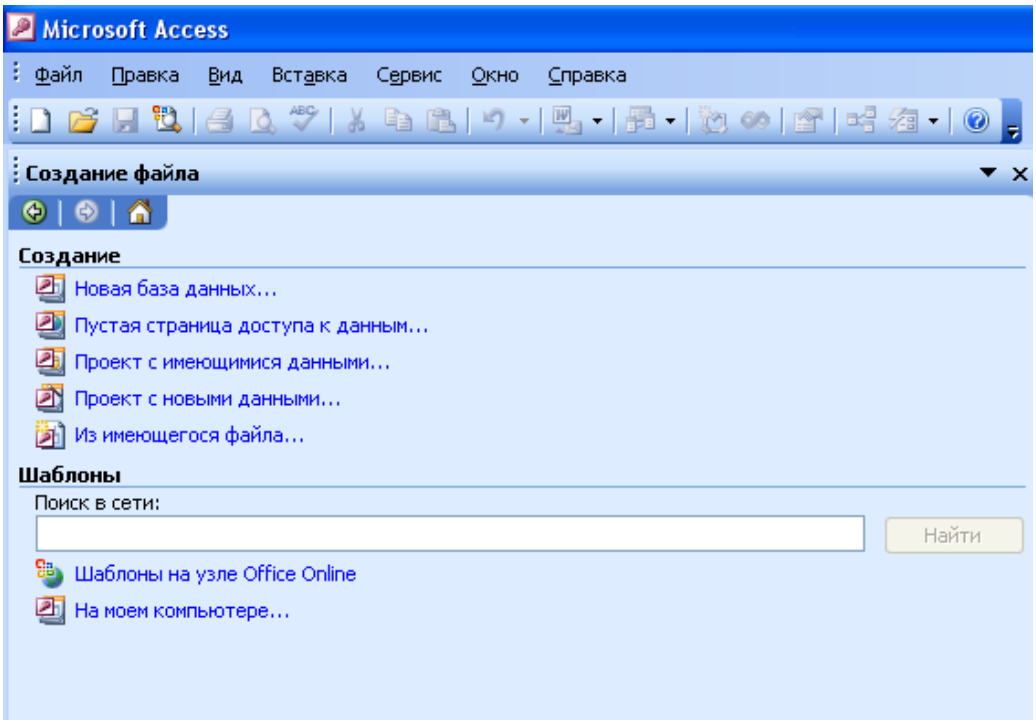

Рис. 1.3. Диалоговое окно СУБД Access

Если диалоговое окно не появляется, необходимо выбрать пункт меню Вид – Область задач.

Перейдем к рассмотрению основ проектирования баз данных.

2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Перед созданием базы данных необходимо определить основные объекты предметной области и связи между ними, формы входных и выходных документов, предполагаемые запросы пользователя и потребности в обработке данных. Структура предметной области отображается, как правило, инфологической моделью предметной области. На основе этой модели достаточно легко строится реляционная модель базы данных.

При разработке модели могут использоваться два подхода. В первом случае сначала определяются основные задачи, для решения которых строится база данных, и выявляются данные, необходимые для решения поставленных задач.

При использовании второго подхода сразу устанавливаются объекты предметной области. На практике используют сочетание обоих подходов.

В процессе разработки модели данных необходимо выделить информационные объекты, соответствующие требованиям нормализации данных и определить связи между ними, что позволяет создать реляционную базу данных без избыточности.

В процессе выделения информационных объектов необходимо выявить документы и их реквизиты, подлежащие хранению в базе данных.

При определении логической структуры данных на основе инфологической модели каждый информационный объект адекватно отображается двумерной таблицей, связи между таблицами соответствуют связям между информационными объектами. Каждый атрибут становится столбцом таблицы.

Выделение информационных объектов рассмотрим на примере предметной области **Сессия**.

Необходимо создать базу данных, содержащую информацию об итогах сдачи курсантами (студентами) сессии.

На этапе анализа предметной области были выявлены документы, являющиеся источниками для создания базы данных: списки курсантов (студентов) по группам, перечень изучаемых дисциплин, результаты сдачи экзаменов (зачетов) по каждой дисциплине.

На основе анализа документа **Список группы** можно выделить следующие объекты (в скобках указаны атрибуты объектов):

1. **Курсант** (Фамилия, Имя, Отчество, ДатаРождения, Адрес, Телефон).
2. **Факультет** (НаименованиеФакультета).
3. **Специальность** (НаименованиеСпециальности).
4. **Группа** (наименование группы).

На основе анализа документа «Экзаменационная ведомость» можно выделить объекты:

1. **Ведомость** (КодКурсанта, КодДисциплины, КодОценки, КодОтчетности, Дата, ФиоПрепод).
2. **Дисциплина** (НаименованиеДисциплины).
3. **Оценка** (НаимОценки).
4. **Отчетность** (НаимОтчетности).

Инфологическая модель предметной области представлена на рис. 2.1.

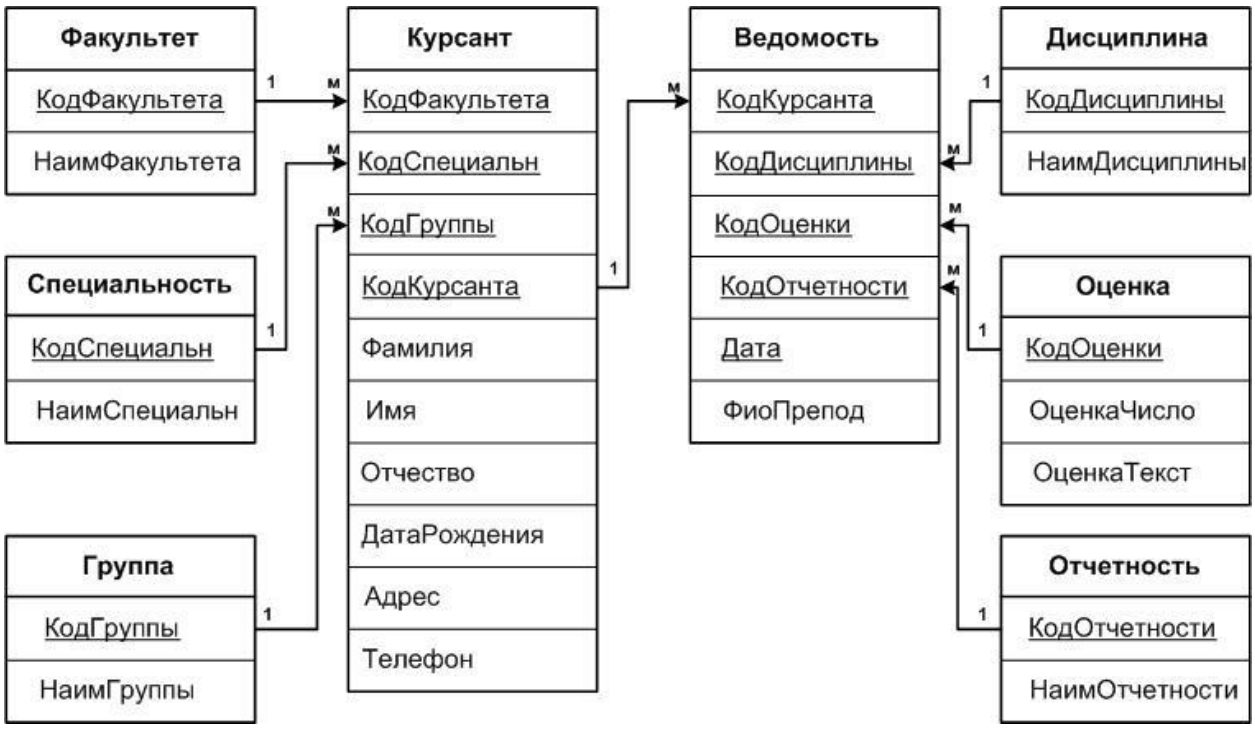

Рис. 2.1. Инфологическая модель предметной области **Сессия**

Связь между объектами, например, **Курсант → Ведомость** характеризуется одно-многозначными отношениями, поскольку курсант (студент) сдает в сессию несколько экзаменов и зачетов. Аналогично устанавливается связь между объектами **Факультет → Курсант**, **Специальность → Курсант**, **Группа → Курсант**, **Курсант → Ведомость**, **Дисциплина → Ведомость**, **Оценка → Ведомость**, **Отчетность → Ведомость**.

Инфологическая модель является адекватным отображением предметной области, не требующим преобразований. Каждый объект предметной области отображается соответствующей таблицей. Структура таблицы определяется реквизитами объекта, где каждый столбец соответствует одному из реквизитов. Ключевые реквизиты объекта (на рисунке подчеркнуты) образуют первичный ключ таблицы. Для каждого столбца задаются формат и размер данных. Строки (записи) таблицы соответствуют экземплярам объекта.

В СУБД Access создается схема данных. Связи между таблицами устанавливаются с обеспечением целостности данных в соответствии с построенной моделью. Внешний вид схемы данных практически совпадает с графическим представлением логической модели (рис. 2.2).

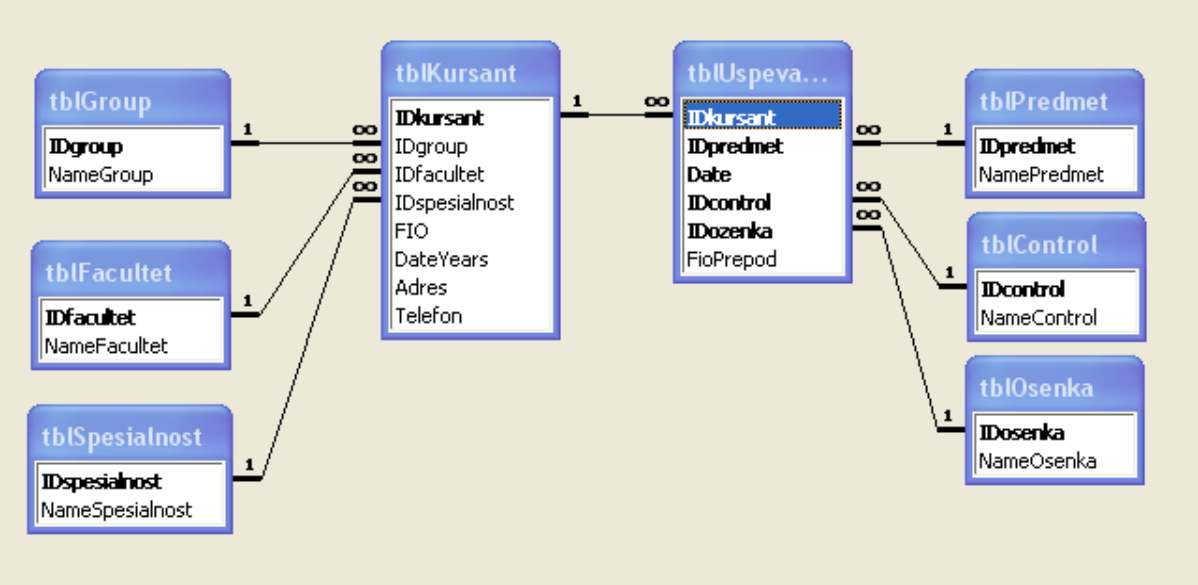

Рис. 2.2. Схема данных в Access

На представленной схеме данных прямоугольники отображают таблицы базы данных с полным списком полей, а связи показывают, по каким полям осуществляются взаимосвязи таблиц. Имена ключевых полей представлены в верхней части таблиц и выделены жирным шрифтом.

Рассмотренные этапы проектирования баз данных позволяют сравнительно легко получить логическую структуру базы данных Access.

3. СОЗДАНИЕ НОВОЙ БАЗЫ ДАННЫХ

Создание новой базы данных в Access осуществляется в соответствии с ее структурой, полученной в результате проектирования. Структура базы данных определяется составом таблиц и их взаимосвязями. Взаимосвязи между двумя таблицами реализуются через внешний ключ, входящий в состав полей связываемых таблиц.

Создание базы данных с помощью СУБД Access начинается с формирования структуры таблиц. При этом формируется состав полей и задается их описание.

Для создания новой таблицы в окне базы данных необходимо выбрать закладку Таблицы и нажать кнопку Создать. В открывшемся окне выбрать режим создания таблицы (рис. 3.1).

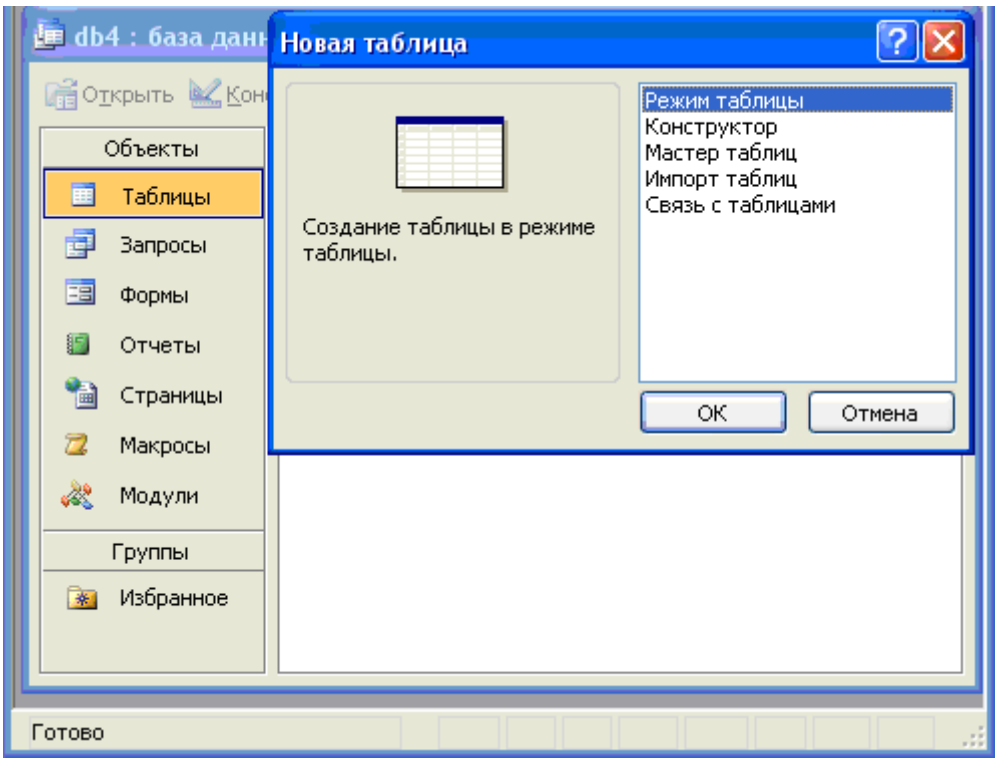

Рис. 3.1. Диалоговое окно выбора режима создания таблицы

При создании таблиц рекомендуется использовать режим **Конструктор**, который позволяет пользователю самому указать параметры всех элементов таблицы.

При выборе режима **Конструктор** появляется окно (рис. 3.2), в котором определяется структура создаваемой таблицы. Для определения поля в окне Таблица задаются **Имя поля**, **Тип данных**, **Описание** (краткий комментарий), **Свойства поля** (вкладки Общие, Подстановка).

В Microsoft Access действуют следующие ограничения на имена полей, элементов управления и объектов:

– имя должно содержать не более 64 знаков;

– имя может включать любую комбинацию букв, цифр, пробелов и специальных знаков за исключением точки (.), восклицательного знака (!), надстрочного знака (‘) и квадратных скобок ([ ]);

– имя не должно начинаться со знака пробела;

– имя не должно включать управляющие знаки (с кодами ASCII от 0 до 31);

– имя не должно включать прямые кавычки (") в именах таблиц, представлений и сохраненных процедур в проекте Microsoft Access.

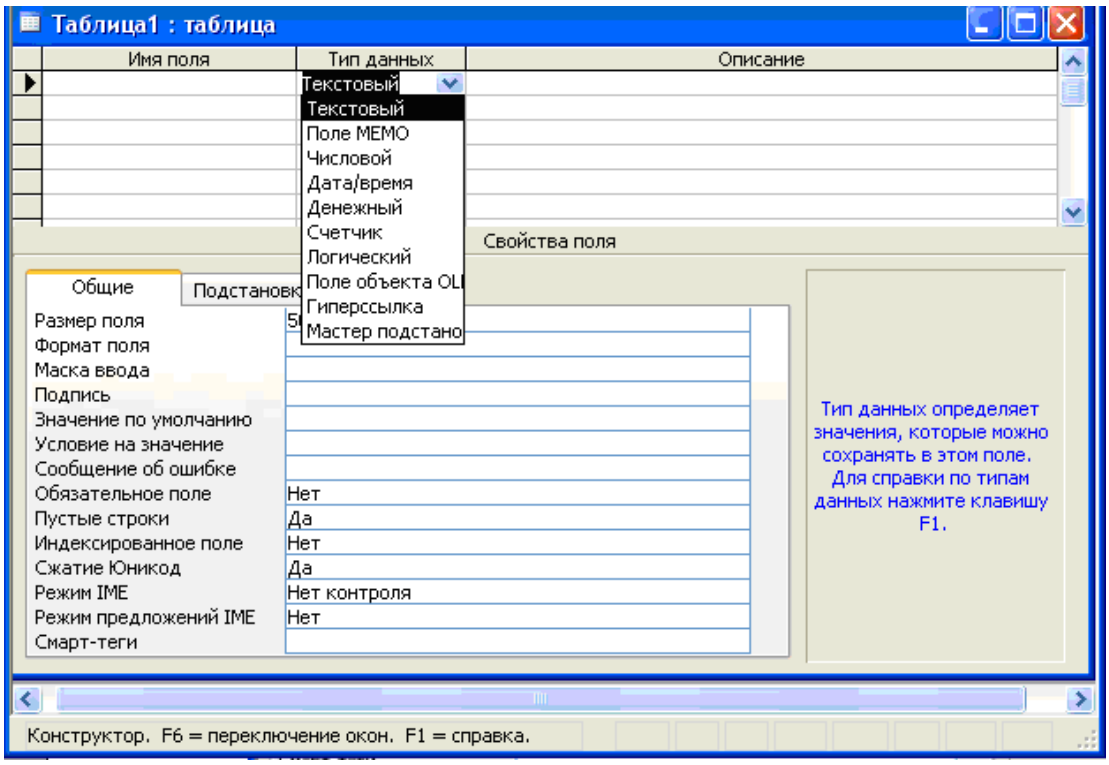

Рис. 3.2. Окно создания таблицы в режиме **Конструктора**

Тип данных определяется значениями, которые предполагается вводить в поле. При выборе типа данных, используемых в поле, необходимо учитывать следующее:

– какие значения должны отображаться в поле (например, нельзя хранить текст в поле, имеющем числовой тип данных);

– сколько места необходимо для хранения значений в поле;

– какие операции должны производиться со значениями в поле (например, суммировать значения можно в числовых полях и в полях, имеющих денежный формат, а значения в текстовых полях нельзя);

– нужна ли сортировка или индексирование поля;

– необходимо ли использование полей в группировке записей в запросах или отчетах;

– каким образом должны быть отсортированы значения в поле. Числа в текстовых полях сортируются как строки цифр (1, 10, 100, 2, 20, 200 и т. д.), а не как числовые значения. Для сортировки чисел как числовых значений используются числовые поля или поля, имеющие денежный формат. Также многие форматы дат невозможно отсортировать надлежащим образом, если они

были введены в текстовое поле. Для обеспечения сортировки дат используется поле типа Дата/время;

– необходимо ли хранение таких данных, как документы Microsoft Word и Microsoft Excel, рисунки, звуковые объекты и другие виды двоичных объектов, созданных в других программах.

Для хранения данных в виде текста или комбинации текста и цифр в Microsoft Access существует два типа данных для полей: текстовые и **поля MEMO**.

Текстовый тип данных используется для хранения таких данных, как имена, адреса, а также чисел, не требующих вычислений, например, номеров телефонов, инвентарных номеров или почтовых индексов. В текстовом поле может находиться до 255 знаков. По умолчанию устанавливается размер поля 50 знаков. Свойство **Размер поля** определяет максимальное количество знаков, которые можно ввести в текстовое поле.

Тип данных Поле MEMO используется для хранения более 255 знаков. В поле MEMO может находится до 65536 знаков. В текстовых полях и полях MEMO могут храниться только введенные знаки, знаки пробелов для неиспользованных позиций в поле храниться не будут.

Можно отсортировать или сгруппировать текстовые поля и поля типа MEMO, но Microsoft Access использует только первые 255 знаков при сортировке и группировке Поля MEMO.

Для хранения числовых данных в Microsoft Access существует два типа данных для полей: числовой и денежный.

Числовые поля используются для хранения числовых данных, которые должны использоваться в математических вычислениях, за исключением денежных расчетов, а также вычислений, требующих высокой точности. Тип и размер значений, которые могут находиться в числовом поле, можно изменить в свойстве **Размер поля.** Например, в занимающее 1 байт на диске поле **Байт** допускается ввод только целых чисел без десятичных знаков от 0 до 255.

Денежный тип поля используют для предотвращения округления во время вычислений. В денежных полях обеспечивается 15 знаков слева от десятичной запятой и 4 знака справа. Денежное поле занимает 8 байт на диске.

Числовые и денежные поля имеют стандартный формат отображения, также можно создать собственный формат.

Тип данных **Счетчик** обеспечивает автоматическую вставку уникальных последовательных (увеличивающихся на 1) или случайных чисел при добавлении записи. Сохраняет 4 байта.

Тип данных **Логический** – данные, принимающие только одно из двух возможных значений, таких как **Да/Нет**, **Истина/Ложь**, **Вкл/Выкл**. Значения **Null** не допускаются. Сохраняет 1 бит.

В **поле объекта OLE** хранятся документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звукозапись или другие данные в двоичном формате, созданные в других программах, использующих протокол OLE. Сохраняет до 1 Гигабайта (ограничивается объемом диска).

Гиперссылка может иметь вид пути UNC либо адреса URL. Сохраняет до 64 000 знаков.

Мастер подстановок создает поле, позволяющее выбрать значение из другой таблицы или из списка значений, используя поле со списком. При выборе данного параметра в списке типов данных запускается мастер для автоматического определения этого поля. Для сохранения требуется тот же размер, что и у первичного ключа, соответствующего полю подстановок, – обычно 4 байта.

Тип элемента управления – свойство, которое задается на вкладке Подстановка в окне конструктора таблиц. Стандартный тип элемента управления можно указать для полей с типами данных **Текстовый**, **Числовой** и **Логический**. Для текстовых и числовых полей стандартным типом элемента управления может быть поле, поле со списком или список (только в формах). Для логических полей стандартным типом элемента управления может быть флажок, поле или поле со списком. Когда для поля задан стандартный тип элемента управления, Microsoft Access использует этот тип для отображения данных из поля в режиме таблицы. Кроме того, этот тип элемента управления создается автоматически при добавлении поля в форму или отчет.

При формировании структуры таблицы необходимо задать ключевое поле. Для этого необходимо выделить соответствующее поле, щелкнуть правой клавишей мыши и выбрать пункт Ключевое поле (рис. 3.3).

После формирования структуры таблиц создается схема данных, в которой устанавливаются связи между таблицами.

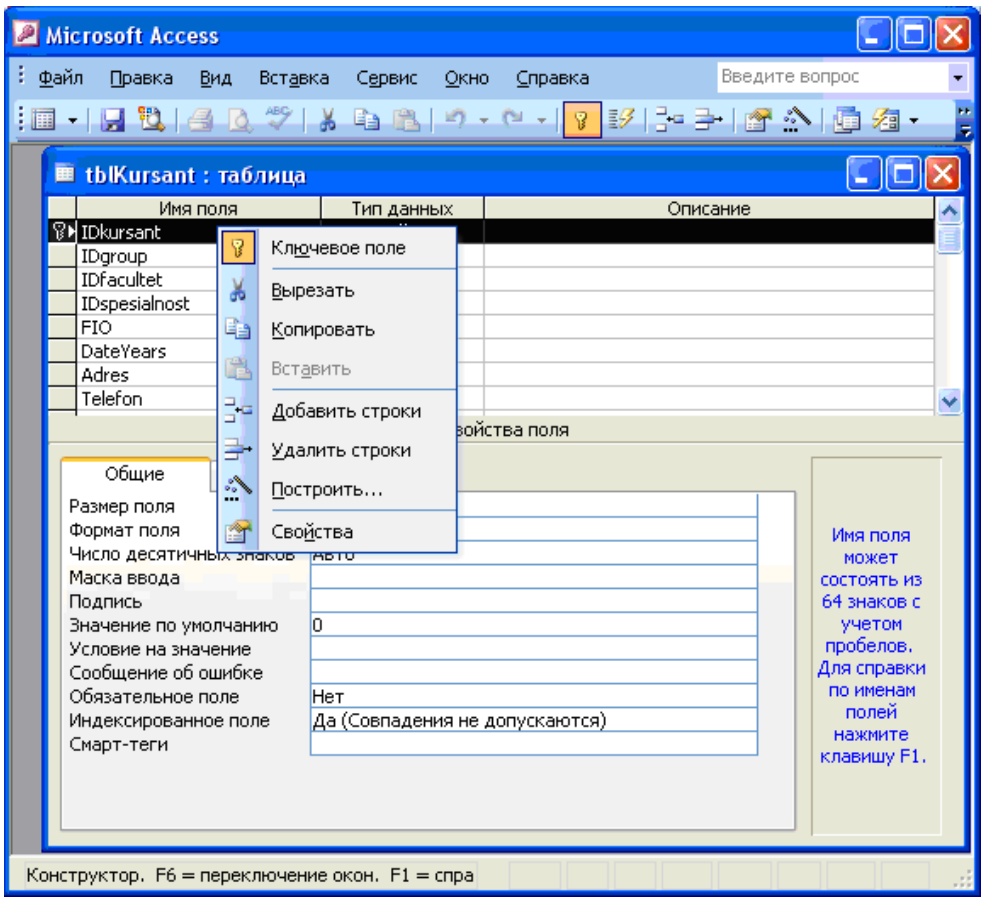

Рис. 3.3. Установка ключевого поля таблицы

Для создания схемы данных необходимо последовательно щелкнуть пункты меню Сервис → Схема данных или нажать кнопку  на панели инструментов, в результате откроется окно **Схема данных**. На свободном месте щелкнуть провой клавишей мыши для открытия окна **Добавление таблицы** (рис. 3.4).

При установлении связей можно задать параметр обеспечения целостности данных, а также автоматическое каскадное обновление и удаление связанных записей.

Обеспечение целостности данных означает выполнение следующих условий корректировки данных:

– в подчиненную таблицу не может быть добавлена запись с несуществующим в главной таблице значением первичного ключа;

– в главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчиненной таблице;

– изменение значений первичного ключа главной таблицы должно приводить к изменению соответствующих значений в записях подчиненной таблицы.

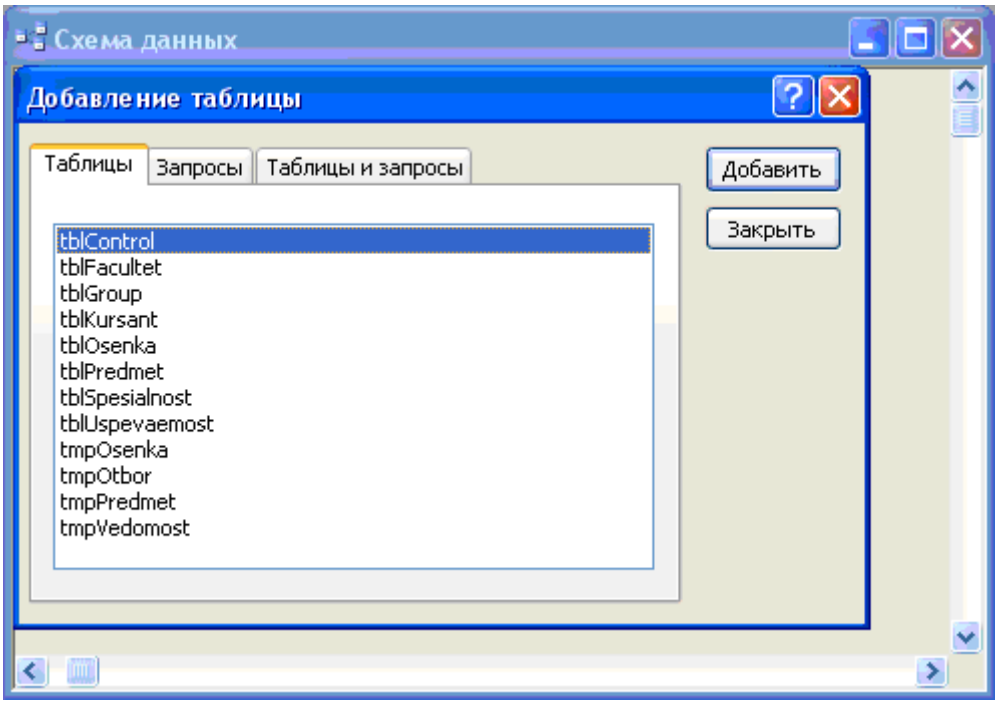

Рис. 3.4. Окно добавления таблицы в схему данных

При установлении связей между таблицами необходимо иметь в виду следующее:

– связываемые поля должны иметь одинаковый тип данных (имена полей могут быть различными, но рекомендуется называть их одинаково);

– главная таблица связывается с подчиненной по первичному ключу.

Если установлена связь с обеспечением целостности данных, Access автоматически отслеживает названный параметр. При вводе некорректных данных в связанные таблицы выводится соответствующее сообщение.

При установке связи с обеспечением целостности данных можно задать режимы **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей**.

В случае каскадного обновления связанных полей при изменении значения в поле связи главной таблицы Access автоматически изменяет значение в соответствующем поле в подчиненной таблице.

При каскадном удалении связанных полей при удалении записи из главной таблицы автоматически удаляются все связанные записи в подчиненных таблицах.

При установлении связей используется разработанная логическая модель. Для установления связи необходимо выделить уникальное ключевое поле в главной таблице и при нажатой кнопке мыши переместить курсор в соответствующее поле подчиненной таблицы. При установлении связи по составному

ключу необходимо выделить все поля, входящие в составной ключ главной таблицы и перетащить их на одно из полей в подчиненной таблице.

После установления связи открывается окно **Изменение связей**, при этом в строке Тип отношения должен установиться тип **один-ко-многим** (рис. 3.5). При этом в окне **Изменение связей** можно задать параметры **Обеспечение целостности данных**, **Каскадное обновление связанных полей**, **Каскадное удаление связанных полей**.

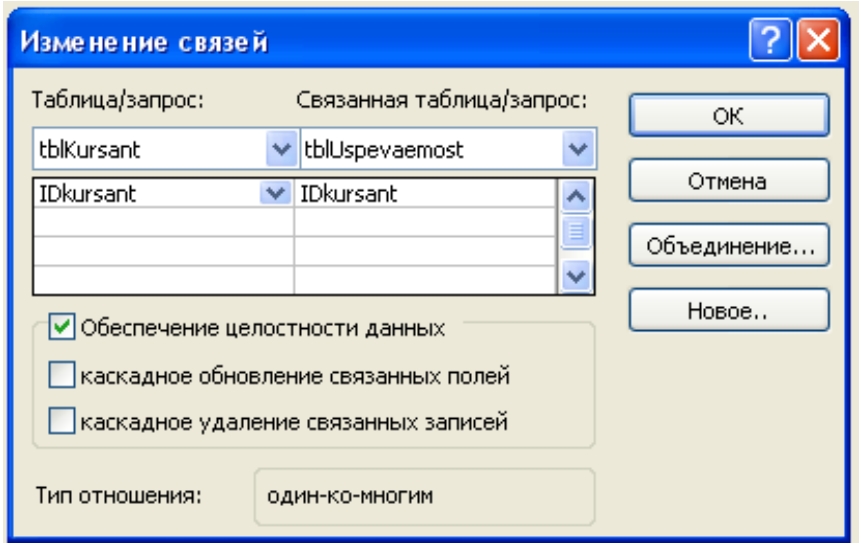

Рис. 3.5. Диалоговое окно при создании связей между таблицами

После создания таблиц и установки связей между ними разрабатываются формы для работы с данными.

4. РАЗРАБОТКА ФОРМ ДЛЯ ЗАГРУЗКИ, ПРОСМОТРА И КОРРЕКТИРОВКИ ДАННЫХ

По окончании формирования схемы данных разрабатывается графический интерфейс пользователя. Одним из важнейших инструментов являются формы, которые позволяют осуществлять первоначальную загрузку записей в таблицы базы данных, просматривать записи, производить корректировку данных. Формы, как правило, соответствуют формам первичных документов.

Перед конструированием форм целесообразно выполнить подготовительную работу для определения последовательности заполнения базы данных. При

разработке форм следует придерживаться определенных требований к последовательности загрузки:

– независимо могут загружаться таблицы, которые не являются подчиненными другим таблицам;

– подчиненные таблицы могут загружаться либо одновременно с главными таблицами, либо после загрузки главных таблиц.

При конструировании формы необходимо определить, из каких таблиц отображаются данные, нужны ли вычисляемые поля, графические элементы, служащие для ее оформления (линии, поясняющий текст, рисунки).

4.1. Порядок создания форм для ввода данных

СУБД Access позволяет организовать удобный и интуитивно понятный интерфейс пользователя для работы с данными с помощью форм. Формами называются настраиваемые диалоговые окна, сохраняемые в базе данных в виде объектов специального типа. Формы Access являются объектами базы данных, так же как таблицы и запросы. Формы используются в приложении для ввода и отображения данных. Формами можно управлять программно с помощью процедур на языке VBA.

Формы предоставляют более удобный способ просмотра и правки данных в таблицах, чем режим **Таблицы**. Формы содержат так называемые э*лементы управления*, с помощью которых осуществляется доступ к данным в таблицах. Элементами управления являются текстовые поля для ввода и правки данных, кнопки, флажки, переключатели, списки, надписи, а также рамки объектов для отображения графики. Создание форм, содержащих необходимые элементы управления, существенно упрощает процесс ввода данных и позволяет предотвратить ошибки.

Формы Access предоставляют функциональные возможности для выполнения многих задач, которые нельзя выполнить другими средствами. Формы позволяют выполнять проверку корректности данных при вводе, проводить вычисления и обеспечивают доступ к данным в связанных таблицах с помощью подчиненных форм.

4.2. Автоматическое создание формы на основе таблицы или запроса

Access предлагает несколько способов создания форм. Самым простым из них является использование средств автоматического создания форм на основе таблицы или запроса. Автоматически создаваемые формы (автоформы) бывают нескольких видов, каждый из которых отличается способом отображения данных:

– автоформа, организованная *«в столбец»*. В такой форме поля каждой записи отображаются в виде набора элементов управления, расположенных в один или несколько столбцов;

– *табличная*. Форма будет выглядеть так же, как обычная таблица Access;

– *ленточная*. В такой форме поля каждой записи располагаются в отдельной строке. Это очень удобно для работы с большими массивами данных, поскольку данные располагаются в таком же порядке, как в простой таблице. Преимуществом именно этого представления формы по сравнению с табличным является то, что каждое поле представлено в виде отдельного элемента управления, которое можно оформить в любом стиле и для которого можно определить функции обработки событий, т. е. «оживить» поле с помощью программирования;

– автоформа в виде *сводной таблицы* или *сводной диаграммы*.

Чтобы создать форму с помощью средства автоматического создания форм:

1. Щелкните по ярлыку **Формы** в окне **База данных** и нажмите кнопку Создать. Появится диалоговое окно **Новая форма**, представленное на рис. 4.1.

2. В списке диалогового окна **Новая форма** выделите один из вариантов автоформы, например: **Автоформа: в столбец**.

3. В поле со списком, находящимся в нижней части диалогового окна **Новая форма**, содержатся имена всех таблиц и запросов базы данных, которые могут быть использованы в качестве источника данных для формы. Щелкните левой кнопкой мыши по кнопке со стрелкой, чтобы раскрыть список, и выберите в нем нужный элемент.

4. Нажмите кнопку ОК.

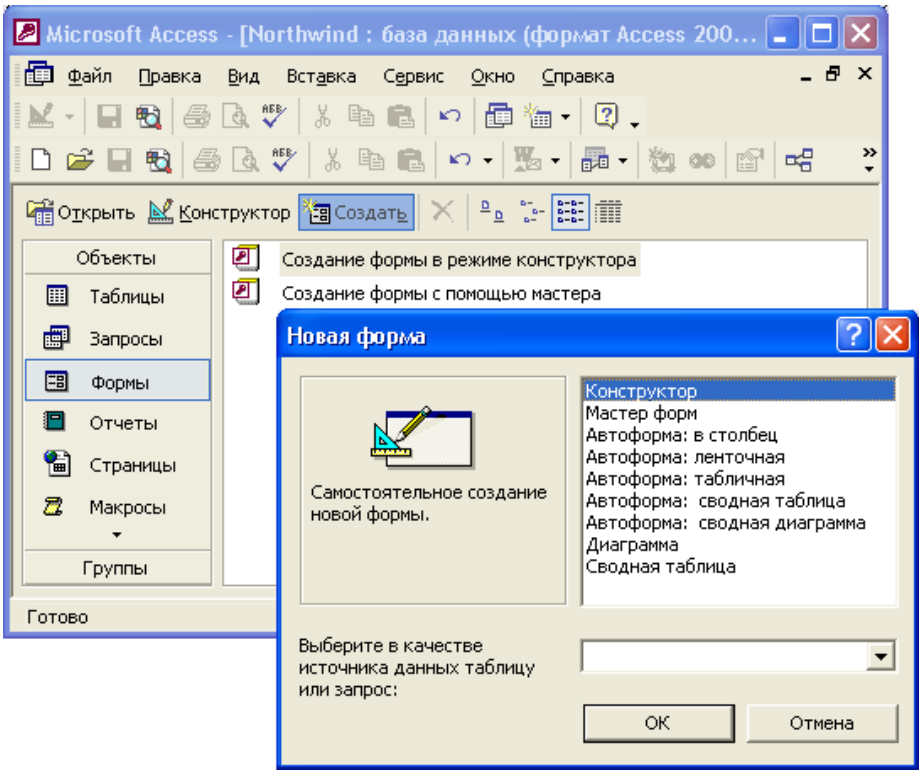

Рис. 4.1. Диалоговое окно **Новая форма**

В результате будет автоматически создана и открыта форма выбранного вида. Чтобы созданную форму можно было использовать в дальнейшем, ее необходимо сохранить. Для сохранения формы выберите команду Файл → Сохранить или нажмите на кнопку Сохранить на панели инструментов Режим формы. В поле **Имя формы** появившегося диалогового окна **Сохранение** введите нужное название и нажмите кнопку ОК.

Замечание. *Для любой формы можно изменить ее представление, определяющее режим отображения данных. В окне свойств формы можно выбрать один из пяти вариантов:* ***Одиночная форма***, ***Ленточные формы***, ***Режим таблицы***, ***Сводная таблица*** *и* ***Сводная диаграмма****. Чтобы изменить режим отображения данных, откройте форму в режиме* ***Конструктора****, затем откройте окно свойств формы и выберите соответствующий элемент в раскрывающемся списке* ***Режим по умолчанию****.*

4.3. Создание формы с помощью мастера

Другим способом создания формы является использование **Мастера форм**. С помощью него можно создавать формы на основе одной таблицы и более сложные формы на основе нескольких таблиц и запросов, имеющие подчиненные

формы. Намного проще и быстрее создавать формы с помощью мастера, а затем усовершенствовать их в режиме **Конструктора.**

Мастер форм разбивает процесс создания формы на несколько этапов. На каждом этапе требуется установить определенные параметры в одном из диалоговых окон Мастера, каждое из которых определяет один шаг создания формы. Если на каком-нибудь шаге была допущена ошибка или возникла необходимость изменения каких-либо установленных параметров, для возвращения к предыдущему шагу нажмите кнопку Назад. Кроме того, в любой момент можно нажать кнопку Отмена для отказа от создания формы и возвращения к окну базы данных.

В качестве примера рассмотрим процедуру создания формы **Курсант** для базы данных **Сессия**. Источником данных для этой формы будет таблица **tblKursant**. Эта форма предназначена для ввода и редактирования информации о курсантах (студентах). Чтобы с помощью мастера создать простую форму, не содержащую подчиненных форм:

1. Щелкните по ярлыку Формы в окне базы данных.

2. Нажмите кнопку Создать на панели инструментов окна базы данных. В списке вариантов в появившемся диалоговом окне **Новая форма** (см. рис. 4.1) выделите элемент **Мастер форм** и нажмите кнопку ОК. То же самое можно сделать, дважды щелкнув по ярлыку Создание формы с помощью мастера, находящемуся перед списком существующих форм в базе данных.

3. Появится первое диалоговое окно **Мастера форм** (рис. 4.2). В поле со списком **Таблицы и запросы**, как и в раскрывающемся списке в окне **Новая форма**, будут отображены имена всех таблиц и запросов базы данных, которые могут использоваться в качестве источника данных для формы. Раскройте этот список и выберите имя таблицы или запроса. В нашем примере это таблица **tblKursant.**

4. В списке **Доступные поля** этого диалогового окна отображаются все поля выбранной таблицы или запроса. Добавление полей в форму позволит просматривать и редактировать данные выбранной таблицы. Чтобы добавить в создаваемую форму только некоторые поля, выделите каждое из этих полей и нажмите кнопку «>». Выделенное поле будет перемещено из списка **Доступные поля** в список **Выбранные поля**. Чтобы добавить в создаваемую форму сразу все поля из выбранной таблицы или запроса, нажмите кнопку «>>». Нажмите кнопку Далее для отображения второго диалогового окна **Мастера форм.**

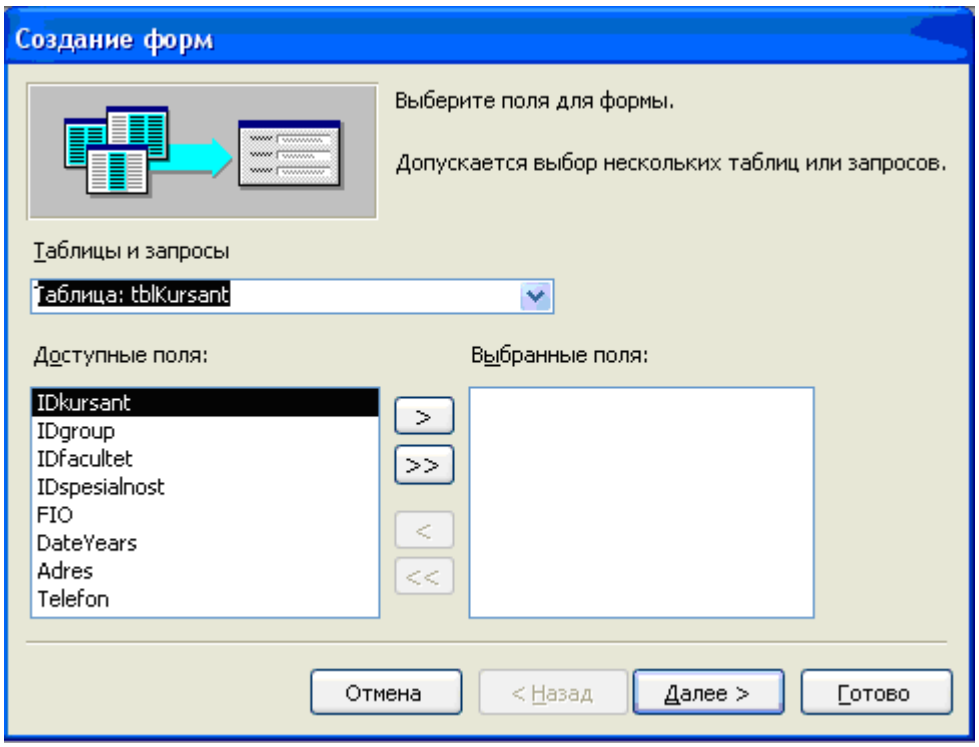

Рис. 4.2. Первое диалоговое окно **Мастера форм**

5. Во втором диалоговом окне мастера (рис. 4.3) можно определить вид формы. Существует несколько видов форм, определяющих представление данных на ней. Чтобы задать внешний вид формы, выберите один из переключателей: в один столбец, ленточный, табличный, выровненный, сводная таблица или сводная диаграмма.

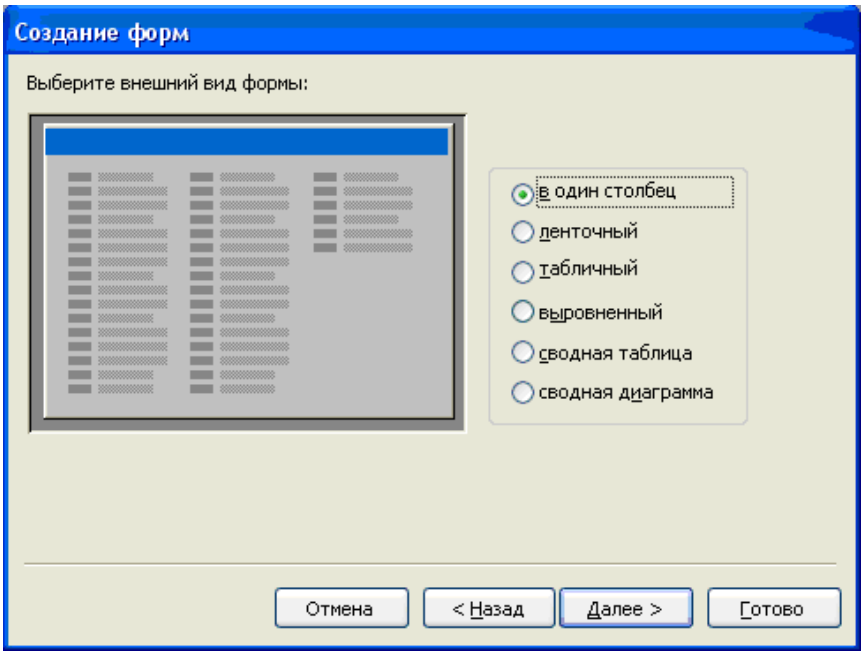

Рис. 4.3. Второе диалоговое окно **Мастера форм**

После выбора подходящего режима отображения данных в форме нажмите кнопку Далее для отображения следующего диалогового окна **Мастера форм** (рис. 4.4).

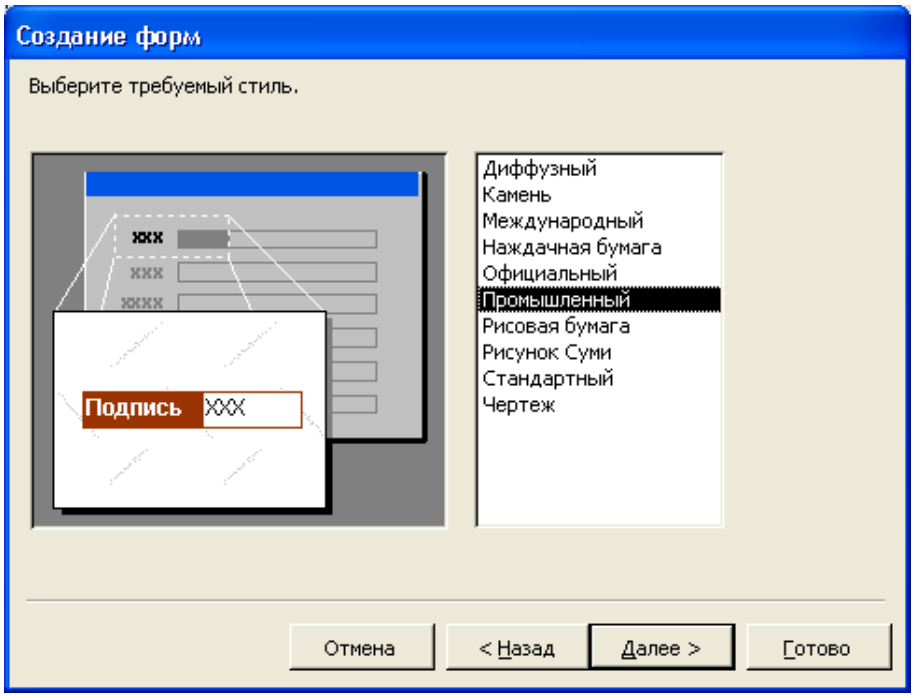

Рис. 4.4. Третье диалоговое окно **Мастера форм**

6. Третье диалоговое окно **Мастера форм** (рис. 4.4) предназначено для выбора стиля оформления новой формы. Мастер предлагает несколько стандартных стилей оформления. Можно определить собственные стили оформления форм с помощью диалогового окна **Автоформат**. Тогда эти стили будут отображаться вместе со стандартными в этом диалоговом окне **Мастера форм**. Выберите один из предлагаемых стилей и нажмите кнопку Далее.

7. В последнем диалоговом окне **Мастера форм** (рис. 4.5) требуется указать название формы. В поле ввода этого диалогового окна введите название формы: **ФормаКурсант**. Чтобы отобразить созданную мастером форму в режиме **Формы**, выберите переключатель Открыть форму для просмотра и ввода данных. Если после автоматического создания формы с помощью мастера требуется внести собственные изменения, выберите переключатель Изменить макет формы – тогда созданная форма будет открыта в режиме **Конструктора**. Если необходима справка о работе с созданной формой, установите флажок Вывести справку по работе с формой?, после чего нажмите кнопку Готово.

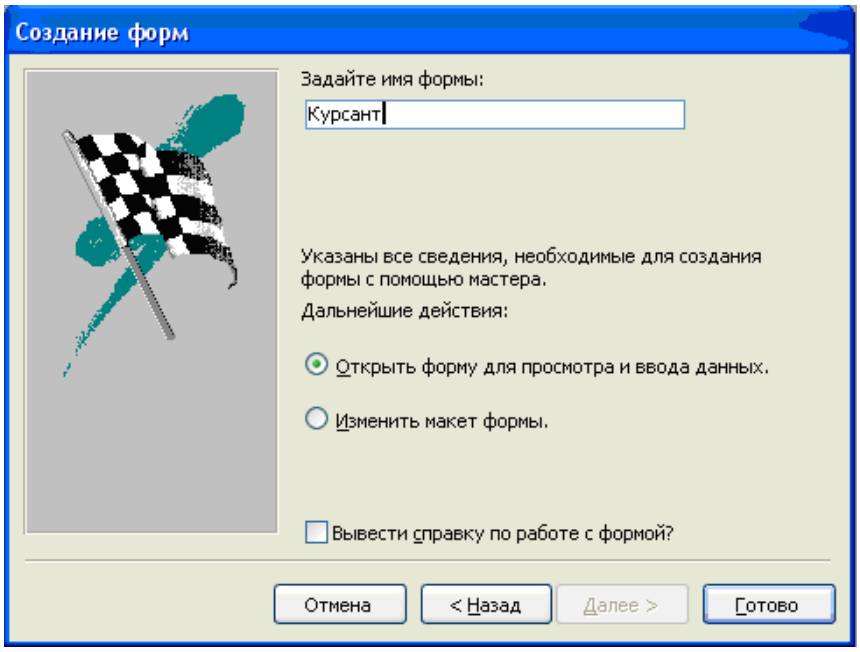

Рис. 4.5. Последнее диалоговое окно **Мастера форм**

В результате мастером будет создана форма в соответствии с выбранными параметрами и сохранена с указанным именем, затем эта форма будет открыта в заданном режиме. На рис. 4.6 представлена созданная форма, открытая в режиме **Формы**.

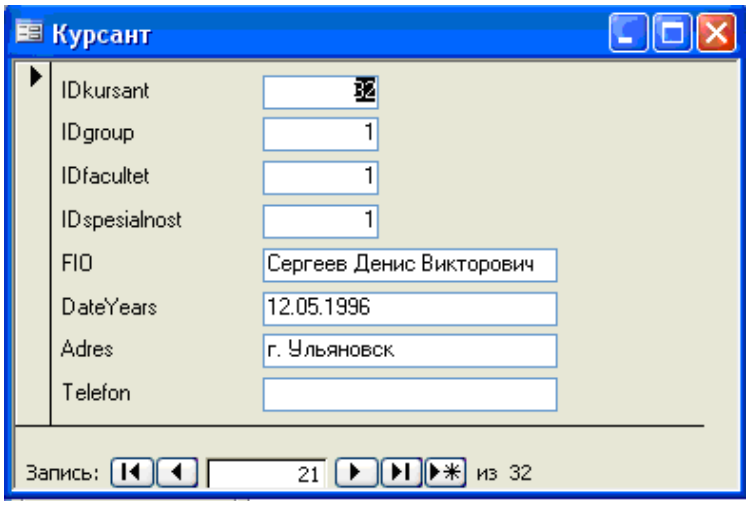

Рис. 4.6. Форма **Курсант**, созданная с помощью **Мастера форм**

Для уточнения текста надписей, размера, шрифта и других параметров элементов формы необходимо перейти в режим **Конструктора форм**. После выбора режима в окне Access появляется панель конструктора (рис. 4.7).

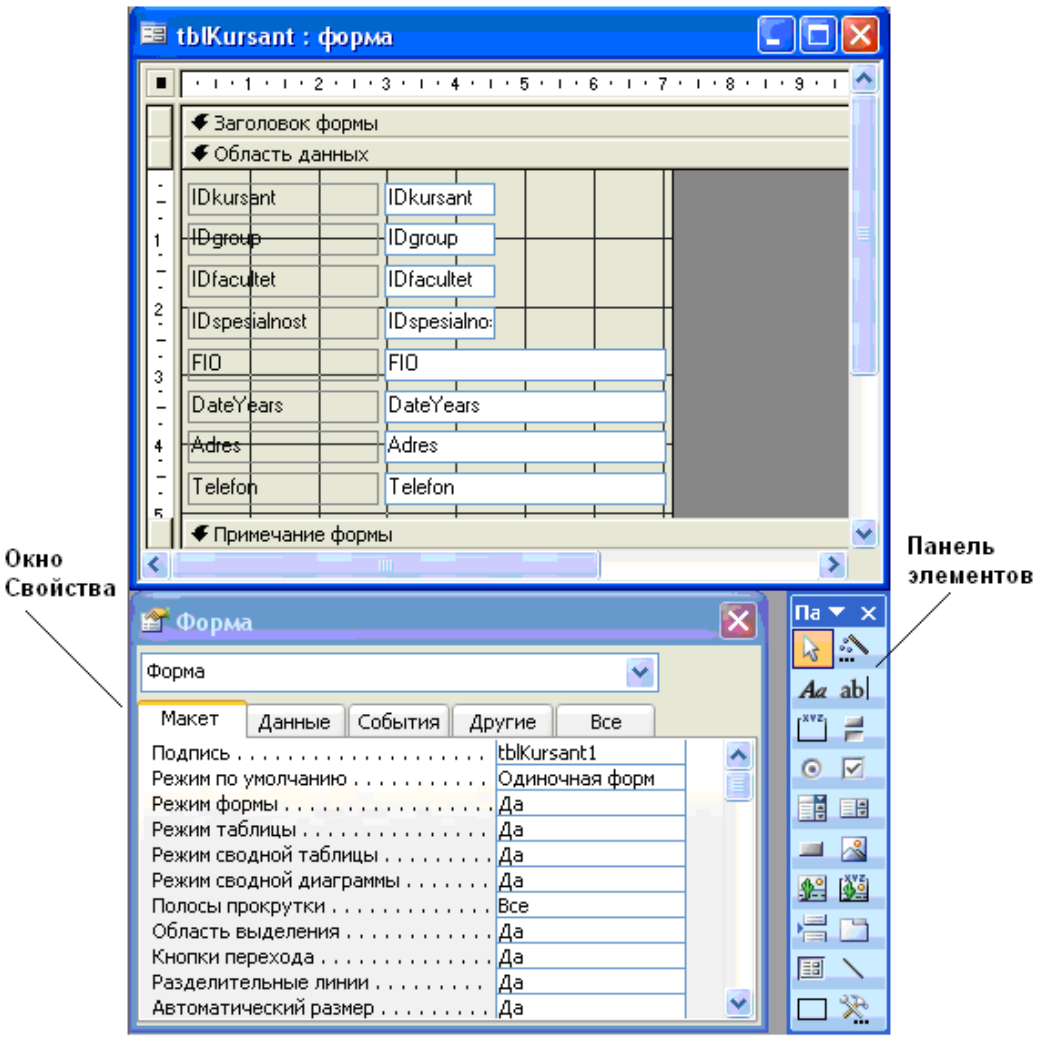

Рис. 4.7. Режим **Конструктора форм**

Для ввода текста заголовка в полученную форму необходимо расширить область заголовка, установив курсор мыши на границу области заголовка и области данных и перетащить границу на нужное расстояние.

Чтобы создать элемент **Надпись**, необходимо на панели элементов (рис. 4.7) щелкнуть мышью на кнопке Aa, затем в области заголовка формы, в окне **Свойства**, вкладка Макет выбрать нужный шрифт и другие параметры оформления.

4.4. Режимы работы с формами

Работа с формами Access может происходить в следующих режимах: в режиме **Формы**, в режиме **Таблицы**, в режиме **Конструктора**, в режиме **Сводной таблицы** и в режиме **Сводной диаграммы**. Выбрать режим работы можно либо с помощью кнопки Вид на панели инструментов текущего режима работы

с формой (например, **Конструктор форм** – одна из таких панелей), либо с помощью соответствующей команды меню Вид.

Режим Формы является «рабочим» для пользователя базы данных. В этом режиме осуществляются просмотр и редактирование записей, удаление записей или добавление новых. В этом же режиме по умолчанию открывается форма из окна базы данных. Если форма была открыта в другом режиме, то для перехода в режим **Формы** выберите команду Вид → Режим формы или нажмите на стрелку, расположенную справа от кнопки Вид на панели инструментов и в открывшемся списке выберите элемент **Режим формы**.

Для пользователя может оказаться удобным работать с формой в режиме **Таблицы**. В этом режиме, как и в режиме **Формы**, можно просматривать и редактировать, добавлять и удалять записи в таблице или запросе, являющемся источником данных для формы. Однако в этом режиме не применяются параметры форматирования элементов управления. Чтобы перейти в режим **Таблицы**, выберите команду Вид → Режим таблицы или нажмите на стрелку, расположенную справа от кнопки Вид панели инструментов и в открывшемся списке выберите элемент **Режим таблицы**. На рис. 4.8 показана форма, открытая в режиме **Таблицы**.

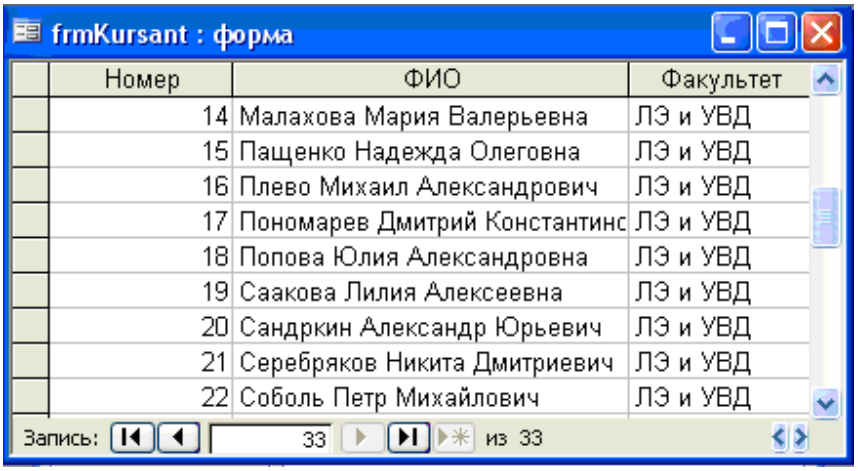

Рис. 4.8. Форма **Курсант** в режиме **Таблицы**

В любое время можно настроить различные свойства формы, изменить ее внешний вид, структуру и функциональность, работая с ней в режиме **Конструктора**. Чтобы перейти в режим **Конструктора**, выберите команду Вид → Конструктор или нажмите на стрелку, расположенную справа от кнопки Вид на панели инструментов, и в открывшемся списке выберите элемент **Конструктор**. На рис. 4.9 показана форма, открытая в режиме **Конструктора**.

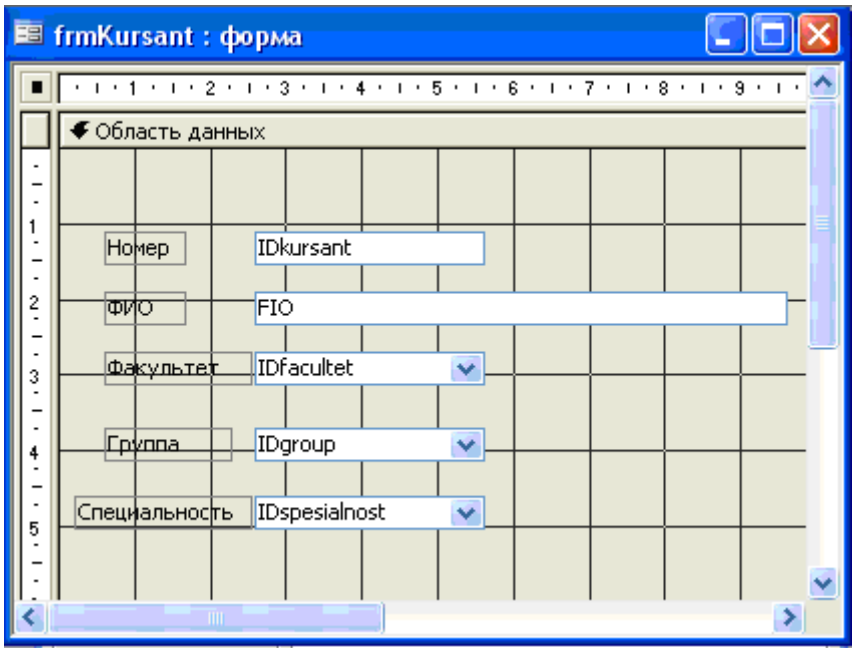

Рис. 4.9. Форма **Курсант** в режиме **Конструктора**

4.5. Печать формы

Форма, так же как и таблица, запрос или отчет базы данных Access, может служить документом для печати. Напечатать форму можно, находясь в любом режиме работы с ней, например, в режиме **Таблицы** или режиме **Формы**. При этом внешний вид печатаемой формы будет соответствовать текущему режиму. Команды печати и способ настройки параметров печати точно такие же, как при печати таблиц и отчетов Access или, например, документов Word. Это стандартные команды меню Файл (Параметры страницы, Предварительный просмотр и Печать), а также кнопки на панели инструментов текущего режима работы с формой и панели инструментов Предварительный просмотр. Печать формы, как и других объектов базы данных, можно выполнить и программным путем с помощью процедуры на VBA или макроса.

Чтобы увидеть, как форма будет выглядеть при печати, следует перейти в режим предварительного просмотра. Для этого выберите команду Файл → Предварительный просмотр или нажмите кнопку Предварительный просмотр на панели инструментов. Чтобы вернуться в прежний режим работы с формой, снова выберите команду меню Файл → Предварительный просмотр или нажмите кнопку Закрыть на панели инструментов. Наконец, можно нажать на стрелку справа от кнопки Вид на панели инструментов и выбрать один из режимов для дальнейшей работы с формой.

4.6. Режим Конструктора

Разработку структуры формы можно выполнить только в режиме **Конструктора**. В этом режиме можно изменить источник данных для формы, количество отображаемых полей, внешний вид формы и элементов управления, добавить или удалить элементы управления, настроить их свойства.

Панель инструментов Панель элементов, отображаемая в режиме **Конструктора**, позволяет создавать элементы управления, при этом некоторые элементы управления можно создавать с помощью **Мастера элементов**, осуществляющего пошаговое руководство этим процессом.

К форме можно применить один из предлагаемых Access стилей оформления для быстрой и качественной настройки ее внешнего вида. Более того, Access позволяет создавать свои собственные стили оформления и применять их к другим формам. Все это выполняется с помощью диалогового окна **Автоформат**.

Чтобы создать пустую форму, не пользуясь автоматическим созданием форм и **Мастером форм**, в окне базы данных щелкните по ярлыку Формы, нажмите кнопку Создать на панели в окне базы данных, в появившемся диалоговом окне **Новая форма** выделите элемент Конструктор и нажмите кнопку ОК. То же самое можно сделать, просто дважды щелкнув по ярлыку Создание формы в режиме **Конструктора**, находящемуся перед списком существующих форм в базе данных.

При работе в режиме **Конструктора форм** используются следующие элементы окна приложения:

– панель инструментов Панель элементов. При переходе в режим **Конструктора** на экране отображается эта панель инструментов, предназначенная для добавления в форму новых элементов управления. Для отображения нажмите кнопку Панель элементов на панели инструментов Конструктор форм или кнопку Закрыть – это кнопка с изображением крестика в правом верхнем углу панели инструментов;

– панель инструментов Конструктор форм. Основные команды, меню, используемые в режиме **Конструктора форм**, продублированы в виде кнопок на панели инструментов Конструктор форм;

– панель инструментов Формат (форма/отчет) предназначена для форматирования выделенных элементов управления. Форматирование элемента управления включает такие действия, как задание цвета фона и рамок, задание стиля и цвета текста;

– вертикальная и горизонтальная линейки предназначены для разметки формы по заданным размерам и выделения нескольких соседних элементов управления, лежащих в одном вертикальном или горизонтальном сегменте макета формы от одной отметки на линейке до другой;

– правая граница макета формы позволяет задать ширину формы. Чтобы изменить ширину формы, перетащите правую границу макета формы с помощью мыши вправо или влево;

– нижняя граница макета формы позволяет определить высоту формы. Чтобы изменить высоту формы, перетащите нижнюю границу макета формы с помощью мыши вниз или вверх;

– вертикальная и горизонтальная полосы прокрутки позволяют просматривать части формы, оказавшиеся за границами экрана в режиме **Конструктора.**

4.7. Структура формы

Макет формы состоит из разделов. Любая форма может включать следующие разделы:

– раздел **Заголовок формы** определяет верхнюю часть формы. Этот раздел добавляется в форму вместе с разделом примечания формы. В область заголовка формы можно поместить текст, графику и другие элементы управления. При печати многостраничной формы раздел заголовка отображается только на первой странице;

– раздел **Верхний колонтитул** определяет верхний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим нижний колонтитул страницы, и отображается только тогда, когда форма открыта в режиме **Предварительного просмотра**. При печати многостраничной формы верхний колонтитул отображается вверху каждой страницы;

– раздел **Область данных** определяет основную часть формы, содержащую данные, полученные из источника. Данный раздел может содержать элементы управления, отображающие данные из таблиц и запросов, а также неизменяемые данные, например, надписи. При печати многостраничной формы этот раздел отображается на каждой странице;

– раздел **Нижний колонтитул** определяет нижний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим верхний колонтитул страницы. Он отображается только тогда, когда

форма открыта в режиме Предварительного просмотра. При печати многостраничной формы нижний колонтитул отображается внизу каждой страницы;

– раздел **Примечание формы** определяет нижнюю часть формы. Этот раздел добавляется в форму вместе с разделом заголовка формы. При печати многостраничной формы примечание формы будет отображено только внизу последней страницы.

Разделы формы в режиме **Конструктора** представлены на рис. 4.10.

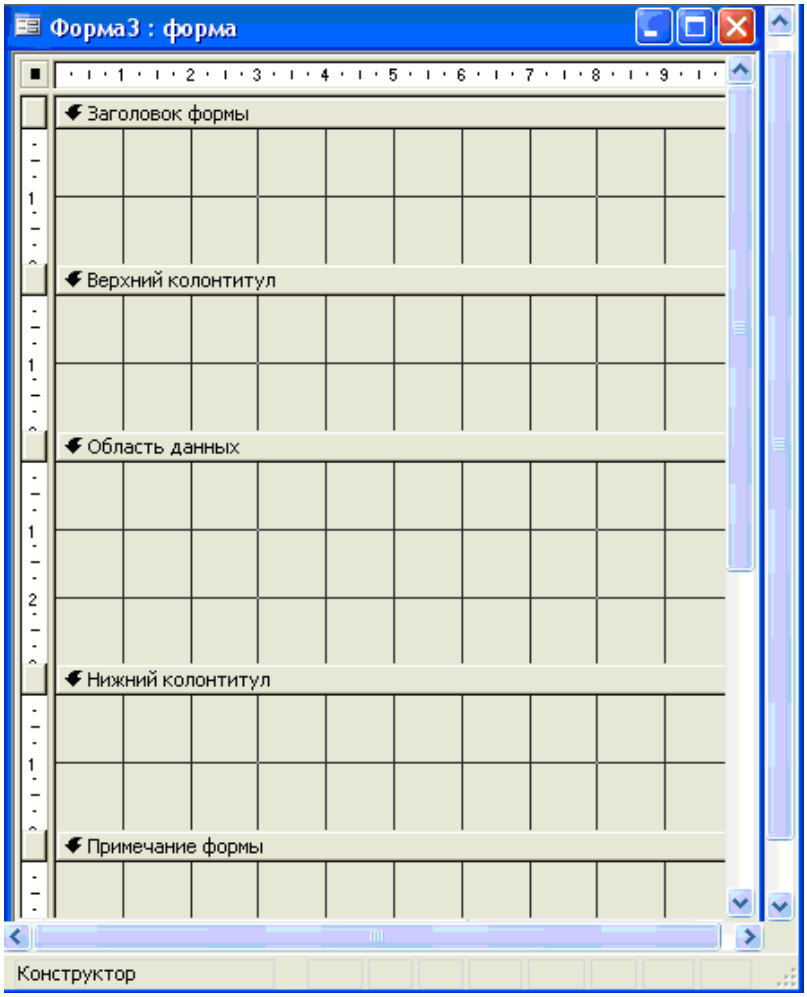

Рис. 4.10. Разделы формы Access

Для добавления или удаления разделов заголовка и примечания формы выберите команду Вид → Заголовок/примечание формы. Если форма уже содержит эти разделы, то команда Заголовок/примечание формы будет отмечена флажком. Для удаления разделов заголовка и примечания формы сбросьте этот флажок.

Для добавления или удаления разделов верхнего и нижнего колонтитулов страницы выберите команду Вид → Колонтитулы. Если форма уже содержит эти

разделы, то команда Колонтитулы будет отмечена флажком. Для удаления разделов верхнего и нижнего колонтитула страницы сбросьте этот флажок.

Высоту раздела формы можно изменить, перетаскивая границу раздела при помощи мыши.

4.8. Параметры работы с формами

При создании новых форм используются параметры, заданные с помощью диалогового окна **Параметры**. Для отображения окна параметров выберите команду Сервис → Параметры и в появившемся диалоговом окне раскройте вкладку Формы и отчеты (рис. 4.11). С помощью этой вкладки можно задать имя существующей формы (в текстовом поле **Шаблон формы**), которая будет использована в качестве образца оформления новых форм, исключая формы, созданные с помощью **Мастера форм**. Можно также задать способ выделения объектов в форме или отчете, выбрав соответствующий элемент в группе **Выделение объектов**, и назначить использование по умолчанию процедур VBA для обработки событий форм, элементов управления или отчетов, установив флажок Всегда использовать процедуры обработки событий.

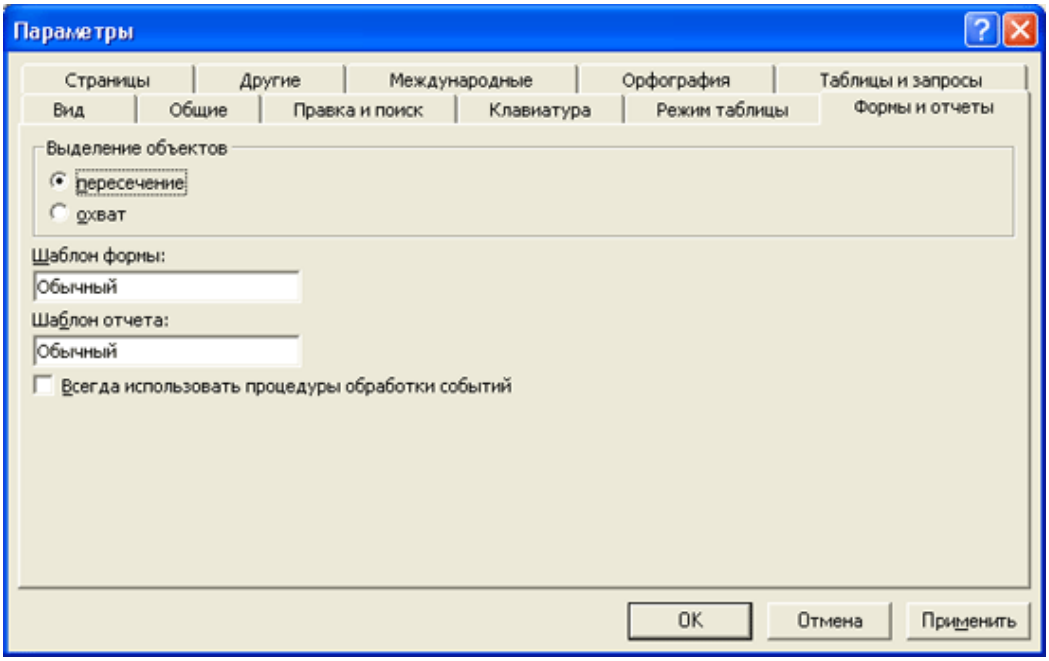

Рис. 4.11. Вкладка **Формы** и отчеты диалогового окна **Параметры**

4.9. Основные элементы управления

Чтобы сделать форму более понятной и легко читаемой, в нее добавляют заголовки, подписи или пояснения. Этот текст является неизменяемым и создается с помощью элемента управления **Надпись.**

Для отображения, ввода или изменения в форме текстовых данных, например, примечаний, используют текстовые поля, которые создаются с помощью элемента управления **Поле.**

Текстовые поля позволяют вводить произвольные значения (хотя можно ограничить множество допустимых значений для поля, задав с помощью окна свойств текстового поля, например, маску ввода или правило проверки введенного значения). Чтобы предоставить пользователю выбор из определенного набора значений, вместо текстовых полей применяют такие элементы управления, как флажки, переключатели, выключатели, объединенные в группы однотипных элементов, а также списки.

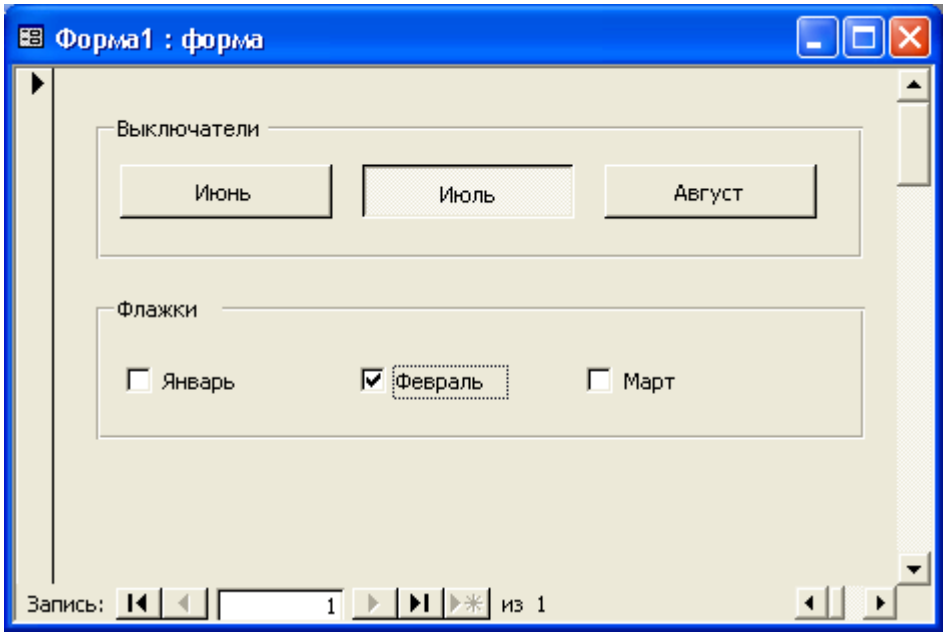

Рис. 4.12. Отдельные флажки, переключатели, выключатели и группы элементов

Группа (рис. 4.12) тоже представляет собой элемент управления, который имеет название **Группа переключателей**. С каждым элементом управления в группе сопоставляется уникальное число, однозначно определяющее данный элемент в группе, и это число сохраняется в качестве значения свойства **Значение элемента управления**. Отдельно стоящий элемент управления типа

Флажки, **Переключатели** или **Выключатели**, не включенный в группу подобных элементов, определяет значение логического типа: **Да** или **Нет**, в данном случае вместо значения **Да** допускается использовать значение **Истина** или любое положительное число, а вместо значения **Нет** – значение **Ложь** или **ноль**. Элементы управления **Флажки**, **Переключатели**, **Выключатели** и **Группа переключателей** имеют свойство **Значение по умолчанию**, позволяющее задать первоначальное состояние элемента управления или группы. Для отдельного элемента управления в качестве значения этого свойства используется значение логического типа, а для группы – число, сопоставленное с одним из элементов группы (с тем элементом, который будет выбран в группе по умолчанию).

Кроме групп элементов, для предоставления выбора значения какого-либо параметра из заданного набора можно использовать элементы управления **Список** или **Поле со списком**. Ввести значение в поле со списком можно двумя способами: ввести значение в поле или выбрать значение в раскрывающемся списке.

4.10. Создание элементов управления с помощью панели элементов

Надпись – самый простой для использования тип элементов управления панели элементов. Для добавления надписи в какой-либо раздел формы:

1. На панели элементов нажмите кнопку Надпись. Когда указатель мыши попадет в активную область формы, он примет вид крестика со значком элемента управления **Надпись**. Центр крестика определяет позицию верхнего левого угла элемента управления **Надпись**.

2. Поместите указатель мыши в виде крестика в область нужного раздела формы. Нажмите левую кнопку мыши и, удерживая ее, перетащите указатель мыши в нижний правый угол надписи (рис. 4.13).

Вместе с перемещением указателя мыши будет изменяться и контур надписи. Число строк и количество символов текущего типа шрифта, которые может отобразить надпись, выводятся в строке состояния.

3. При достижении элементом управления **Надпись** нужных размеров, отпустите левую кнопку мыши. Если надпись будет иметь размеры, превышающие область того раздела, к которому она добавляется, то область раздела формы будет увеличена для того, чтобы надпись поместилась целиком.

4. Внутри контура надписи появится текстовый курсор. Введите нужный текст. Если, создав надпись, не ввести в нее хотя бы одного символа, то после следующего щелчка кнопкой мыши этот элемент управления исчезнет.

После добавления элемента управления для его перемещения и изменения размеров используются угловой маркер перемещения и маркеры изменения размеров. Положение углового маркера перемещения определяет значения свойств элемента управления **От левого края** и **От верхнего края**. Маркеры изменения размеров устанавливают значения свойств элемента управления **Ширина** и **Высота**.

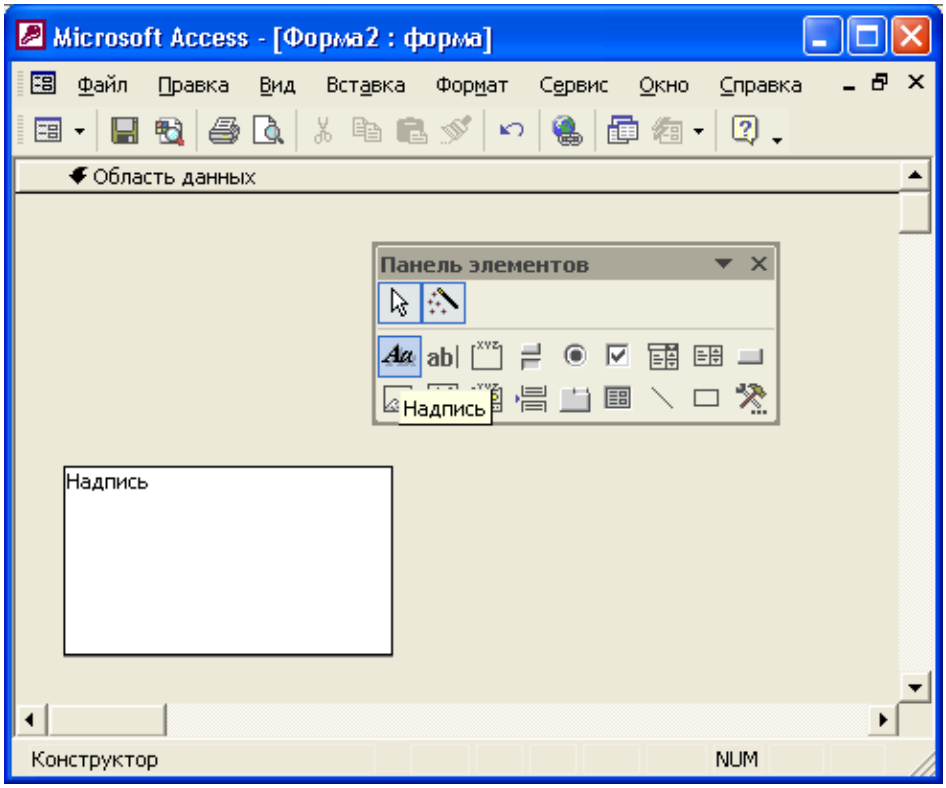

Рис. 4.13. Добавление в форму элемента управления **Надпись**

Для добавления большинства других типов элементов управления (выключателей, переключателей, флажков) в форму используется процесс, описанный выше. Некоторые кнопки панели элементов, например Список или Поле со списком, при создании элемента управления вызывают **Мастер элементов**, если кнопка **Мастера** нажата.

Чтобы создать элемент управления **Список** с помощью мастера:

1. На панели элементов нажмите кнопку Мастера, если она еще не нажата.

2. На панели элементов нажмите кнопку Список. Когда указатель мыши попадет в активную область формы, он примет вид крестика со значком элемента

управления **Список**. Центр крестика определяет позицию верхнего левого угла элемента управления **Список.**

3. Поместите указатель мыши в виде крестика в область нужного раздела формы. Нажмите левую кнопку мыши и, удерживая ее, перетащите указатель мыши в нижний правый угол списка.

Вместе с перемещением указателя мыши будет изменяться и контур списка. Число строк и количество символов текущего типа шрифта, которые может отобразить список, выводятся в строке состояния.

4. При достижении элементом управления **Список** нужных размеров отпустите левую кнопку мыши. Появится первое диалоговое окно **Мастера списков**. В этом диалоговом окне выберите переключатель Объект «список» будет использовать значения из таблицы или запроса. Нажмите кнопку Далее. Появится второе диалоговое окно **Мастера списков** (рис. 4.14).

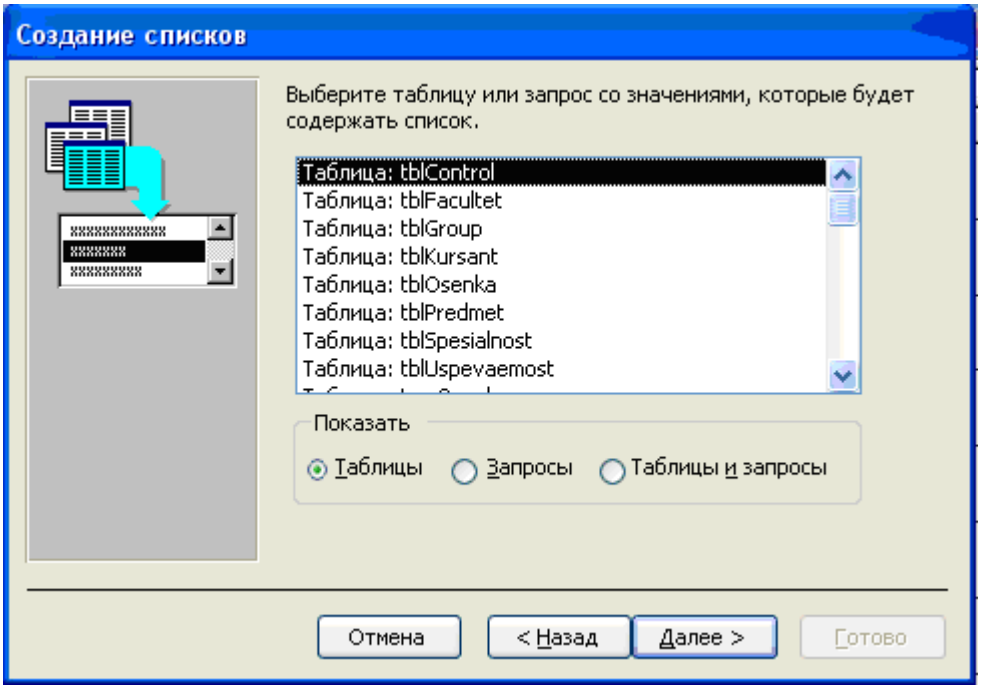

Рис. 4.14. Второе диалоговое окно **Мастера списков**

5. Во втором диалоговом окне **Мастера списков** нужно указать таблицу или запрос, значения из которого должны появляться в списке. Выделите в списке нужную таблицу или запрос и нажмите кнопку Далее. Появится третье диалоговое окно мастера (рис. 4.15).

6. В этом диалоговом окне нужно определить, какие поля таблицы (в нашем примере – таблицы **Факультет**) будут отображаться в списке. Чаще всего в

списке должно отображаться одно поле, значение из которого будет выбираться пользователем, но необходимо включить в список еще и ключевое поле, т. к. именно его значение будет использовано в качестве значения данного элемента управления. В нашем примере полей всего два. Поэтому достаточно нажать кнопку «>>», чтобы переместить их из списка доступных полей в список выбранных. Затем нажмите кнопку Далее.

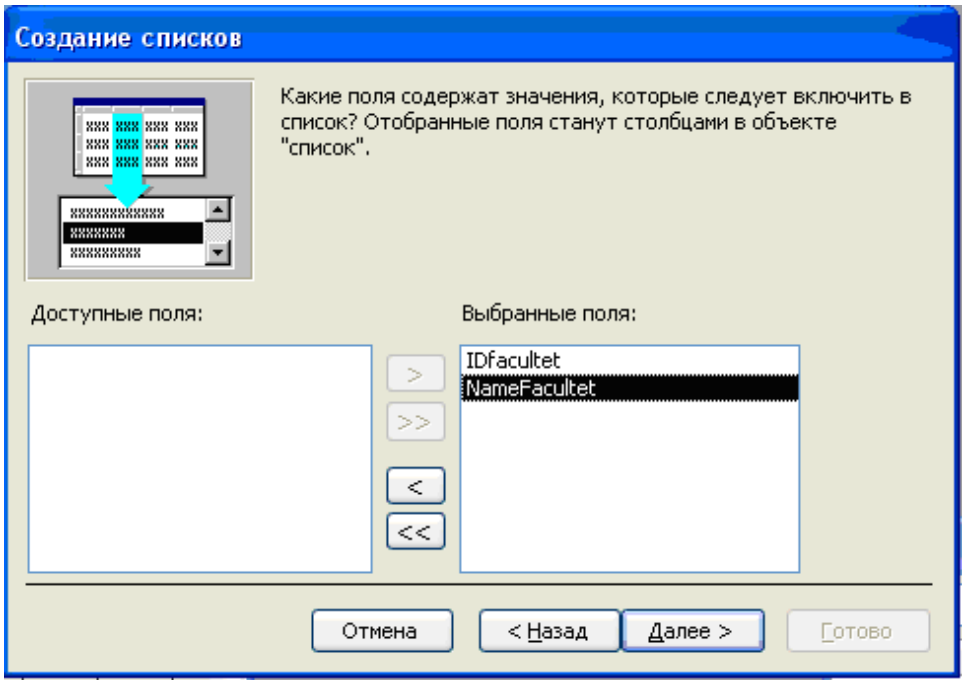

Рис. 4.15. Третье диалоговое окно **Мастера списков**

7. В следующем диалоговом окне (рис. 4.16) отображаются выбранные поля, причем поле **IDfakultet**, которое показывать пользователю не нужно, скрыто. Если ключевое поле содержит нужную пользователю информацию, его можно отобразить в списке (список может содержать два и более полей). Для этого достаточно сбросить флажок Скрыть ключевой столбец (рекомендуется). Перемещая с помощью мыши правую границу столбцов, можно задать ширину столбцов. Чтобы настроить ширину столбца по ширине самого длинного значения, достаточно подвести указатель мыши к правой границе столбца и дважды щелкнуть левой кнопкой. Нажмите кнопку Далее.

8. Если для формы задан источник записей: таблица или запрос, появится пятое диалоговое окно **Мастера списков**, представленное на рис. 4.17 (иначе отобразится сразу последнее диалоговое окно **Мастера списка**, описанное в следующем шаге). В этом диалоговом окне нужно указать, требуется ли сохранить выбранное значение списка в поле источника данных формы. Если требуется

(как в нашем примере), выберите переключатель Сохранить в поле. Тогда в раскрывающемся списке справа отобразятся все поля источника данных формы. В нашем примере этим источником данных является таблица **Факультет** и из списка нужно получить значение поля **IDfakultet**, поэтому необходимо выбрать в раскрывающемся списке имя поля **IDfakultet**. После этого нажмите кнопку Далее. Появится последнее диалоговое окно **Мастера списков.**

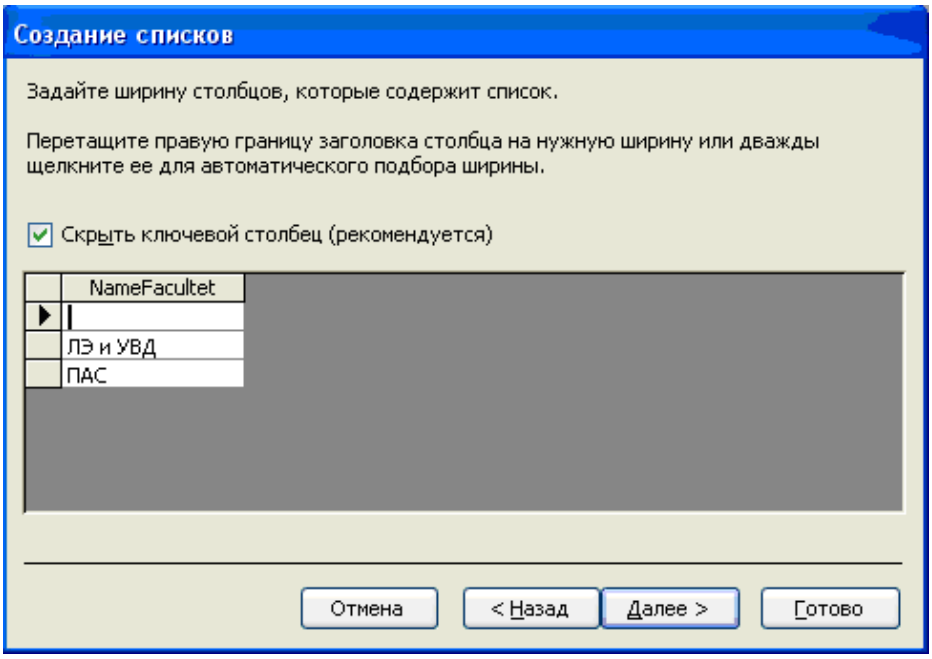

Рис. 4.16. Четвертое диалоговое окно **Мастера списков**

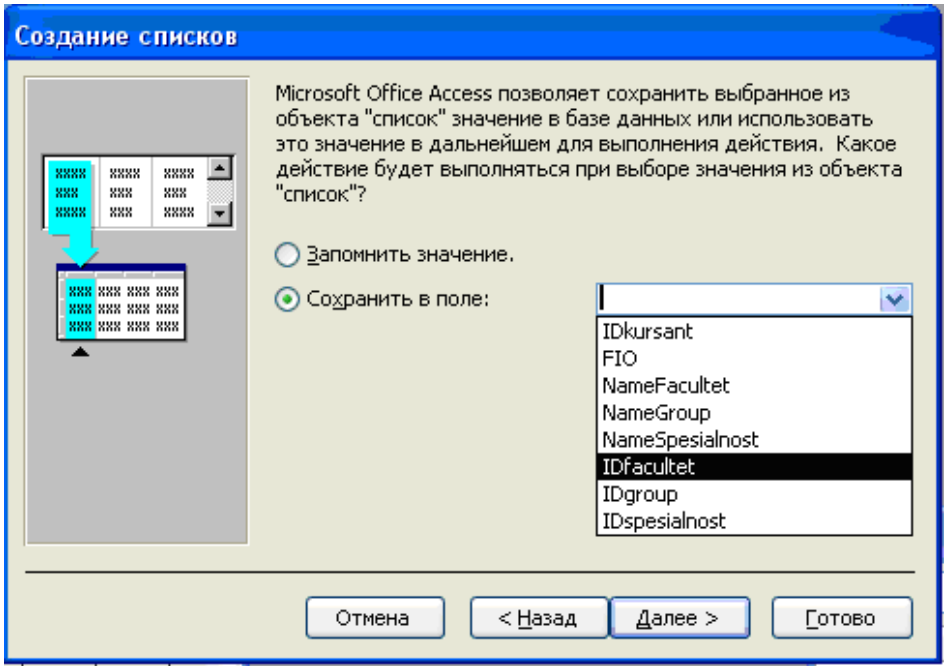

Рис. 4.17. Пятое диалоговое окно **Мастера списков**

9. В последнем диалоговом окне можно указать название, которое будет использоваться в качестве метки для создаваемого списка. Чтобы после создания списка получить справку по возможностям его настройки, установите флажок, расположенный внизу диалогового окна.

10. Для завершения процесса создания списка нажмите кнопку Готово.

При создании таких элементов управления без помощи **Мастера элементов** свойства создаваемых элементов нужно настраивать вручную с помощью окна свойств элемента управления. Чтобы открыть окно свойств какого-либо элемента управления формы, выделите его и нажмите кнопку Свойства на панели инструментов Конструктор форм либо дважды щелкните по элементу управления. На рис. 4.18 изображено окно свойств списка.

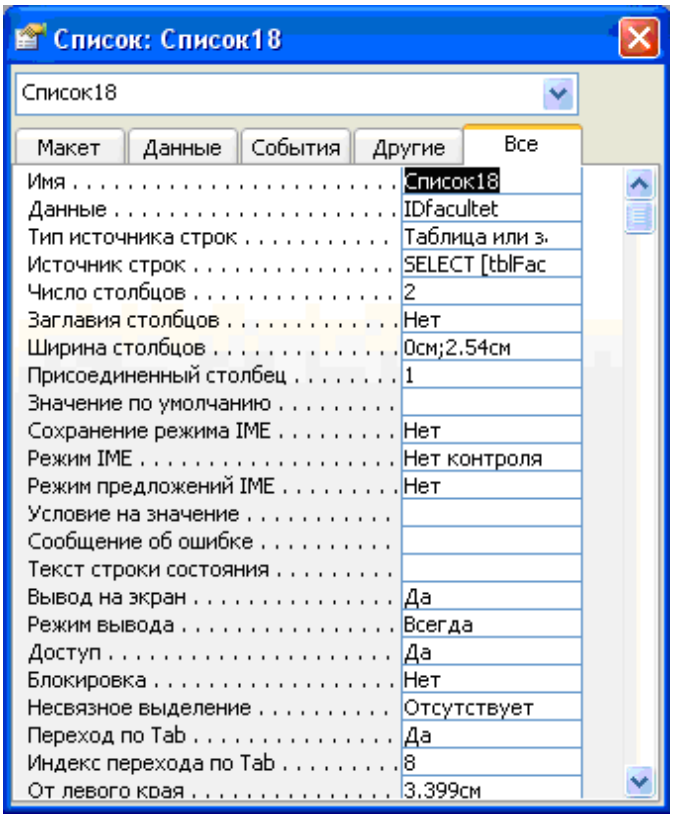

Рис. 4.18. Окно свойств списка

Еще одним достаточно простым и удобным способом создания элементов управления является использование списка полей. Последовательность действий при создании **Поля со списком** аналогична созданию списка.

Пример формы с использованием поля со списком приведен на рис. 4.19.

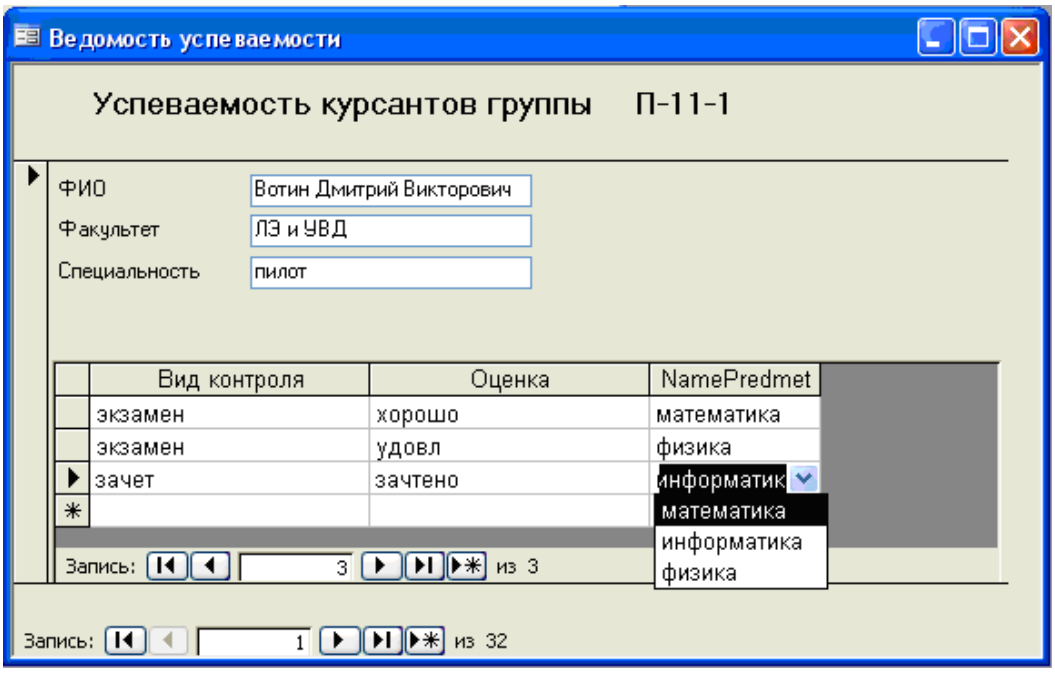

Рис. 4.19. Форма ввода и просмотра данных об успеваемости

Рассмотренные формы позволяют производить просмотр, добавление и удаление записей, обновление полей. Необходимо отметить, что одним из основных инструментов обработки данных в СУБД Access являются запросы, что и будет рассмотрено в следующем разделе.

5. ОБРАБОТКА ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ЗАПРОСОВ

Одним из основных инструментов обработки данных в СУБД Access являются запросы. Запрос строится на основе одной или нескольких таблиц, а также на основе другого запроса. Запрос позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц, произвести вычисления и получить результат в виде таблицы. Через запрос можно произвести обновление данных в таблицах, добавление и удаление записей.

Запрос позволяет:

– выбрать записи, удовлетворяющие условиям отбора;

– включить в результирующую таблицу запроса заданные пользователем поля;

– произвести вычисления в каждой из полученных записей;

– сгруппировать записи с одинаковыми значениями в одном или нескольких полях для выполнения над ними групповых функций;

– произвести обновление полей в выбранном подмножестве записей;

– создать новую таблицу базы данных, используя данные из существующих таблиц;

– удалить выбранное подмножество записей из таблицы базы данных;

– добавить выбранное подмножество записей в другую таблицу.

В Access может быть создано несколько видов запроса:

– запрос на выборку – выбирает данные из взаимосвязанных таблиц и других запросов. Результатом его является таблица, которая существует до закрытия запроса. Запрос на выборку играет особую роль, так как на его основе строятся запросы другого вида;

– запрос на создание таблицы – основан на запросе на выборку, но в отличие от него результат сохраняется в новой таблице;

– запросы на обновление, добавление, удаление – запросы действия, в результате выполнения которых изменяются данные в таблицах;

– перекрестные запросы – это запросы, в которых происходит статистическая обработка данных, результаты которой выводятся в виде таблицы. Перекрестные запросы используют для расчетов и представления данных в структуре, облегчающей их анализ. Перекрестный запрос подсчитывает сумму, среднее число значений или выполняет другие статистические расчеты, после чего результаты группируются в виде таблицы по двум наборам данных, один из которых определяет заголовки столбцов, а другой заголовки строк. Таким образом, структура перекрестного запроса напоминает структуру электронной таблицы. Если в перекрестный запрос нужно включить поля из нескольких таблиц и/или запросов, **сначала создается запрос на выборку, содержащий все необходимые поля**, а затем в мастере проектируется собственно перекрестный запрос.

5.1. Разработка запросов

Разработка запросов производится в режиме **Конструктора запросов** или с помощью **Мастера запросов**. Создание запросов с помощью мастера особых трудностей не представляет. Рассмотрим создание запросов в режиме **Конструктора**.

Для создания запроса в окне базы данных выбрать закладку Запрос и нажать кнопку Создание запроса в режиме конструктора. В окне **Добавление таблицы**

выбрать используемые в запросе таблицы и нажать кнопку Добавить. В результате появится окно **Конструктора запросов** (рис. 5.1). Окно **Конструктора запросов** разделено на две панели. Верхняя панель содержит схему данных запроса, нижняя панель является бланком запроса по образцу, который необходимо заполнить.

При заполнении бланка запроса необходимо:

– в строку **Поле** включить имена полей, используемых в запросе;

– в строке **Вывод** на экран отметить поля, которые должны быть включены в результирующую таблицу;

– в строке **Условие отбора** задать условия отбора записей;

– в строке **Сортировка** выбрать порядок сортировки записей.

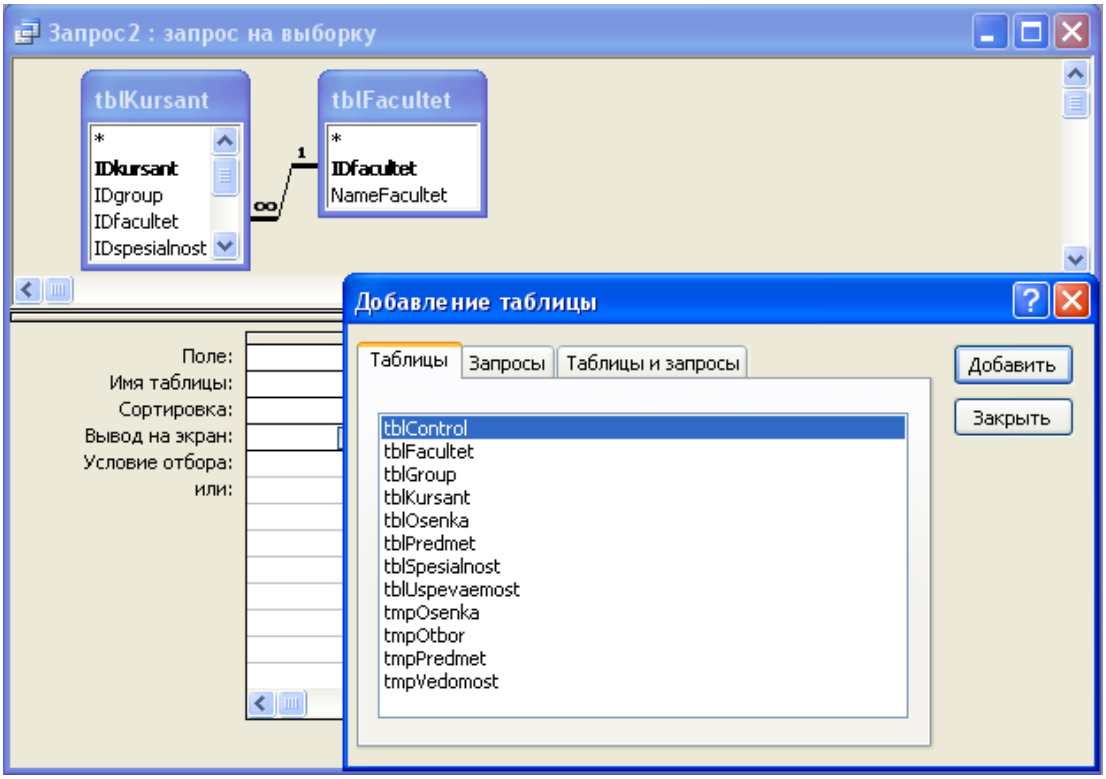

Рис. 5.1. Окно **Конструктора запросов**

Поля бланка запроса. Каждый столбец бланка запроса соответствует одному из полей таблиц, из которых строится запрос. Кроме того, здесь может размещаться вычисляемое поле.

Условие отбора. Условием отбора является выражение, которое состоит из операторов сравнения и операндов, используемых для сравнения.

Параметры запроса. Конкретное значение поля в условии отбора может вводиться непосредственно в бланк запроса или задаваться пользователем при

выполнении запроса в диалоговом окне. Чтобы выводилось диалоговое окно, необходимо определить параметр запроса.

В качестве примера создадим многотабличный запрос на выборку с целью получения информации об оценках, полученных курсантом (студентом) по всем предметам. Результат должен содержать фамилию курсанта (студента), наименование сданных предметов и оценки.

Порядок выполнения:

1. Сформировать новый запрос в режиме **Конструктора**.

2. Добавить необходимые таблицы (курсант (студент), успеваемость, предмет, оценка).

В результате формируется схема данных запроса, содержащая выбранные таблицы (рис. 5.2).

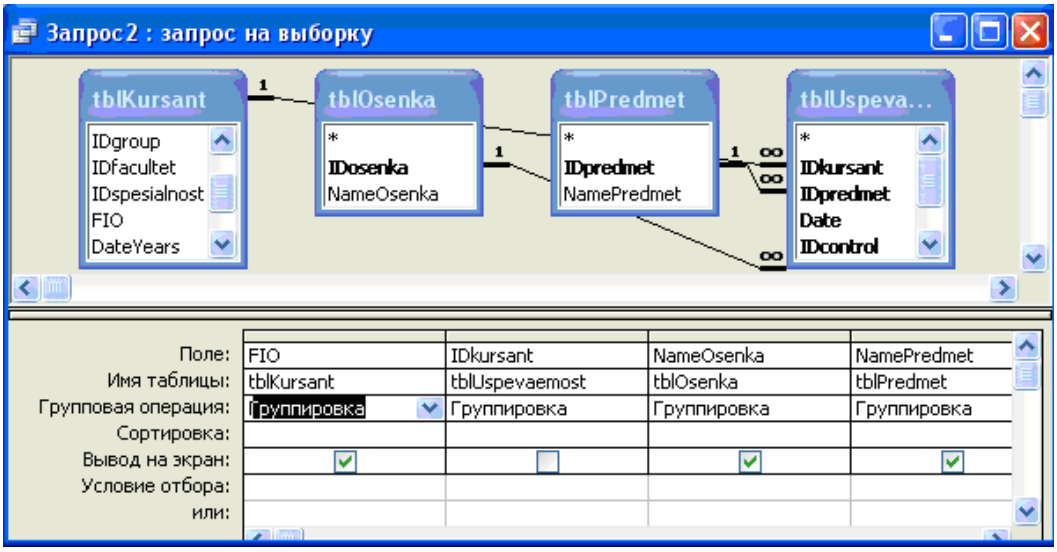

Рис. 5.2. Схема данных запроса

Между таблицами автоматически устанавливаются необходимые связи.

3. Выполнить запрос.

На рис. 5.3 представлены результаты выполнения запроса.

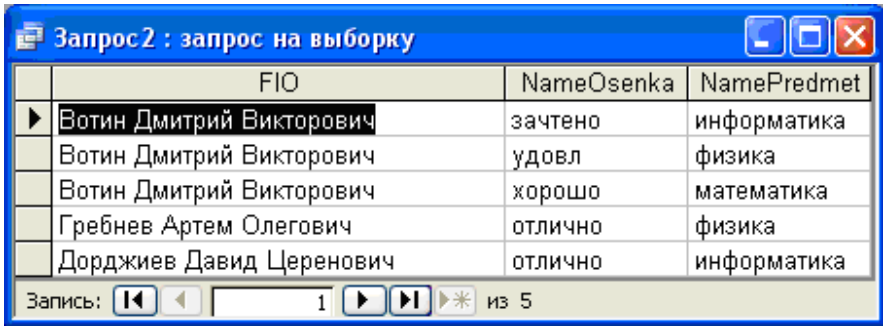

Рис. 5.3. Результаты выполнения запроса об успеваемости

Запрос на выборку в данном случае не информативен. На рис. 5.4 представлены данные об успеваемости, полученные с помощью перекрестного запроса, выполненного на основе запроса об успеваемости (см. рис. 5.3).

Можно сказать, что в данном случае предпочтительнее использовать перекрестный запрос.

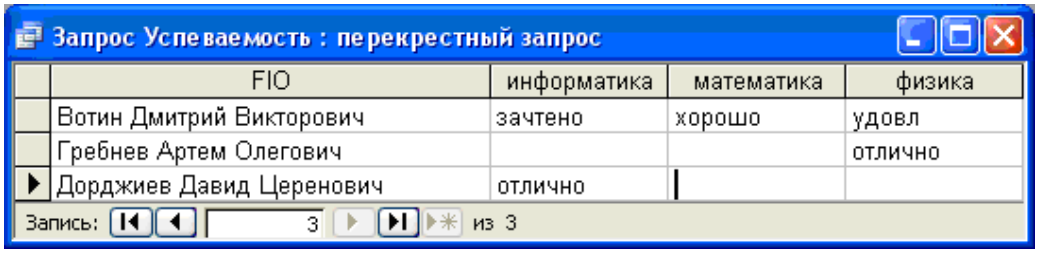

Рис. 5.4. Результаты выполнения перекрестного запроса об успеваемости

В запросе на выборку можно использовать параметры, например, нас интересует дисциплина «физика». Для выполнения запроса необходимо ввести в столбце **NamePredmet** запроса условие отбора **физика** (рис. 5.5).

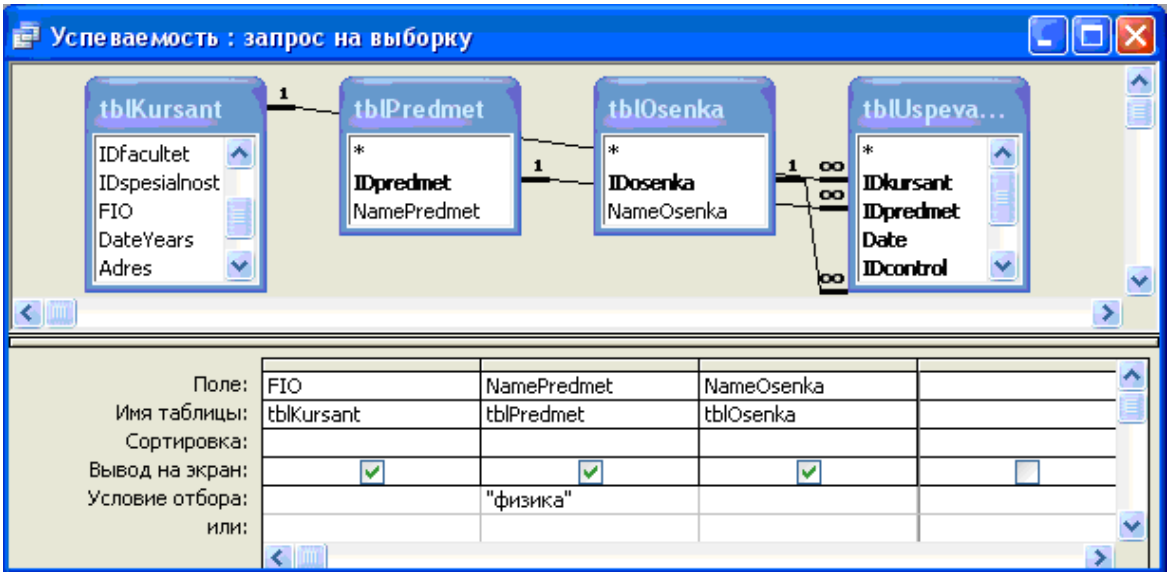

Рис. 5.5. Запрос об успеваемости по дисциплине **Физика** в режиме **Конструктора**

Результаты выполнения запроса по физике представлены на рис 5.6.

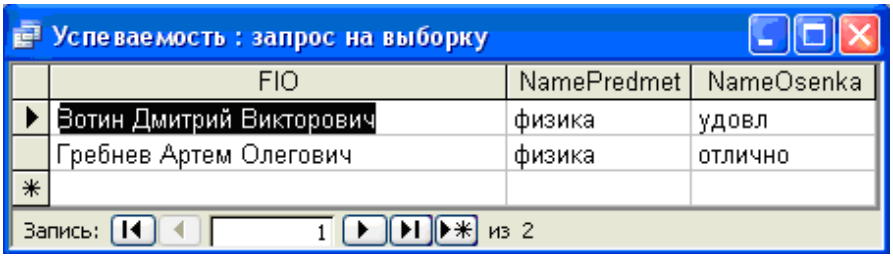

Рис. 5.6. Результаты выполнения запроса об успеваемости по дисциплине **Физика**

5.2. Использование вычисляемых полей в запросах

Вычисляемое поле, включенное в запрос, позволяет получить новое поле с результатами вычисления только в таблице запроса и не создает полей в таблицах базы данных.

Требуется определить:

1. Средний балл по предметам. Для этого необходимо использовать функцию определения среднего значения (avg). Для формирования выражений в вычисляемом поле целесообразно использовать построитель выражений.

2. Количество курсантов (студентов) в группе (функция count).

Запрос в режиме **Конструктора** представлен на рис. 5.7.

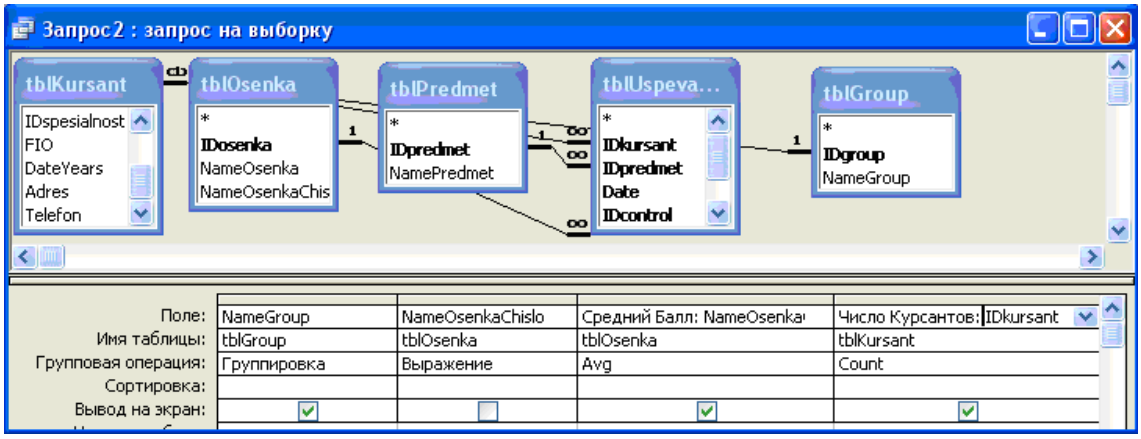

Рис. 5.7. Запрос по количеству курсантов (студентов)
и среднему баллу в режиме **Конструктора**

Результаты выполнения запроса представлены на рис. 5.8.

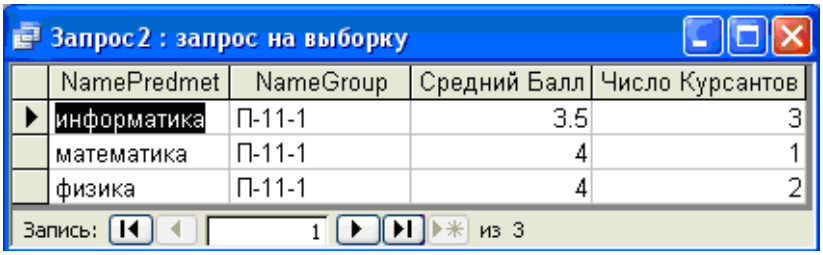

Рис. 5.8. Результат выполнения запроса по среднему баллу
и количеству курсантов (студентов), сдававших экзамены по предметам

Запросы являются мощным средством обработки данных, позволяют легко получать данные из нескольких таблиц, достаточно наглядны, но для вывода данных в выходные документы используются отчеты, что и будет рассмотрено в следующем разделе.

6. РАЗРАБОТКА ОТЧЕТОВ

Средства СУБД Access по разработке отчетов предназначены для создания макета отчета, по которому может быть осуществлен вывод данных из таблиц в виде выходного печатного документа. Эти средства позволяют конструировать отчет сложной структуры, обеспечивающий вывод взаимосвязанных данных из многих таблиц. При этом могут быть выполнены самые высокие требования к оформлению документа.

В процессе конструирования формируется состав и содержание разделов отчета, а также размещение в нем значений, выводимых из полей таблиц базы данных. Кроме того, оформляются заголовки, подписи реквизитов отчета, размещаются вычисляемые реквизиты.

Средства конструирования отчета позволяют группировать данные по нескольким уровням. Для каждого уровня могут проводиться вычисление итогов, определяться заголовки и примечания по каждой группировке.

Отчет может создаваться с помощью **Мастера** или в режиме **Конструктора отчетов**. Во многих случаях удобно использовать **Мастер отчетов**. Созданный мастером отчет можно доработать в режиме **Конструктора.**

При необходимости вывода отчета данных из многих таблиц в качестве основы для отчета может быть использован многотабличный запрос. На запрос могут быть возложены наиболее сложные виды выборки и предварительной обработки данных. Разнообразные возможности **Конструктора отчетов** позволяют полученные в запросе данные успешно структурировать и оформлять.

Создание и изменение макета отчета осуществляется в окне **Конструктора отчетов.**

6.1. Порядок создания отчета в режиме Конструктора

1. В окне базы данных выбрать закладку Отчеты и нажать кнопку Создать. В окне **Новый отчет** выбрать таблицу **Курсант**, которая будет служить источником данных для отчета. Выбрать режим **Конструктор.**

В открывшемся окне **Конструктора** отчет содержит разделы, показанные на рис. 6.1.

Если отсутствует раздел **Заголовок отчета**, добавить его с помощью кнопки Заголовок.

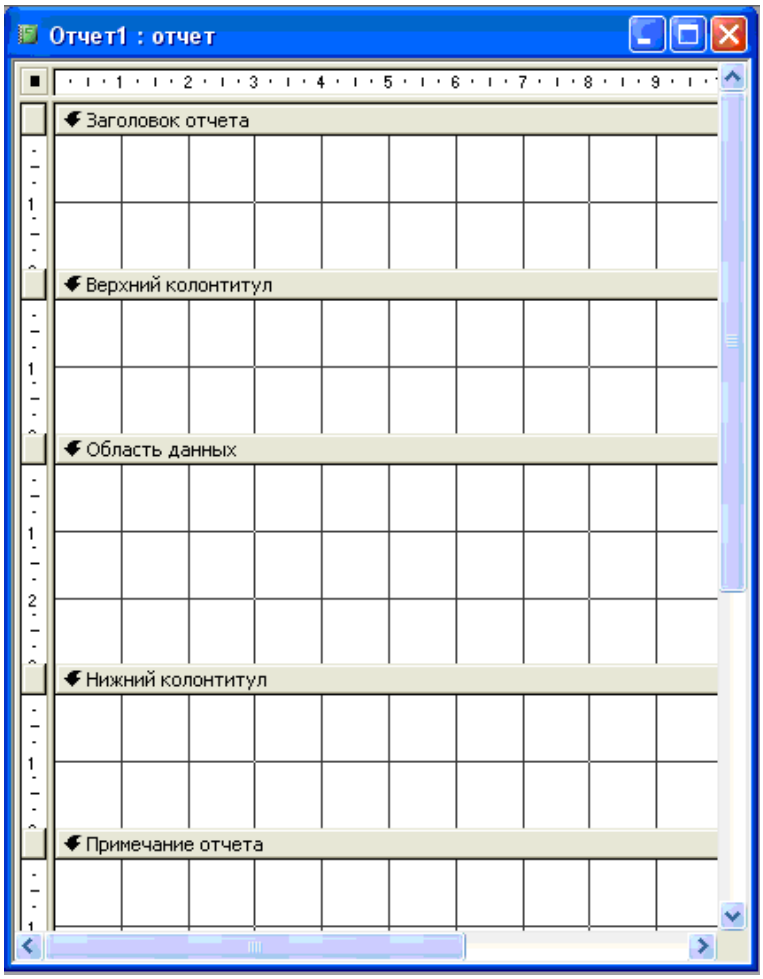

Рис. 6.1. Разделы отчета в режиме **Конструктора отчетов**

2. Группировка данных отчета. Поскольку общий список курсантов (студентов) должен быть разбит по группам, выполнить группировку по полю **Идентификатор группы**. Для этого нажать кнопку Сортировка и группировка на панели инструментов **Конструктора** и заполнить поля открывшегося окна, как показано на рис. 6.2.

В окне будет представлено поле **Идентификатор группы**, по которому определена группировка, и поле **Идентификатор курсанта**, по которому определена только сортировка. Для создания в отчете заголовка и примечания группы в соответствующих строках области свойства группы в окне **Сортировка и группировка** выбрано **Да**. Если установить курсор на строку поля **Идентификатор курсанта**, в его свойствах группы будет проставлено **Нет** как в строке

заголовка, так и в строке примечания, что и определяет только сортировку по этому полю.

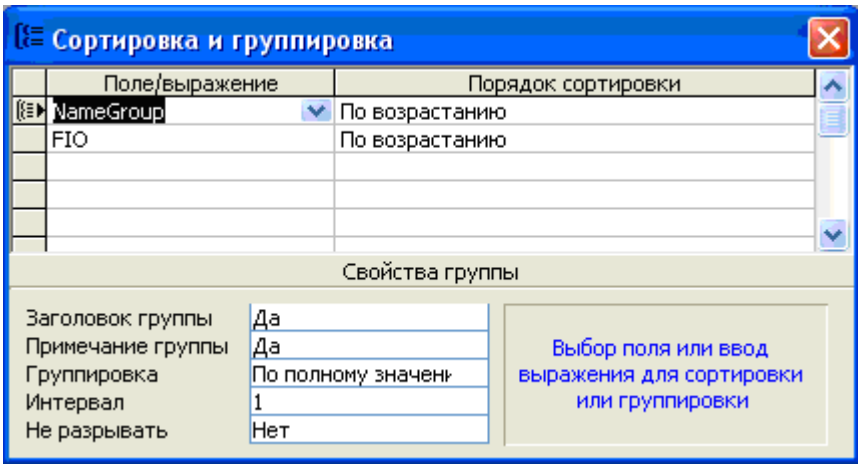

Рис. 6.2. Окно определения групп записей по полю **Идентификатор группы** и сортировки по полю **Номер курсанта**

После определения группировки в окне **Конструктора отчетов** появляются дополнительные разделы **Заголовок группы** и **Примечание группы**.

Рассмотрим на примере создание отчета. Данные в отчете располагаются так, как показано на рис. 6.3.

СПИСКИ·КУРСАНТОВ¶ **(Текущая·дата)¤**			
Список·курсантов·группы·_____¶ ¤			
Номер¤	**Фамилия·И.О.¤**	**Дисциплина¤**	**Оценка¤**
¤	¤	¤	¤
¤	¤	¤	¤
Средний·балл·группы····_____¤			

Рис. 6.3. Макет для создания отчета со списками курсантов (студентов)

Порядок создания отчета:

1. **Разместить поля из таблицы**. В заголовок отчета поместить надпись **СПИСКИ КУРСАНТОВ**. Значение номера группы должно быть представлено один раз в заголовке группы. Разместить поле **Идентификатор группы** в

разделе **Заголовок группы**, **Примечание группы**. Для этого нажать кнопку панели инструментов Список полей и перетащить поле Идентификатор группы в раздел заголовка группы. Откорректировать подпись поля, изменив ее на **Список курсантов группы.**

2. Установить в элементах шрифт Times New Roman, 13 пунктов. Нажать кнопку панели инструментов Выбрать размер по данным для установки размеров рамки по размеру текста подписи.

3. Для формирования табличной части отчета последовательно разместить поля **Номер курсанта**, **ФИО**, **Дисциплина**, **Оценка в области данных**. Поле размещается вместе с подписью, которая берется из таблицы **Курсант**. Подписи полей можно перенести в область заголовка путем вырезания и вставки. Если они не совпадают с названиями столбцов в макете, их надо откорректировать. Подписи также можно создать заново, воспользовавшись кнопкой панели элементов Надпись.

4. **Включить вычисляемое поле в отчет**. Для включения расчетного реквизита **Средний балл** нажать кнопку Поле на панели элементов и разместить элемент **свободный** в раздел **примечание группы**. Определить в свойствах этого элемента выражение для расчета среднего значения. Для этого записать на закладке Данные функцию ***= Avg([Балл])***, в строку **Число десятичных знаков – 2** на закладке Макет, в строку **Формат поля** – **фиксированный**. Отредактировать подпись поля. Для этого выделить подпись и вызвать ее свойства. В свойствах на закладке Макет в строке подпись записать **Средний балл группы.**

5. **Добавить текущую дату и номер страницы**. Для этого воспользоваться встроенной функцией ***Now( )***. Создать в заголовке отчета свободный элемент и в окне его свойств на закладке Данные в строке **Данные** записать ***= Now( )***. На закладке Макет в строке **Формат поля** выбрать **Полный формат даты.** Подпись этого поля выделить и удалить.

6. **Добавить номер страницы в нижний колонтитул**. Для этого создать свободный элемент и в его свойствах на закладке Данные в строке **Данные** записать ***=[Page]***. Отредактировать подпись этого поля, записав в его свойствах на закладке Макет в строке **Подпись – Стр.**

Полученный отчет представлен на рис. 6.4.

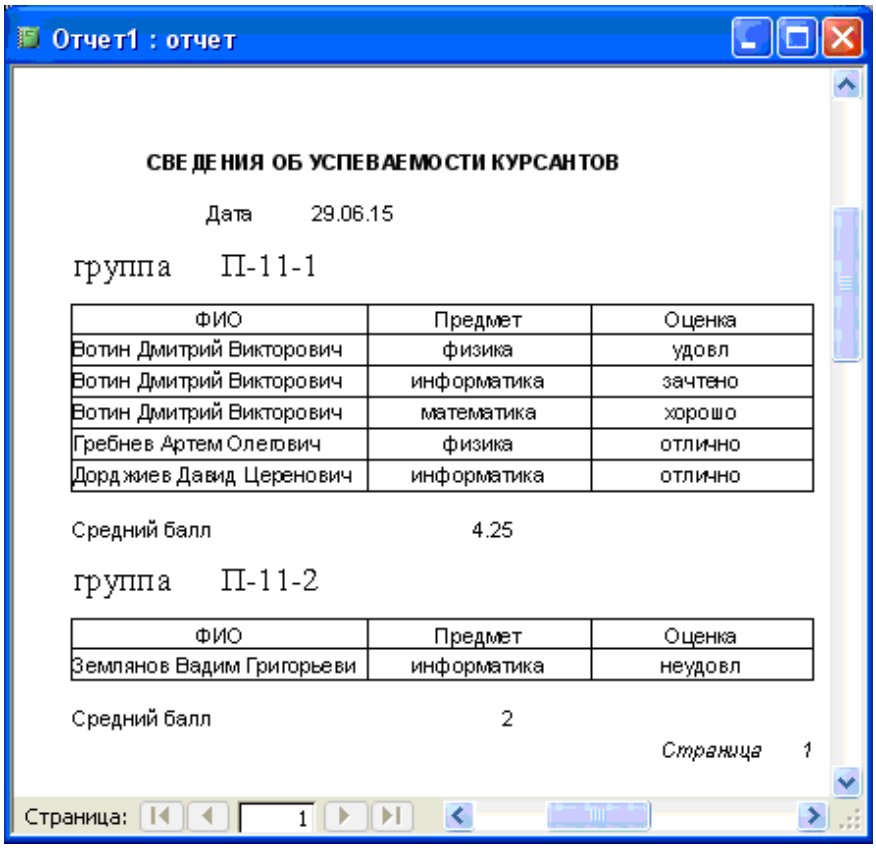

Рис. 6.4. Отчет **Сведения об успеваемости курсантов**

Рассмотренный порядок конструирования позволяет создавать базы данных начинающему разработчику, разрабатывать простой интерфейс пользователя, позволяющий осуществлять ввод, корректировку и вывод данных, но создание более сложных форм и отчетов требует знания встроенного языка программирования, что является следующим шагом в освоении процесса создания приложений пользователя.

7. РАЗРАБОТКА ПРИЛОЖЕНИЯ ПОЛЬЗОВАТЕЛЯ

Приложение пользователя, как правило, образуется объединением некоторого множества задач предметной области. Задача может быть определена как совокупность действий по формированию выходной информации на основе входной информации. Выходной информацией являются печатные документы, файлы, сообщения, содержащие результаты решения и имеющие определенное назначение в данной предметной области. Входной информацией являются данные входных документов.

Для решения задач пользователя средствами СУБД необходимо формализовать эту задачу. При реализации задач широко используются объекты Access: формы, отчеты, запросы, ориентированные на конечного пользователя, не являющегося программистом. При проектировании задачи необходимо выполнить постановку и алгоритмизацию задачи на основе исходной базы данных. В процессе постановки определяется цель, назначение и сущность задачи, установка периодичности и сроков решения, входная и выходная информация. При этом разрабатываются необходимые функции по получению выходной информации на основе входной.

При разработке алгоритма необходимо обеспечить требование, при котором для любой совокупности исходных данных должен быть получен искомый результат после выполнения конечного числа шагов. Можно выделить следующие этапы алгоритмизации задач:

– декомпозиция общей задачи на подзадачи, реализуемые средствами СУБД;

– разработка блок-схемы задачи, определяющей взаимосвязи подзадач;

– формализация выполнения каждой подзадачи.

Для проверки разработанных приложений необходимо подобрать контрольный пример, который должен содержать набор необходимых и достаточных данных для тестирования разработанных алгоритмов.

При разработке алгоритмов с ориентацией на средства создания форм, запросов и отчетов рассматриваются укрупненные операции обработки данных, например, такие, которые реализуются одним запросом. Описание таких алгоритмов, как правило, не содержит циклов и может быть достаточно полно представлено блок-схемой.

Наряду с такими объектами, как формы, запросы и отчеты, для реализации практических задач пользователя используются средства программирования: макросы и встроенный язык программирования VBA. Завершающим этапом создания приложения является конструирование интерфейса приложения пользователя.

7.1. Основы создания макросов

Язык макросов является некоторой разновидностью языка программирования, который позволяет реализовать задачи пользователя, выполняя необходимые действия над объектами Access и их элементами.

Макрос составляется из последовательности макрокоманд.

Макрокоманда – это инструкция, ориентированная на выполнение определенного действия.

Например, макрокомандой можно открыть форму, отчет, выполнить запрос и др. Язык макросов обеспечивает возможность выполнения большинства задач, не прибегая к программированию на VBA. Макросы обеспечивают пользователя средствами решения задач, не требующих знания программирования.

В Access имеются средства, обеспечивающие взаимодействие макросов с объектами на основе событий. События наступают при выполнении определенных действий, к которым относятся действия пользователя.

Создание макросов осуществляется в диалоговом режиме и сводится к записи в окне макроса последовательности макрокоманд, для которых задаются параметры. Каждому макросу присваивается имя. При выполнении макроса макрокоманды выполняются последовательно в порядке их расположения. Выполнение макроса сводится к его запуску.

Макросы и порядок их создания достаточно полно рассмотрены в справочной системе Access, поэтому подробно рассматривать их не будем и перейдем к рассмотрению основ программирования на встроенном языке программирования – но для этого необходимо хорошо представлять объектные модели Microsoft Access.

7.2. Объектные модели Microsoft Access

Язык Visual Basic for Applications является объектно-ориентированным языком программирования. Стандартные объекты Visual Basic представляют собой основное средство манипуляции с данными Microsoft Access и других приложений семейства Microsoft Office. Знание технологии объектно-ориентированного программирования и состава объектных моделей Visual Basic позволяет разрабатывать профессиональные приложения, выполняющие всю необходимую обработку данных.

Основными понятиями языка VBA являются:

- объект;
- семейство;
- метод;

– класс;
– свойство;
– событие;
– объектная модель.

Объект – это абстракция, с которой производятся операции в объектно-ориентированных языках программирования. Объект обладает собственными характерными признаками, отличающими его от других объектов, и имеет свое поведение. Примерами объектов Access являются таблицы, формы, отчеты, запросы.

Класс представляет собой описание совокупности однотипных объектов. Класс можно сравнить с типом данных, где переменной такого специфического типа является объект. В этом случае говорят, что объект представляет собой экземпляр определенного класса.

Каждый объект имеет свойства и методы, которые различны у разных классов объектов, но применяются одинаково.

Свойством называют отдельную характеристику объекта или класса. Например, свойства формы являются свойствами объекта **Form**. Свойство объекта может принимать определенное значение. Например, свойство **Вывод на экран** (Visible) может принимать значение **True** или **False**, в зависимости от чего форма будет появляться или исчезать с экрана.

Метод представляет собой процедуру (или функцию) объекта или класса. Совокупность методов объекта определяет его «поведение». Например, объект **Form** имеет метод **Refresh**, вызов которого позволяет обновить данные в форме Access.

Объект может реагировать на определенные *события*, происходящие в процессе работы приложения и влияющие на объект. Совокупность событий, на которые объект способен реагировать, определяется создателем класса, экземпляром которого является данный объект. Например, набор событий, которые определены для формы Access, можно увидеть на вкладке События (Event) диалогового окна **Свойства** (Properties). Реакцией объекта на произошедшее событие может быть выполнение объектом некоторой специальной процедуры, которая называется *процедурой обработки события*. Любому событию объекта может быть назначена некоторая процедура его обработки.

Упорядоченный набор однотипных объектов – экземпляров одного класса – называется *семейством*. Семейство тоже является объектом. Одним из методов этого объекта является процедура, возвращающая ссылку на конкретный объект в семействе. Одним из свойств семейства является число объектов, хранящихся в нем. Например, совокупность элементов управления в форме образует семейство **Controls.**

Объекты и семейства сгруппированы в виде иерархических структур, которые называются объектными моделями. В VBA определены специальные объектные модели для каждого компонента семейства Microsoft Office и объектные модели, общие для всех компонентов Microsoft Office. Объектные модели VBA можно изучать, используя справочную систему и окно просмотра объектов. Окно просмотра объектов представляет собой специальное средство редактора VBA, позволяющее просматривать содержимое библиотек объектов и производить поиск справочной информации.

Объектная модель Microsoft Access реализована в виде набора объектов, собранных в библиотеке Access. Основным элементом в иерархии объектов библиотеки Access является объект **Application**. Он содержит ссылки на все объекты и семейства объектов Microsoft Access. Каждый объект из библиотеки Access имеет в качестве свойства объект **Application** (в том числе и сам объект **Application** имеет свойство **Application**), который ссылается на активное приложение Microsoft Access.

Иерархия объектов и семейств объектов Microsoft Access представлена на рис. 7.1–7.3. Названия объектов, являющихся элементами семейств, приведены в скобках. Иерархия объектов, представленная на рисунках, образована следующим образом:

– каждый объект может содержать набор свойств, часть из которых может являться ссылками на другие объекты;

– в каждый новый уровень иерархии входят объекты, ссылки на которые хранятся в объектах, расположенных на предыдущем уровне иерархии.

Если свойство объекта представляет собой ссылку на объект, определенный в другой библиотеке (не в библиотеке Access), для него приводится название этой библиотеки.

В Microsoft Access 2003 появились два новых семейства, содержащих новые объекты:

– **Printers** – обеспечивает программное управление параметрами печати;

– **AllFunctions** – используется для программного доступа к пользовательским функциям, определенным в базе данных Microsoft SQL Server (объект, аналогичный запросу, который существует в проекте Microsoft Access). Этот объект может быть открыт в режиме **Конструктора**, режиме **Таблицы**, режиме **Предварительного просмотра** и в режиме **Сводной таблицы** или **Сводной диаграммы** как обычный запрос.

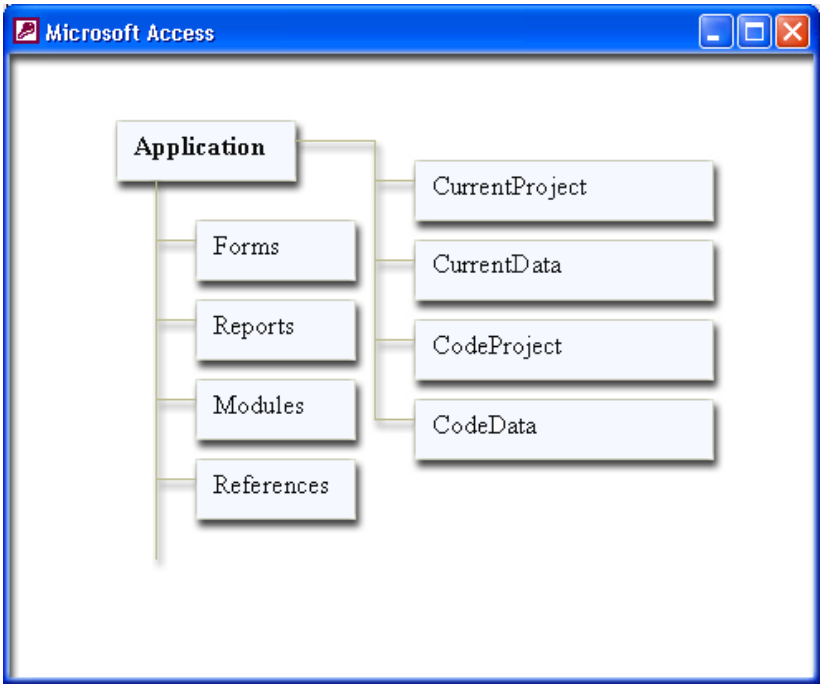

Рис. 7.1. Первый уровень иерархии объектной модели Microsoft Access

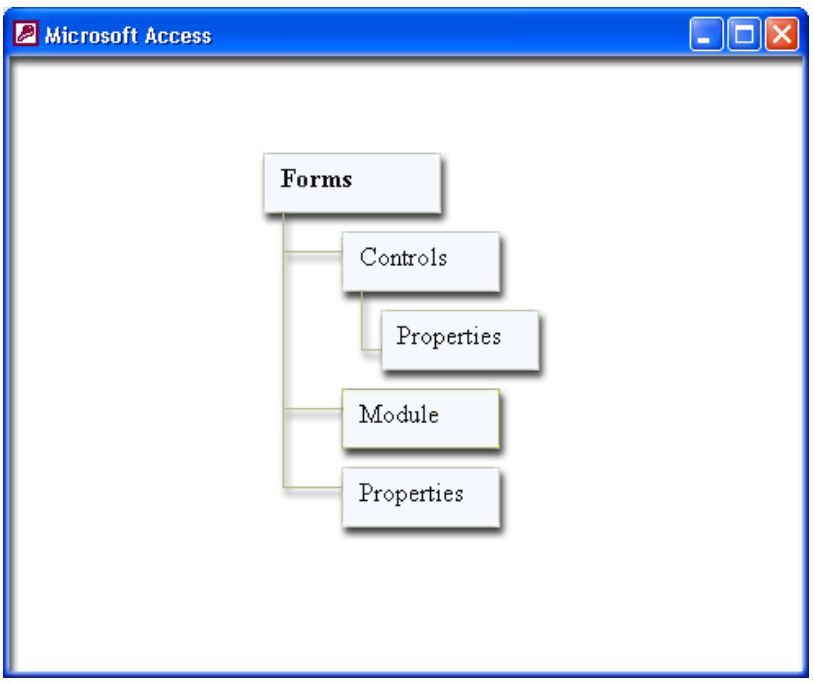

Рис. 7.2. Второй уровень иерархии объектной модели Microsoft Access

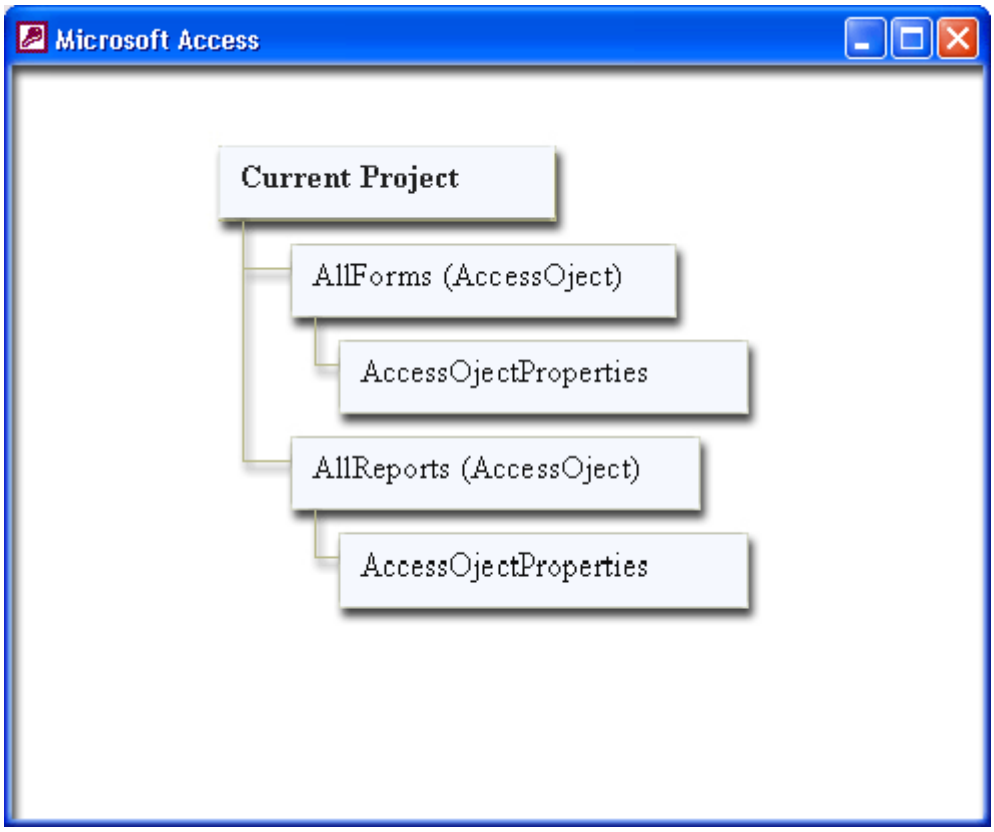

Рис. 7.3. Второй уровень иерархии объектной модели Microsoft Access для работы с базой данных и проектом

Далее рассмотрим вопросы доступа к данным.

7.3. Объектные модели доступа к данным

Объекты доступа к данным предназначены для программного доступа и управления данными в локальной или удаленной базе данных, а также для программного управления самими базами данных, их объектами и структурой. В Microsoft Access существует два способа работы с данными:

– старый, посредством DAO (Data Access Objects);

– новый, посредством ADO (ActiveX Data Objects). Каждому из этих способов соответствует своя объектная модель.

Несмотря на разнообразие библиотек объектов, методы работы с объектами в процедурах VBA общие и не зависят от того, к какой библиотеке относится конкретный объект.

Для обращения к объекту существует одно общее правило: необходимо проследить путь в иерархии объектов, начиная от объекта самого верхнего уровня до целевого объекта и записать последовательность имен встреченных на пути

объектов или семейств, отделяя их друг от друга точкой. Если на пути встречается семейство, то, кроме имени семейства, необходимо в скобках указать индекс или имя его элемента, т. е. задать конкретный объект в семействе. Например, чтобы обратиться к форме, входящей в состав семейства **AllForms**, необходимо написать следующее выражение:

Application. CurrentProject .AllForms («Успеваемость»)

Такие длинные ссылки особенно характерны для моделей, которые имеют многоуровневую, иерархическую структуру, например, DAO. На самом деле, на практике чаще применяют не полные, а сокращенные ссылки. Дело в том, что наиболее часто используемые семейства, объекты, свойства и методы считаются *глобальными*. Ссылки на них хранятся в специальном объекте с именем **Global**. Для обращения к глобальному объекту можно пропустить объекты более высокого уровня.

Например, семейство **Forms** является глобальным. Для доступа к объекту этого семейства можно использовать сокращенную ссылку вида

Forms («Успеваемость») вместо полной ссылки:

Application. Forms («Успеваемость»)

Узнать, какие объекты, свойства и методы являются глобальными, позволяет окно просмотра объектов. Чтобы отобразить список глобальных компонентов объектной модели, выберите элемент **<globals>** в списке **Classes** в окне просмотра объектов.

Обычно имя семейства и имя объекта разделяются оператором «!» (восклицательный знак), например:

Forms !Успеваемость

Если имя объекта состоит из нескольких слов, разделенных пробелом, запись выглядит следующим образом:

Forms![Новый курсант]

Третий способ ссылки состоит в том, что на объект в семействе ссылаются не по названию, а по индексу, например:

Properties(0)

Такой способ применяется обычно тогда, когда имя объекта неизвестно. В то же время каждый объект в семействе имеет индекс (порядковый номер), который обычно начинается с нуля. Есть исключения из этого правила, например, в семействе **CommandBars** модели Microsoft Office нумерация объектов начинается с единицы. Поэтому перед использованием индексов лучше узнать о способе нумерации в справочной системе Access.

И, наконец, последний способ ссылки на объект состоит в использовании вместо имени объекта строковой переменной, например:

```
Reports (strИмяОтчета)
```

Если какой-то объект содержит несколько семейств объектов более низкого уровня, то одно из этих семейств, обычно наиболее часто используемое, считается стандартным или семейством по умолчанию. Можно обращаться к объекту, входящему в это семейство, не указывая имени семейства. Например, для объекта TableDef модели DAO стандартным семейством считается **Fields**. Поэтому для обращения к столбцу таблицы можно использовать сокращенную ссылку:

```
TableDefs!Должности!Код Должности вместо полной ссылки
TableDefs!Должности.Fields!КодЦолжности
```

Еще одним способом сокращения ссылки на объект является использование объектной переменной. Особенно полезно использовать объектную переменную, если к объекту нужно обратиться несколько раз.

Доступ к объектам, входящим в семейство, в большинстве случаев возможен только через упоминание имени семейства, поэтому в программах VBA приходится выполнять различные операции с семействами. Например, чтобы обратиться к объекту в семействе, нужно перебрать несколько или даже все объекты семейства. Допустим, требуется проверить, открыта ли определенная форма в нашем приложении. Для этого мы можем написать функцию, которая должна проверить, входит ли эта форма в семейство **Forms**. При этом функция может иметь следующий вид:

```
Function IsLoaded (strFormName As String) As Boolean
'Возвращает значение True, если форма открыта и False, если нет
Dim frm As Form
IsLoaded = False
For each frm in Forms
If frm.Name = FormName
Then IsLoaded = True
Exit Function
EndIf
Next frm
End Function
```

В данной функции сначала объявлена объектная переменная типа **Form** и установлено начальное значение функции. Затем организуется цикл, в котором перебираются элементы семейства **Forms** до тех пор, пока не обнаружится объект **Form** с именем, совпадающим со строковой переменной, которая является

аргументом функции. Если такой объект найден, присваивается функции значение **True**. Если нет, то функция вернет значение **False**.

Для того чтобы организовать цикл с перебором элементов, количество которых неизвестно, в настоящем примере использован оператор **For Each. . .Next.** Этот оператор обычно используется при работе с семействами. Однако можно организовать такой же цикл, используя обычный оператор **For. . .Next**, т. к. любое семейство (в том числе и **Forms**) имеет свойство **Count**, которое возвращает количество элементов в семействе. Ниже приводится другой вариант организации этого цикла:

```
For I = 0 To Forms.Count - 1
If frm(I).Name = FormName Then
IsLoaded = True
Exit Function
EndIf
Next I
```

Кроме указанного свойства, семейства объектов доступа к данным имеют два метода, которые позволяют добавлять объекты в семейство и удалять их из семейства – методы **Append** и **Delete**. Это обеспечивает оперативное создание объектов доступа к данным, например, временных таблиц. Чтобы создать таблицу программным путем, необходимо определить эту таблицу и добавить ее в соответствующее семейство. Ниже приводится фрагмент кода процедуры, в котором создается новая таблица, определяются два ее поля и добавляются объекты в семейства **Fields** и **TableDefs**

```
'Объявляем объектные переменные для объектов:
'база данных, таблица и поле
Dim db As Database,
td As TableDef, fid As Field
'Устанавливаем ссылку на текущую базу данных
Set db = CurrentDb
'Создаем новую таблицу, используя метод
CreateTableDef объекта Database
Set td = db.CreateTableDef("Временная")
'Создаем поле в таблице, используя метод
CreateField объекта TableDef
'Поле будет иметь имя "Дата" и тип Дата/время
Set fid = td.CreateField("Дата",dbDate)
'Добавляем поле "Дата" в семейство Fielfs таблицы
td.Fields.Append fid
```

```
'Создаем второе поле с именем "Сумма" и типом Денежный
Set fid = td.CreateField("Сумма",dbCurrency)
'Добавляем поле "Сумма" в семейство Fielfs таблицы
td.Fields.Append fid
'Добавляем таблицу к семейству TableDefs базы данных
db.TableDefs.Append td
'Обновляем семейство TableDefs
db.TableDefs.Refresh
```

После выполнения этой программы на вкладке Таблицы (Tables) окна базы данных появится новая таблица **Временная.**

Удалить эту таблицу можно аналогичным способом, только уже воспользовавшись методом **Delete** семейства **TableDefs**:

```
db.TableDefs.Delete "Временная"
db.TableDefs.Refresh
Set db = Nothing
```

Метод **Refresh** обновляет количество объектов семейства после добавления или удаления объектов. В последнем предложении освобождается объектная переменная **db.**

Каждый объект имеет свойства, которые являются его характеристиками, и методы, которые позволяют управлять поведением этого объекта. То же справедливо и для семейств. Работа с объектами и семействами заключается в установке или получении значений конкретных свойств объекта или семейства и вызове их методов. Поэтому для управления объектами приложения необходимо хорошо знать свойства и методы каждого объекта.

Установка свойства объекта – это присвоение значения данному свойству. Поэтому для установки свойства используется оператор присваивания, например, чтобы установить свойство **Visible** элемента управления формы, можно использовать инструкцию VBA:

```
Forms! Kursant ! IDkursanta . Visible = False
```

Получить свойство означает прочитать текущее значение этого свойства. Например, инструкция VBA

```
intCount = Forms . Count
```

присваивает переменной значение свойства **Count** семейства **Forms.**

Метод объекта в инструкциях VBA обозначается так же, как и свойство. Однако, в отличие от свойств, методы могут иметь аргументы. Например, в следующей инструкции VBA применяется метод **OpenReport** объекта **DoCmd.**

```
DoCmd.OpenReport "Ведомость", acPreview
```

Объект **DoCmd** – это специальный объект, который позволяет в программах VBA выполнять макрокоманды. Имя каждой макрокоманды является методом этого объекта. В приведенном примере создается отчет **Ведомость** в режиме **Предварительного просмотра**. При этом метод **OpenReport** не возвращает никакого значения, и аргументы в данном случае не требуется заключать в скобки.

Если же метод используется как функция, возвращающая значения, как показано в следующем примере, аргументы необходимо заключить в скобки:

```
Dim db As Database, rs As Recordset
Set db = CurrentDB
Set rs = db.OpenRecordset ("Ведомость")
.....
rs.Close
db.Close
```

Метод **OpenRecordset** объекта **Database** возвращает ссылку на объект **Recordset** (Набор записей), присваиваемую объектной переменной **rs.** Метод close, который имеют оба объекта – **Recordset** и **Database**, – не использует аргументов.

7.4. Модули VBA

Код VBA в приложении Access содержится в модулях. Модули являются объектами Access, такими же, как таблицы, запросы, формы, отчеты, страницы и макросы. Основное содержание модулей – это процедуры на языке VBA. Существуют два типа модулей: стандартные модули и модули класса.

Стандартные модули содержат общие процедуры, которые не связаны с конкретным объектом: формой или отчетом. Эти процедуры могут вызываться из других модулей и использоваться при обработке событий в разных объектах для вычисления значений в разных запросах или формах. Если в процедурах модуля нет ссылок на конкретные объекты данного приложения (формы, отчеты, элементы управления), то такой модуль может быть с успехом использован другими приложениями Access. Стандартные модули применяются также для объявления глобальных (то есть доступных из всех модулей приложения) переменных, констант, типов.

Вторым типом модуля в Access является модуль класса. Модуль класса отличается от стандартного модуля тем, что, кроме процедур, он содержит описание

объекта и используется для создания объектов. Процедуры, определенные в этом модуле, являются методами и свойствами объекта. Примерами модулей класса являются модули форм и отчетов.

Модули форм и отчетов связаны с конкретной формой или отчетом и содержат процедуры обработки событий для этой формы или отчета. Модуль формы не создается сразу при создании новой формы. Он создается и связывается с формой, как только вы попытаетесь создать первую процедуру обработки событий для этой формы или одного из элементов управления формы или же нажмете кнопку Программа (Code) в окне **Конструктора формы.**

Формы и отчеты являются стандартными классами объектов в Access, однако можно использовать модули класса для создания пользовательских объектов. Имя, под которым сохраняется модуль класса, становится именем специального объекта.

Для того чтобы создать стандартный модуль или модуль класса, необходимо:

1. Выбрать команду Модуль (Module) или Модуль класса (Class Module) в меню Вставка (Insert) или в списке кнопки Новый объект (New Object) выбрать соответствующий объект (рис. 7.4). При этом откроется редактор кода VBA с пустым окном модуля.

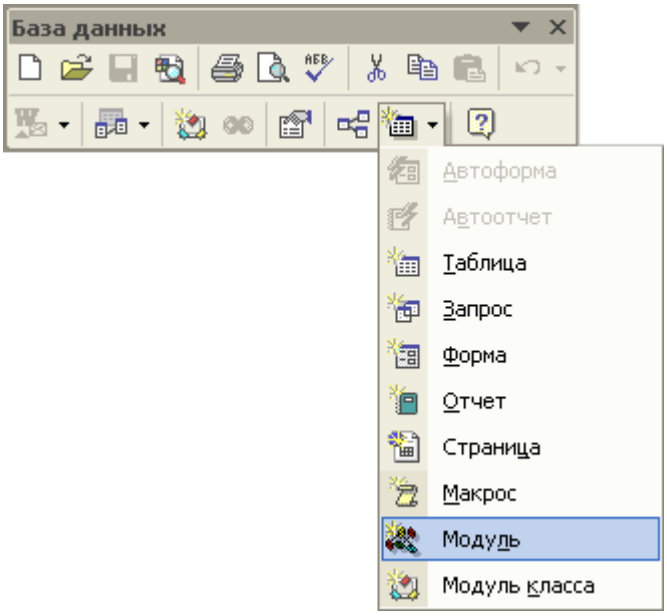

Рис. 7.4. Создание нового модуля

2. Создать необходимые процедуры и описания.

3. Сохранить модуль, нажав кнопку Сохранить (Save) на панели инструментов. При этом выдается диалоговое окно Сохранение (Save), в котором необходимо ввести имя нового модуля и нажать кнопку ОК.

После этого новый модуль появляется в списке модулей окна базы данных. Чтобы его открыть, можно нажать кнопку Конструктор (Design) окна базы данных. Если открыты форма или отчет в режиме **Конструктора**, то для того, чтобы открыть модуль формы (отчета), следует нажать кнопку Программа (Code) на панели инструментов.

Для обращения к модулям в программах VBA используется семейство **Modules**, которое содержит все открытые объекты типа **Модуль** (Module). Для того, чтобы открыть объект **Module**, можно использовать макрокоманду **ОткрытьМодуль** (OpenModule). Ссылка на модуль может быть создана тремя способами:

– имяСемейства!имяОбъекта, например, Modules!Startup;

– имяСемейства ("имяОбъекта"), например, Modules ("Startup");

– имяСемейства (индекс), где индекс – индекс объекта в семействе.

Для ссылки на модуль формы или отчета можно использовать или имя модуля, например Modules ! Form_Курсант, или свойство формы, например

Forms!Курсант.Module

Для того чтобы открыть окно редактора, достаточно открыть любой модуль Access (рис. 7.5).

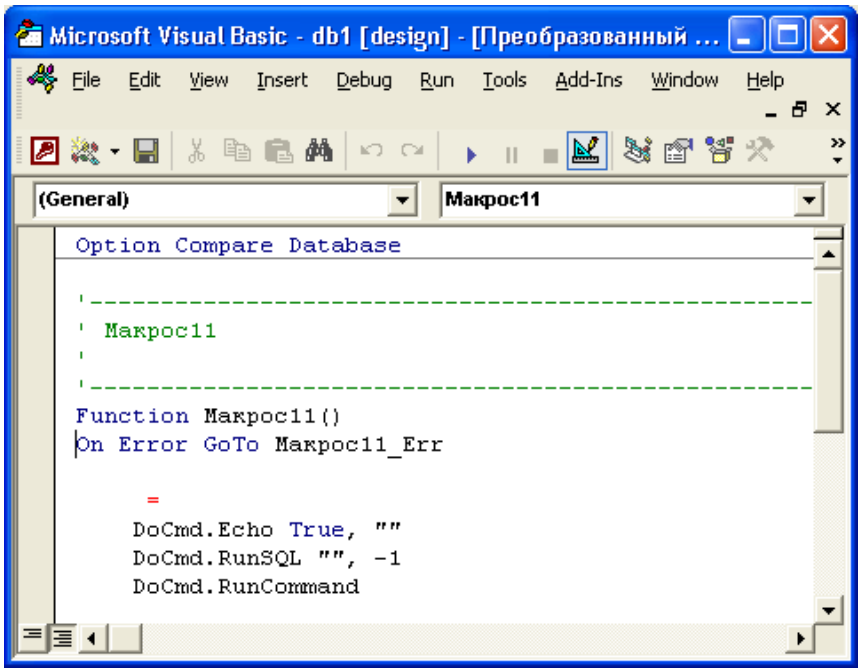

Рис. 7.5. Окно редактора кода VBA

Обычно в окне редактора используются три панели (три отдельных окна):

– **Project** (Панель проекта), располагается в верхнем левом углу редактора. В ней отображается иерархическое дерево модулей приложения. Если это окно неактивно, выполните команду View → Project Explorer либо нажмите комбинацию клавиш [Ctrl] + [R];

– **Properties** (Панель свойств), находится под панелью проекта. Она позволяет просматривать и изменять свойства различных входящих в проект объектов, отображаемых на панели проекта. Список свойств может отображаться как в алфавитном порядке, так и по категориям;

– **Code** (Панель редактора кода). Это окно занимает большую часть экрана и является «многодокументным», т. е. можно открыть одновременно несколько окон данного типа для разных модулей. Оно представляет собой высокоинтеллектуальный текстовый процессор, существенно облегчающий написание кода VBA.

Проект приложения состоит из модулей, которые делятся на три категории:

– **Microsoft Access Class Objects** (Модули классов Access) – включает все модули форм и отчетов;

– **Modules** (Модули) – стандартные модули;

– **Class Modules** (Модули классов) – модули пользовательских классов, если они присутствуют в приложении.

Список объектов в окне проекта является иерархическим, кроме модулей самого проекта, он может включать ссылки на объекты из внешних библиотек и список модулей этих библиотек:

– окно **Object Browser** (Обозреватель объектов) позволяет просматривать все объекты, их свойства и методы, доступные для текущего проекта. Объекты могут быть встроенными объектами Access или VBA, объектами, которые вы создали в своем приложении, а также объектами, входящими во внешние библиотеки, на которые имеются ссылки в текущем проекте;

– окно **Object Browser** (Обозреватель объектов) состоит из нескольких списков (рис. 7.6), которые обеспечивают трехуровневое представление информации;

– список **Project/Library** (Проект/Библиотека) в левом верхнем углу окна содержит перечень всех библиотек и проектов, на которые имеются ссылки в данном проекте. Как минимум, он включает библиотеку Access, библиотеку VBA, библиотеку текущего проекта.

При выборе из списка одной из библиотек в нижнем левом поле **Classes** (Классы) отображается список следующего уровня – перечень всех объектов, входящих в эту библиотеку. Например, если выбрать библиотеку Access, то в списке **Classes** можно увидеть много знакомых объектов. Выбрав один из них, например **DoCmd**, в правом поле **Members of** можно увидеть все методы этого объекта. Если выбрать объект **Form**, то в списке справа отобразятся все свойства и методы объекта **Form**.

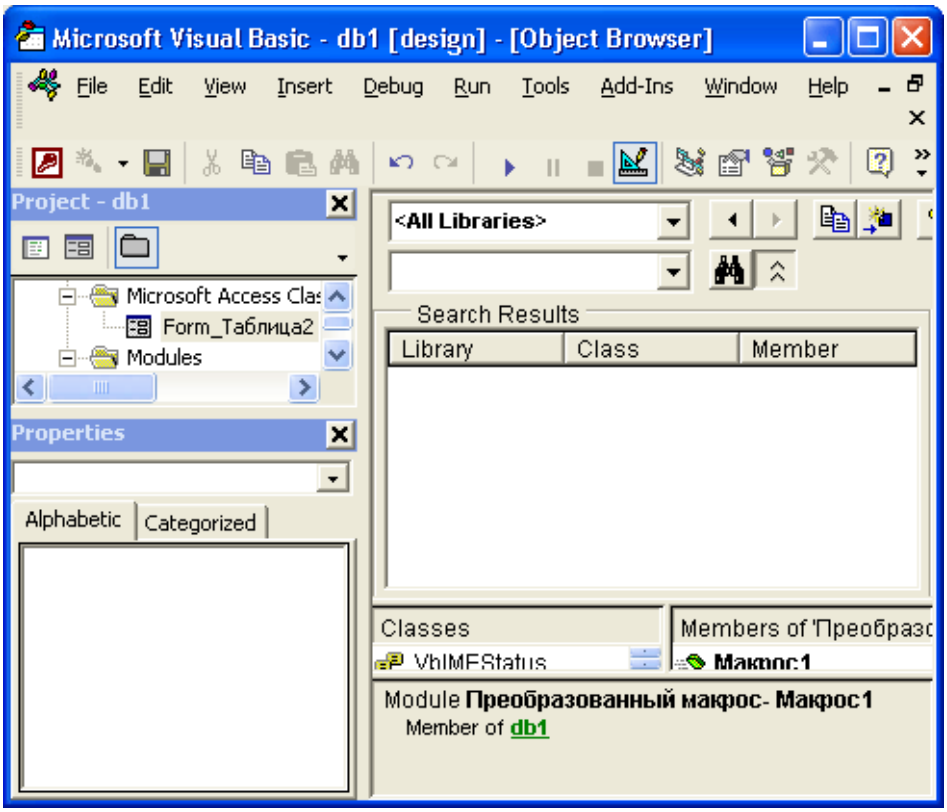

Рис. 7.6. Окно просмотра объектов

При этом в нижней части окна, которая называется *область описания*, отобразится описание выбранного элемента.

Помимо редактора текста профессиональная среда программирования обычно содержит инструментальные средства отладки. Инструментальные средства отладки призваны дать разработчику максимально ясное представление о том, как работает его программа. И уже искусство разработчика состоит в том, чтобы, используя все имеющиеся в его распоряжении средства, быстро выявить ошибки. Набор средств отладки в Access достаточно широк: это и специальное меню **Debug** (Отладка), и во многом дублирующие его кнопки на панели инструментов, и специальные окна отладки.

Кнопки на инструментальной панели в основном повторяют описанные команды. Это видно из значков, находящихся рядом с командой и на кнопках.

Специальные окна отладки используются главным образом для того, чтобы просматривать промежуточные значения данных: переменных, выражений, свойств объектов и т. д. К ним относятся два окна: **Locals** (Локальные) и **Watches** (Контрольные).

Если необходимо добавить выбранное выражение к контрольным значениям, просто нажмите кнопку Add (Добавить).

События Access инициируются действиями пользователя. В зависимости от производимых пользователем действий события можно разделить на несколько типов: события данных, события фокуса, события клавиатуры, события мыши, события печати, события фильтра, события окна, события ошибок и события таймера. Ниже рассмотрим все эти типы событий.

7.5. События данных

К этому типу относятся события, происходящие тогда, когда пользователь вводит, удаляет или изменяет данные в форме, а также перемещается от одной записи к другой.

Событие **Текущая запись** (Current) происходит, когда очередная запись получает фокус или выполняется повторное обращение к источнику данных формы – таблице или запросу. Таким образом, оно возникает как при открытии формы, так и при переходе от одной записи к другой. Чаще всего это событие используется для синхронизации записей в связанных формах.

Событие **Удаление** (Delete) происходит, когда пользователь пытается удалить запись из формы. Оно происходит до того, как запись реально удаляется из базы данных. Процедура обработки этого события имеет параметр **Cancel**. Установка значения этого параметра в процедуре равным **True** позволит предотвратить удаление записи.

Удаление записи из базы данных имеет такое большое значение, что с ним связано несколько событий. После события **Удаление** (Delete) Access выдает специальное окно, запрашивающее подтверждение удаления записи. Событие **До подтверждения Del** (BeforeDelConfirm) возникает до появления этого окна. Процедура обработки данного события имеет два параметра: **Cancel** и

Response. Присваивая в процедуре значение **True** параметру **Cancel**, можно отменить удаление, и окно подтверждения выдаваться на экран не будет. Так что это еще одна возможность отменить удаление программно (третья возможность отмены будет предоставлена пользователю в диалоговом окне подтверждения удаления). Если же параметру **Cancel** присвоить значение **False**, то параметр **Response** можно использовать, чтобы определить, нужно ли выдавать окно подтверждения. Если **Response = 1**, то запись будет удалена без подтверждения, если же **Response** установить равным 0, то Access выдаст окно, запрашивающее у пользователя подтверждение удаления записи.

Замечание. *Если окно подтверждения удаления не выдается или событие* ***До подтверждения Del*** *(BeforeDelConfirm) не возникает, проверьте установку флажка* Изменения записей *(Record Changes) в окне* ***Параметры*** *(Options) (на вкладке* Правка и поиск*, группа* ***Подтверждение).***

Событие **После подтверждения Del** (AfterDelConfirm) происходит как после подтверждения удаления записи, так и при отмене удаления. Процедура обработки данного события имеет один параметр – **Status**, который принимает значения 0, 1 или 2 и определяет, была ли удалена запись. Значение 0 указывает, что запись была успешно удалена, 1 означает, что удаление отменено программой обработки события, а значение 2 указывает, что удаление было отменено пользователем в окне подтверждения удаления. Это событие может быть использовано в программе для проверки, была ли удалена запись.

Со вставкой новой записи связаны два события: **До вставки** (Beforelnsert) и описанное далее **После вставки** (AfterInsert). Событие **До вставки** (Beforelnsert) происходит, как только пользователь вводит первый символ в новую запись (одно из полей, необязательно первое), но до того, как запись фактически будет создана. Процедура обработки этого события может быть использована для проверки того, разрешена ли вставка. Процедура имеет один параметр: **Cancel**. Если установить его значение равным **True**, то вставка записи будет запрещена. После этого события отменить вставку будет уже нельзя, можно только удалить вставленную запись.

После вставки. Событие происходит после того, как в таблицу добавлена новая запись. Обычно это бывает при переходе к следующей записи в форме.

Процедура обработки этого события обычно используется для того, чтобы сделать повторный запрос к источнику данных с целью вывода новой записи.

До обновления. Событие **До обновления** (BeforeUpdate), так же как и следующее событие **После обновления** (AfterUpdate), возникает при любом изменении данных в записи или элементе управления. Это событие может относиться как к элементу управления, так и к записи в целом. Процедура обработки данного события имеет один параметр – **Cancel**, использующийся для того, чтобы отменить введенные изменения. Для этого ему необходимо присвоить значение **True**. Данное событие обычно применяется с целью проверки условий на значение в поле таблицы или записи в целом, если эти условия сложные (простые условия обычно задаются в свойстве **Условие на значение** (ValidationRule) элемента управления). Условия проверяются сразу для нескольких значений, причем в них используются ссылки на элементы управления в других формах. При разных значениях введенных данных выдаются разные сообщения об ошибках.

При невыполнении условий можно отменить введенные изменения перед переходом на другую запись.

Событие **После обновления** (AfterUpdate) происходит после обновления данных в записи или элементе управления. И хотя обновление уже произошло, можно восстановить старые значения, воспользовавшись свойством **OldValue** элемента управления. Оно сохраняет старое значение элемента управления, которое сменится только после события **После обновления** (AfterUpdate).

Замечание. *События* ***До обновления*** *(BeforeUpdate) и* ***После обновления*** *(AfterUpdate), а также* ***До вставки*** *(Beforelnsert) и* ***После вставки*** *(Afterlnsert) не возникают, когда значения элементов управления формы изменяются с помощью программы VBA или макрокоманды* ***УстановитьЗначение*** *(SetValue). Кроме того, события* ***До обновления*** *(BeforeUpdate) и* ***После обновления*** *(AfterUpdate) не возникают для вычисляемых элементов управления.*

Событие **Изменение** (Change) возникает в следующих случаях:

– при изменении содержимого текстового поля или поля со списком, при этом изменением может считаться любой непосредственно введенный или удаляемый символ;

– при изменении значения свойства **Текст** (Text) элемента управления с помощью макроса или процедуры VBA;

– в элементе управления **Набор вкладок** (Tab Control) при переходе с одной вкладки на другую.

Замечание. *Событие* ***Изменение*** *(Change) не возникает при изменении значения вычисляемого элемента управления, а также, если с помощью макроса или программы VBA установлено значение текстового поля или поля со списком или если значение поля со списком выбрано из списка.*

Некорректная программа обработки данного события может привести к каскадным событиям. Чтобы избежать этого, не следует использовать в настоящей процедуре команды, которые меняют содержимое элемента управления, а также не стоит создавать два и более поля, которые воздействуют друг на друга, например, обновляют друг друга.

Событие **Отсутствие в списке** (NotlnList) возникает в поле со списком, когда пользователь вводит вручную значение в текстовую часть поля, которое отсутствует в списке, и после этого пытается перейти в другое поле или сохранить запись. Для того чтобы данное событие происходило, нужно присвоить свойству **Ограничиться списком** (LimitToList) значение **Yes**. Если это свойство имеет значение **No**, то разрешается ввод в поле данных, не совпадающих ни с одним значением из списка. Процедура обработки настоящего события имеет два параметра: **NewData** и **Response**. Параметр **NewData** содержит введенные данные, а **Response** управляет обработкой события и может иметь значения 0, 1 или 2. Значение 0 позволяет вывести на экран стандартное сообщение о том, что введенные данные отсутствуют в списке, и запретить ввод. Значение 1 позволяет вместо стандартного сообщения вывести специальное сообщение, например, запрашивающее, следует ли сохранить введенное значение. Новые данные при этом не добавляются в список. Значение 2 разрешает добавить новое значение в список. При этом в процедуре обработки данного события нужно добавить значение к источнику строк для поля со списком, после чего поле обновляется, т. к. Access повторно запрашивает источник строк.

Однако, если источником строк для поля со списком является таблица-справочник, простого добавления значения может оказаться недостаточно. Скорее всего, придется вывести специальную форму, в которой пользователь должен будет заполнить все необходимые поля. После сохранения записи в

этой форме новые данные добавляются в список. Типичная ситуация, когда требуются такие действия – добавить нового клиента при выписке ему стандартного документа.

События **Внесены изменения** (Dirty) и **Изменение** (Change) возникают в следующих ситуациях:

– при изменении содержимого текстового поля или поля со списком, при этом изменением может быть любой непосредственно введенный или удаляемый символ;

– при изменении значения свойства **Текст** (Text) элемента управления с помощью макроса или процедуры VBA;

– в элементе управления **Набор вкладок** (Tab Control) при переходе с одной вкладки на другую.

Но, в отличие от события **Изменение** (Change), оно относится к форме. Процедура имеет один параметр: **Cancel**. Если установить его значение равным **True**, то событие будет отменено. Отмена события будет вызывать откат всех изменений в записи, что эквивалентно нажатию клавиши <Esc>. Это событие удобно использовать для проверки, были ли изменения в записи.

Событие **При обновлении** (Updated) возникает при изменении объекта OLE и применяется только к свободным и присоединенным рамкам объекта.

Процедура обработки данного события используется для проверки, были ли данные в объекте OLE изменены после последнего сохранения. Процедура имеет один параметр – **Code**, который указывает, каким образом обновлялся объект, и может иметь значения 0, 1, 2 и 3. Значение 0 указывает, что данные объекта изменены. Значение 1 указывает, что данные объекта сохранялись приложением, создавшим объект. Значение 2 указывает, что файл объекта OLE закрывался приложением, которое его создало. Значение 3 указывает, что файл объекта OLE переименован создавшим его приложением.

Событие **Уход с записи** (RecordExit) происходит всякий раз, когда пользователь пытается выйти (переместить фокус) с текущей записи: перейти к другой записи, закрыть форму, обновить данные в форме и т. д. Процедура обработки этого события может использоваться для проверки корректности данных в текущей записи. Процедура имеет один параметр: **Cancel**. Если установить его значение равным **True**, то можно запретить пользователю покидать текущую запись.

Событие **Отмена** происходит, когда пользователь отменяет изменения, сделанные в текущем поле или текущей записи, например, нажав клавишу [Esc] или кнопку $$ (Undo Field/Record) на панели инструментов. Процедура обработки этого события имеет один параметр: **Cancel**. Установив его значение равным **True**, можно прервать операцию отмены, и тогда данные в текущем поле или записи останутся измененными.

События фокуса происходят, когда форма, отчет или элемент управления в форме получают или теряют фокус, а также когда форма или отчет становятся активными или, наоборот, неактивными.

Событие **Вход** (Enter) происходит перед тем, как элемент управления в форме получает фокус от другого элемента управления в той же форме или когда при открытии формы получает фокус первый элемент управления. Его удобно использовать для вывода на экран каких-либо сведений об этом элементе. Оно происходит до события **Получение фокуса** (GetFocus), но после события **Текущая запись** (Current).

Событие **Выход** (Exit) происходит перед тем, как данный элемент управления передаст фокус другому элементу управления той же формы, но до события **Потеря фокуса** (LostFocus).

Замечание. *События* ***Вход*** *(Enter) и* ***Выход*** *(Exit) не определены для флажков и переключателей в группах, они определены только для группы как целого. События* ***Вход*** *(Enter) и* ***Выход*** *(Exit) не происходят, если фокус переходит к элементу другой формы или отчета. Это отличает их от событий* ***Получение фокуса*** *(GetFocus) и* ***Потеря фокуса*** *(LostFocus).*

Событие **Получение фокуса** (GetFocus) происходит, когда форма или элемент управления формы получают фокус. Элемент управления может получить фокус, только если оно видимо и доступно (его свойства **Вывод на экран** (Visible) и **Доступ** (Enabled) имеют значения **Да**). При этом событие **Получение фокуса** (GetFocus) происходит после события **Вход** (Enter). Форма может получить фокус, только если все поля в ней заблокированы, в противном случае событие **Получение фокуса** (GetFocus) для формы не возникает.

Событие **Потеря фокуса** (LostFocus) происходит каждый раз, когда форма или элемент управления в форме теряют фокус. Событие происходит после события **Выход** (Exit).

Замечание. *События* ***Получение фокуса*** *(GetFocus) и* ***Потеря фокуса*** *(LostFocus) происходят не только, когда фокус перемещается путем действий пользователя (с помощью клавиатуры или мыши), но и в результате вызова метода SetFocus объекта в форме или выполнения макрокоманд* ***ВыделитьОбъект*** *(SelectObject),* ***НаЗапись*** *(GoToRecord),* ***КЭлементуУправления*** *(GoToControl) и* ***НаСтраницу*** *(GoToPage).*

Кроме того, эти события определены для элементов (флажков и переключателей) внутри группы.

Событие **Включение** (Activate) возникает, когда форма или отчет получают фокус, становясь активной формой или отчетом. Это происходит, когда форма или отчет открываются, когда пользователь щелкает мышью на одном из элементов управления, перенося, таким образом, фокус, и когда в программе VBA выполняется метод SetFocus объекта. Форма при этом обязательно должна быть видима. Событие **Включение** (Activate) возникает до события **Получение фокуса** (GetFocus). Это событие удобно использовать для того, чтобы вывести на экран панель инструментов, связанную с формой.

Событие **Отключение** (Deactivate) происходит, когда фокус из формы или отчета переносится на другое окно (таблицы, запроса, формы, отчета, макроса, модуля или окно базы данных). Однако это событие не возникает, когда фокус переходит в диалоговое окно или другое приложение. Событие **Отключение** (Deactivate) возникает после события **Потеря фокуса** (LostFocus).

События клавиатуры происходят в форме и элементе формы, когда пользователь нажимает клавиши на клавиатуре или же выполняется макрокоманда **sendKeys**.

Все события клавиатуры связываются с тем объектом в форме, который имеет в данный момент фокус. Обычно это один из элементов управления. Форма может получить фокус (а значит, и события клавиатуры могут относиться к форме), только если все ее элементы управления заблокированы или невидимы. Если нужно привязать эти события именно к форме, а не к элементу формы, то можно присвоить свойству **Перехват нажатия клавиш** (KeyPreview) для формы значение **Да** (Yes). Тогда все события клавиатуры возникают сначала для формы, а потом уже для элемента управления, имеющего фокус. Это дает возможность запрограммировать реакцию формы на нажатие

определенных клавиш вне зависимости от того, в каком элементе управления формы находится фокус.

Замечание. *События клавиатуры не определены для элементов управления в отчетах, а также для флажков и переключателей в группах. Они определены только для группы в целом.*

События **Клавиша вниз** (KeyDown) и **Клавиша вверх** (KeyUp) возникают всякий раз, когда пользователь нажимает или отпускает клавишу на клавиатуре и при этом фокус находится на элементе управления или форме. Процедуры обработки этих событий используют, когда требуется определить, какую клавишу нажал пользователь – функциональную, клавишу управления курсором, клавишу цифровой панели или комбинацию клавиш с [Shift], [Ctrl] или [Alt]. Они имеют два параметра: **KeyCode** и **Shift**. Параметр **KeyCode** – это целое число, представляющее код нажатой клавиши. Параметр **Shift** позволяет определить, какие сочетания клавиш нажимались: 1 – соответствует [Shift], 2 – [Ctrl], 4 – [Alt], 0 – не использовалось сочетание клавиш. Если же нажималась комбинация клавиш [Shift], [Ctrl], [Alt] в любом сочетании, то параметр **Shift** будет равен сумме значений каждой клавиши.

Событие **Нажатие клавиши** (KeyPress) происходит, если пользователь нажимает и отпускает любую комбинацию клавиш для элемента управления или формы, имеющей фокус. В отличие от событий **Клавиша вниз** (KeyDown) и **Клавиша вверх** (KeyUp), данное событие не происходит, когда нажимаются функциональные клавиши, клавиши управления курсором и клавиши [Shift], [Ctrl] или [Alt]. Кроме того, эти события различны для верхнего и нижнего регистров. Процедура обработки настоящего события имеет один аргумент: **KeyAscii** – целое число, представляющее код нажатой клавиши.

Если пользователь нажимает и удерживает некоторую клавишу, то возникают повторяющиеся события **Клавиша вниз** (KeyDown) и **Нажатие клавиши** (KeyPress) до тех пор, пока он не отпустит клавишу. Тогда возникает событие **Клавиша вверх** (KeyUp).

Если результатом нажатия клавиши является перевод фокуса с одного элемента на другой, то событие **Клавиша вниз** (KeyDown) возникает для первого элемента, а события **Нажатие клавиши** (KeyPress) и **Клавиша вверх** (KeyUp) – для второго.

Если в результате нажатия клавиши появляется диалоговое окно, то возникают события **Клавиша вниз** (KeyDown) и **Нажатие клавиши**, а событие **Клавиша вверх** (KeyUp) не возникает.

События **мыши** происходят, когда какое-либо действие в форме или ее элементе управления выполняется с помощью мыши. События мыши не определены для элементов управления в отчетах, а также для флажков и переключателей в группах, они определены только для группы в целом.

Событие **Нажатие кнопки** – наиболее широко распространенное событие. Оно возникает как в самой форме, так и в элементах управления формы. Для формы событие **Нажатие кнопки** (Click) возникает, когда пользователь щелкает мышью на пустой области формы или на области выделения записи в форме. Для элемента управления событие **Нажатие кнопки** (Click) возникает при щелчке мыши не только на самом элементе, но и на присоединенной к нему надписи. Однако для элемента управления оно возникает не только при щелчке мыши, но и в некоторых других случаях, а именно:

– при выборе элемента из списка, независимо от того, был ли он выбран с помощью мыши или клавиш управления курсором с последующим нажатием [Enter];

– при нажатии клавиши [Пробел], когда фокус установлен на флажке, переключателе или командной кнопке;

– при нажатии клавиши [Enter] в форме, которая содержит кнопку свойства **По умолчанию** (Default) со значением **Да**, тогда именно на эту кнопку по умолчанию устанавливается фокус;

– при нажатии клавиши [Esc] в форме, которая содержит кнопку свойства **Отмена** (Cancel) со значением **Да;**

– при нажатии клавиш доступа, если они связаны с кнопками на форме.

Таким образом, процедуры обработки события **Нажатие кнопки** (Click) для кнопок запускаются независимо от того, каким образом эта кнопка выбрана – щелчком мыши, нажатием клавиши [Enter] или клавиши доступа. Процедура обработки события запускается только один раз. Если требуется, чтобы она запускалась несколько раз, пока кнопка остается нажатой, необходимо использовать свойство **Автоматический повтор** (AtitoRepeat) для кнопки. Если нужно

определить, какой кнопкой мыши выполнялся щелчок, следует использовать события **Кнопка вниз** (MouseDown) и **Кнопка вверх** (MouseUp).

Событие **Двойное нажатие кнопки** (DblClick) происходит после быстрого двойного щелчка любой клавиши на форме или элементе управления, при этом интервал между щелчками не должен превышать предельного времени, заданного в панели управления Windows. Событие **Двойное нажатие кнопки** (DblClick) для формы и элемента управления формы определено так же, как и событие **Нажатие кнопки** (Click). Однако для элементов управления результат этого события зависит от типа элемента управления. По умолчанию двойной щелчок мыши в текстовом поле приводит к выделению слова, а в объекте OLE – запускает этот объект для редактирования. Вводя процедуру обработки для настоящего события, можно переопределить стандартные действия Access. Процедура имеет один параметр: **Cancel**. Если присвоить ему в процедуре значение **True**, то можно отменить это событие.

Замечание. *Двойное нажатие кнопки мыши на элементе управления на самом деле вызывает сразу два события: сначала* ***Нажатие кнопки*** *(Click), а потом –* ***Двойное нажатие кнопки*** *(DblClick).*

Событие **Перемещение указателя** генерируется непрерывно, когда пользователь перемещает указатель мыши по объектам формы. Пока указатель движется в границах объекта, событие **Перемещение указателя** (MouseMove) генерируется для данного объекта, когда указатель попадает на пустую область формы, область выделения записи или полосу прокрутки, генерируется событие **Перемещение указателя** (MouseMove) для формы. Событие возникает также при перемещении формы или элемента управления, например, с помощью процедуры VBA, при неподвижном указателе мыши. Процедура обработки события имеет четыре параметра:

– **Button** – определяет состояние кнопок мыши в момент возникновения события (перемещение указателя может происходить при нескольких нажатых или не нажатых кнопках мыши);

– **Shift** – определяет состояние клавиш [Shift], [Ctrl] и [Alt] в тот момент, когда нажата кнопка, определяемая параметром **Button**;

– **X** и **Y** – текущие координаты указателя мыши в типах.

Событие **Колесико мыши** возникает, когда пользователь перемещает указатель мыши с помощью колесика скроллирования. Процедура обработки события имеет два параметра:

– **Page** – принимает значение **True** при перемещении указателя на другую страницу;

– **Count** – количество линий, на которое переместился указатель при прокрутке формы с помощью колесика мыши.

События **Кнопка вни**з (MouseDown) и **Кнопка вверх** (MonseUp) возникают, когда пользователь нажимает и, соответственно, отпускает кнопку мыши, и, в отличие от событий **Нажатие кнопки** (Click) и **Двойное нажатие кнопки** (DblClick), оно позволяет определить, какая кнопка нажата. Процедуры обработки этих событий имеют четыре параметра: **Button**, **Shift**, **X** и **Y**. Эти параметры аналогичны параметрам процедуры для события **Перемещение указателя** (MouseMove) за исключением первого параметра – **Button**. Так как в данном случае нажимается конкретная кнопка мыши, параметр **Button** определяет, какая это кнопка. Если пользователь нажмет сразу две кнопки, то возникнут отдельно события для первой и для второй кнопок.

Замечание. *Если кнопка мыши была нажата, когда указатель находился на одном из элементов управления формы, то именно к этому объекту будут относиться все остальные события мыши до последнего* ***Кнопка вверх*** *(MouseUp) включительно.*

События **печати** вызываются отчетом и любой из его областей при печати или предварительном просмотре.

Событие **Форматирование** (Format) происходит после того, как отобраны данные для отчета, но перед тем, как фактически форматируется каждый раздел отчета. При этом для раздела данных это событие происходит для каждой записи в отчете, что позволяет при необходимости по-разному форматировать каждую запись. Для заголовка группы в отчете событие возникает для каждой группы. Процедура обработки данного события имеет два параметра – **Cancel** и **FormatCount**. **Cancel** позволяет отменить форматирование данного раздела, для чего нужно присвоить ему значение **True**. Это дает возможность пропускать разделы отчета, не оставляя пустого места на странице. **FormatCount** – счетчик, который считает, сколько раз произошло событие **Форматирование** (Format).

Событие **Возврат** происходит, если при форматировании раздела требуется вернуться к разделу, который уже отформатирован. Оно происходит после события **Форматирование** (Format), но до события **Печать** (Print). Процедура обработки данного события позволяет изменить любое уже выполненное форматирование и обеспечить, таким образом, нужное расположение элементов отчета на странице.

Замечание. *Событие* ***Возврат*** *(Retreat) не определено для верхних и нижних колонтитулов отчета.*

Событие **Печать** (Print) возникает после того, как выполнено форматирование раздела отчета, но до того, как он напечатан. Это событие возникает практически после каждого события **Форматирование** (Format), кроме тех разделов, которые не будут печататься. Так же, как и при форматировании, процедура обработки событий имеет два параметра: **Cancel** и **PrintCount**. **Cancel** позволяет отменить печать текущего раздела или текущей записи в отчете, для чего нужно присвоить ему значение **True**. Однако при этом остается пустое место на странице, поэтому данную процедуру можно использовать, когда изменения не касаются формата страницы отчета. **PrintCount** – счетчик, который считает, сколько раз произошло событие **Печать** (Print).

Событие **Страница** (Page) возникает после форматирования страницы отчета, но до вывода ее на печать, и позволяет с помощью процедуры обработки этого события добавить на страницу некоторые элементы оформления, например, рамку.

Событие **Отсутствие данных** (No Data) возникает после форматирования отчета, но до его вывода на печать (до первого события **Страница** (Page)) и позволяет обнаружить отсутствие записей для отчета; в этом случае печать можно отменить. Процедура обработки данного события имеет один параметр: **Cancel**, которому следует присвоить значение **True**, если нужно отменить печать отчета.

События **фильтра** происходят при применении или удалении фильтра в форме.

Событие **Применение фильтра** (ApplyFilter) возникает во всех случаях, когда пользователь выполняет фильтрацию записей в форме с помощью соответствующих команд меню, контекстного меню или кнопки панели инструментов (применить или удалить фильтр). Программу обработки этого события обычно

используют либо для проверки условия в фильтре, либо для изменения вида формы перед применением фильтра, если требуется скрыть лишние поля или, наоборот, показать скрытые. Программа обработки события имеет два параметра – **Cancel** и **ApplyType**. **Cancel** позволяет отменить операцию фильтрации, если, например, условие сформулировано неправильно, для чего нужно присвоить ему значение **True**. **ApplyType** определяет исполняемое действие и может принимать значения 0, 1 и 2. Значение 0 указывает на удаление фильтра, 1 – на применение фильтра, 2 – на закрытие окна фильтра.

Замечание. *Событие не возникает при выполнении операций фильтрации с помощью макрокоманд* ***ПрименитьФильтр*** *(ApplyFilter),* ***ОткрытьФорму*** *(OpenForm),* ***ПоказатьВсеЗаписи*** *(ShowAllRecords), соответствующих им методов объекта* ***DoCmd****, или при закрытии окна фильтра (любого) с помощью макрокоманды* ***Закрыть****(Close).*

Событие **Фильтрация** (Filter) возникает перед открытием окна фильтра или расширенного фильтра, когда пользователь пытается выполнить команду **Изменить фильтр** (Filter by Form). Использовать это событие очень удобно, если требуется, например, ввести в фильтр некоторые условия по умолчанию или запретить включать в условия отбора некоторые поля. Чтобы запретить включать некое поле в условие отбора в окне фильтра, достаточно скрыть его в процедуре обработки события **Фильтрация** (Filter). Правда, это относится только к окну обычного фильтра, т. к. в окне расширенного фильтра выводятся все поля, в том числе и скрытые. Можно даже заменить стандартное окно фильтра своим собственным, в котором пользователь и будет задавать условия отбора. Процедура обработки события имеет два параметра – **Cancel** и **FilterType**. **Cancel** позволяет отменить открытие стандартного окна фильтра, если вместо него будет выводиться специальная форма, для чего нужно присвоить ему значение **True**. Параметр **FilterType** определяет, какое окно открывается, и может принимать значения 0 или 1. Значение 0 соответствует обычному фильтру, 1 – расширенному фильтру.

События **окна** запускаются при открытии и закрытии форм и отчетов, а также при изменении размеров формы.

Событие **Открытие** (Open) происходит после того, как выполнен запрос, лежащий в основе формы или отчета, но до отображения первой записи или

печати отчета. Процедура обработки этого события имеет один параметр – **Cancel**, при установке которого в значение **True** отменяется открытие формы или отчета. Обычно процедура обработки события **Открытие** (Open) используется для проверки условий и предотвращения открытия формы, т. к. следующее по времени событие **Загрузка** (Load) уже не может быть отменено.

Событие **Закрытие** (Close) является последним перед тем, как форма будет удалена с экрана. Обычно его используют для открытия другой формы. Для отчета событие происходит, когда закрывается режим **Предварительного просмотра** или заканчивается печать отчета. Как и в случае с формой, его можно использовать для определения дальнейших действий пользователя.

Событие **Загрузка** (Load) происходит сразу после события **Открытие** (Open), но в отличие от него не может быть отменено. Обычно его используют для динамического изменения свойств формы или элементов управления перед тем, как форма будет выведена на экран.

Событие **Выгрузка** (Unload) происходит при закрытии формы до события и может быть отменено. Обычно это событие используется для проверки различных условий, которые определяют, можно ли закрывать форму. Процедура обработки этого события имеет один параметр – **Cancel**, при установке которого в значение **True** отменяется закрытие формы.

Замечание. *Если вы используете процедуру обработки события* ***Выгрузка*** *(Unload), в которой параметру* ***Cancel*** *присваивается значение* ***True****, не забудьте явно присвоить ему значение* ***False*** *в случае выполнения всех условий для закрытия формы. Иначе после того как этот параметр будет установлен в* ***True****, форму нельзя будет закрыть никогда.*

Событие **Изменение размера** (Resize) возникает при открытии формы и при изменении ее размеров. Его обычно применяют, если требуется подстроить размер элементов управления под изменяющиеся размеры формы или вычислить заново вычисляемые элементы. Если нужно, чтобы при каждом изменении размеров формы происходило обновление экрана, используйте в процедуре обработки этого события метод **Repaint.**

Замечание. *При создании процедур обработки событий часто возникают сомнения, какое из двух событий –* ***Открытие*** *(Open) или* ***Загрузка*** *(Load) – использовать (или, соответственно,* ***Закрытие*** *(Close) или* ***Выгрузка*** *(Unload)). Рекомендуется*

принимать во внимание следующее соображение. Если требуется возможность отменить событие, то пользуйтесь событиями ***Открытие*** *(Open) и* ***Выгрузка*** *(Unload), в противном случае можно использовать любое.*

Событие **Ошибка** (Error) возникает, когда в процессе обработки формы или отчета ядром Access возникает ошибка. В процедуре обработки этого события можно перехватить стандартное сообщение об ошибке, которое выдает Access, и выдать собственное сообщение. Процедура имеет два параметра – **DataErr** и **Response**. Параметр **DataErr** содержит код ошибки, а параметр **Response** может принимать два значения – 0 и 1. Значение 0 отменяет выдачу стандартного сообщения об ошибке, а 1 – позволяет его отобразить.

Замечание. *Это событие не возникает, когда ошибка встречается в коде VBA.*

Событие **Таймер** (Timer) возникает регулярно через интервал времени, который задается свойством **Интервал таймера** (TimerInterval) формы. Оно позволяет определять действия, которые должны выполняться периодически по сигналу таймера. Обычно используется для регулярных обновлений экрана в многопользовательском приложении, тогда в процедуре обработки события **Таймер** (Timer) нужно использовать метод **Requery**, который будет выполнять повторный запрос источника данных формы.

При написании процедур обработки событий очень важно понимать, в каком порядке они происходят, т. к. приложение Access управляется событиями и результат работы зависит от того, в каком порядке эти процедуры будут выполняться. Рассмотрим порядок возникновения событий в формах и отчетах.

Последовательность событий фокуса для элементов управления в формах

При установке фокуса на элемент управления щелчком мыши, при нажатии клавиши [Tab] или при открытии формы происходят события: **Вход** (Enter) → **Получение фокуса** (GotFocus).

Когда элемент теряет фокус, например, при закрытии формы или переносе фокуса на другой элемент управления той же формы, происходят события: **Выход** (Exit) → **Потеря фокуса** (LostFocus).

Когда вводят или изменяют данные в элементе управления, а затем переходят к следующему элементу управления, возникает следующая цепочка событий: **{Клавиша вниз → Нажатие клавиши → Внесены изменения → Изменение → Клавиша вверх} → До обновления → После обновления → Выход → Потеря фокуса**.

Фигурные скобки в приведенном выражении означают, что выделенная цепочка событий возникает при каждом нажатии клавиши на клавиатуре.

Если ввод выполняется в поле со списком, то в конец последней цепочки, после события **Клавиша вверх**, могут добавиться события **Отсутствие в списке** (NotlnList) и **Ошибка** (Error), и цепочка будет выглядеть следующим образом: **Клавиша вниз → Нажатие клавиши → Внесены изменения → Изменение → Клавиша вверх} → Отсутствие в списке → Ошибка → До обновления → После обновления → Выход → Потеря фокуса.**

Если просто просматриваются записи в форме, то при переходе к каждой новой записи выполняются событие **Текущая запись** (Current) для формы и все события, связанные с установкой фокуса в элементах формы. Если данные в записи меняются, то сохранение изменений происходит только при переходе к следующей записи или при закрытии формы (если нашли нужную запись, изменили и закрыли форму). Поскольку все изменения в записи происходят в элементах управления формы, одновременно будут возникать соответствующие события в элементах формы. Рассмотрим типичную ситуацию, когда в форме выводится запись, в ней перемещаются по элементам управления до нужного элемента, изменяют в нем данные и переходят к следующей записи. В этом случае последовательность возникновения событий будет выглядеть так: **Текущая запись → {Вход → Получение фокуса → Выход → Потеря фокуса} → Вход → Получение фокуса → До обновления → После обновления → До обновления → После обновления → Выход → Потеря фокуса → RecordExit → Текущая запись → Вход → Получение фокуса.**

В этой цепочке для упрощения опущены все события клавиатуры, а фигурные скобки выделяют цепочки событий, которые возникают при переходе между элементами управления формы. При изменении данных события **До обновления** (BeforeUpdate) и **После обновления** (AfterUpdate) происходят сначала для элемента, а затем для формы. Затем последний элемент (в данном случае тот, в котором происходили изменения) теряет фокус, происходит событие **RecordExit** (Выход из записи), выводится следующая запись, и фокус устанавливается на первый элемент в этой записи. События **До обновления** (BeforeUpdate) и **После обновления** (AfterUpdate) всегда возникают непосредственно перед переходом к следующей записи. После этого запись (элемент управления, в котором перед этим находился фокус) теряет фокус.

Замечание. *Если запись сохраняется с помощью команды меню Записи → Сохранить запись (Records → Save Record), то последних событий* ***Выход*** *(Exit) и* ***Потеря фокуса*** *(LostFocus) не происходит. Это полезно знать, т. к. иногда требуется обойти эти события.*

При удалении записи происходят события: **Удаление → До подтверждения Del → После подтверждения Del → Текущая запись → Вход → Получение фокуса**.

Если событие **Удаление** (Delete) отменяется, то остальные события не возникают. После удаления записи фокус переходит на следующую запись, поэтому происходят события **Текущая запись** (Current) формы и **Вход** (Enter) и **Получение фокуса** (GetFocus) первого элемента в этой записи.

Добавление новой записи осуществляется после того, как пользователь вводит первый символ новой (пустой) записи. При этом события происходят в следующем порядке: **Текущая запись → Вход → Получение фокуса → До вставки → Изменение → До обновления → После обновления → После вставки.**

В этой цепочке пропущены все события клавиатуры и **До обновления** (BeforeUpdate) и **После обновления** (AfterUpdate) для элементов формы, т. к. они описаны выше. Событие **Изменение** (Change) происходит, если первый символ новой записи вводится в текстовое поле.

При открытии формы происходит следующая цепочка событий: **Открытие → Загрузка → Изменение размера → Включение → Текущая запись → Вход → Получение фокуса.**

События Вход (Enter) и **Получение фокуса** (GetFocus) возникают для первого элемента в первой записи. Если форма не имеет видимых или доступных элементов управления, то последовательность событий несколько иная: **Открытие → Загрузка → Изменение размера → Включение → Получение фокуса → Текущая запись.**

В этом случае событие **Получение фокуса** (GetFocus) относится к форме. При закрытии формы последовательность событий следующая: **Выход → Потеря фокуса → Выгрузка → Отключение → Закрытие.**

То есть сначала теряет фокус последний элемент в форме, а затем выполняются события для формы. Если в форме не было видимых или доступных

элементов, то последовательность другая: **Выгрузка → Потеря фокуса → Отключение → Закрытие**.

Если из открытой формы открывают другую форму, то сначала в открываемой форме происходят события **Открытие** (Open), **Загрузка** (Load) и **Изменение размера** (Resize), и только после этого в первой форме произойдет событие **Отключение** (Deactivate): **Открытие** (ф2) → **Загрузка** (ф2) → **Изменение размера** (ф2) → **Отключение** (ф1) → **Включение** (ф2) → **Текущая запись** (ф2).

Это дает возможность проконтролировать открытие второй формы. Кроме того, событие **Включение** (Activate) происходит каждый раз, когда форма получает фокус, а события **Открытие** (Open) и **Загрузка** (Load) не происходят, если форма уже открыта, даже если переход в эту форму выполняется с помощью макрокоманды **ОткрытьФорму** (OpenForm). Событие **Отключение** (Deactivate) для формы не происходит, если фокус переносится в диалоговое окно.

Если открываемая форма содержит подчиненные формы, то сначала загружаются подчиненные формы и осуществляются все события в них, которые обычно происходят при открытии формы, кроме события **Включение** (Activate) – оно не возникает. Затем загружается главная форма в обычном порядке и для нее выполняется событие **Включение** (Activate).

Аналогично при закрытии такой формы сначала выгружаются все подчиненные формы, но в них не возникает событие **Отключение** (Deactivate). События при этом происходят в следующем порядке:

1. События для элементов подчиненной формы, например, **Выход** (Exit) и **Потеря фокуса** (LostFocus).

2. События элементов управления в главной форме, в том числе для того элемента, который содержит подчиненную форму.

3. События для главной формы.

4. События для подчиненной формы.

При нажатии и отпускании любой клавиши на клавиатуре в том случае, если фокус находится в одном из элементов управления формы, возникает следующая цепочка событий: **Клавиша вниз → Нажатие клавиши → Клавиша вверх**.

При щелчке мышью на элементе управления формы, соответственно: **Кнопка вниз → Кнопка вверх → Нажатие кнопки.**

В отчетах события возникают только для самого отчета и его разделов. Для полей отчета события отсутствуют. При выводе отчета на печать события обычно возникают в следующей последовательности: **Открытие → Включение → {Формат → Печать} → Закрытие → Отключение.**

События **Форматирование** (Format) и **Печать** (Print) относятся к разделам отчета и возникают при обработке каждого раздела. Дополнительными событиями могут быть события:

1. **Отсутствие данных** (NoData), которое возникает перед первым событием **Печать** (Print), если источник записей не содержит записей.

2. **Страница** (Page), которое возникает после всех событий **Печать** (Print) для страницы отчета.

3. Событие **Возврат** (Retreat) возникает в процессе форматировании страницы отчета, когда требуется вернуться к предыдущему разделу, перед событиями **Печать** (Print) для страницы.

7.6. Создание процедур обработки событий

Рассмотрим, как создавать процедуры обработки событий. Для большинства элементов управления формы, а также самой формы и отчета, стандартный набор действий следующий:

1. Откройте форму в режиме **Конструктора**. Если при этом окно свойств отсутствует на экране, щелкните на кнопке Свойства (Properties) на панели инструментов.

2. Выберите нужный элемент управления (или щелкните мышью на маленьком черном квадрате в верхнем левом углу формы, тогда выберется вся форма). В окне свойств отобразятся свойства выбранного элемента.

3. Откройте вкладку События (Events).

4. Выберите событие, для которого будет создаваться процедура обработки, и щелкните по нему правой кнопкой мыши.

5. Выберите из контекстного меню (рис. 7.7) пункт Построить (Build). В открывшемся диалоговом окне **Построитель** (Choose Builder) выберите из списка

элемент **Программы** (Code Builder) и нажмите кнопку ОК. Откроется окно редактора VBA, в котором появятся первая и последняя строки процедуры (рис. 7.8).

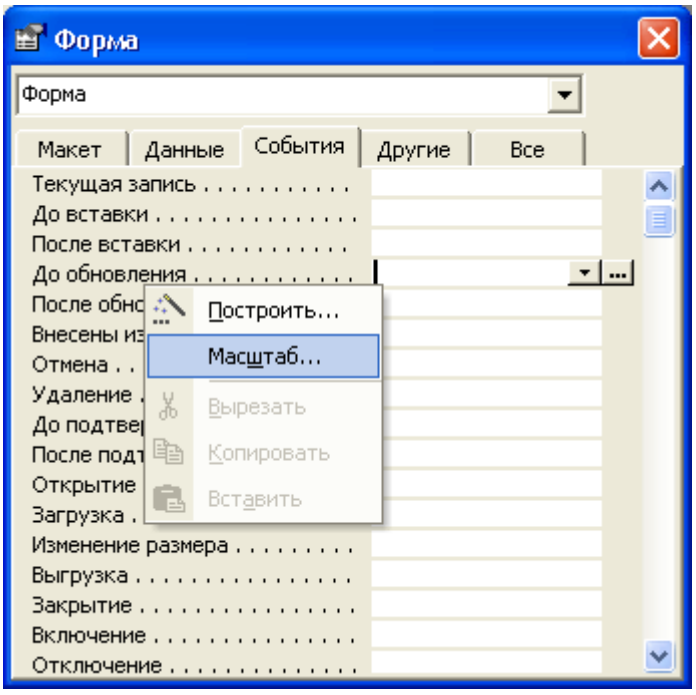

Рис. 7.7. Диалоговое окно событий формы

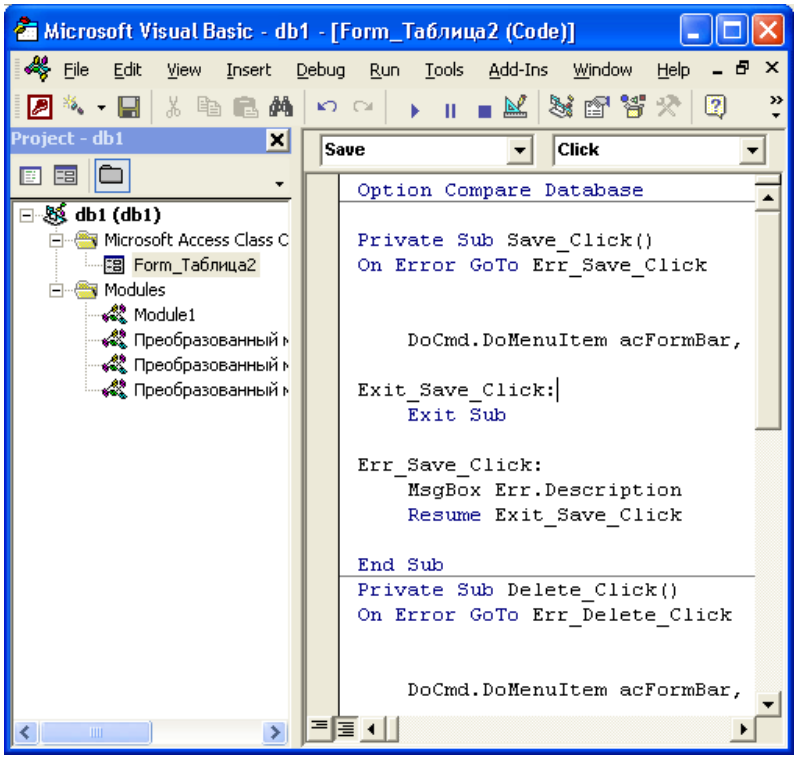

Рис. 7.8. Заготовка процедуры обработки события

Если процедура обработки выбранного события имеет аргументы, они будут также присутствовать в заголовке процедуры (рис. 7.9).

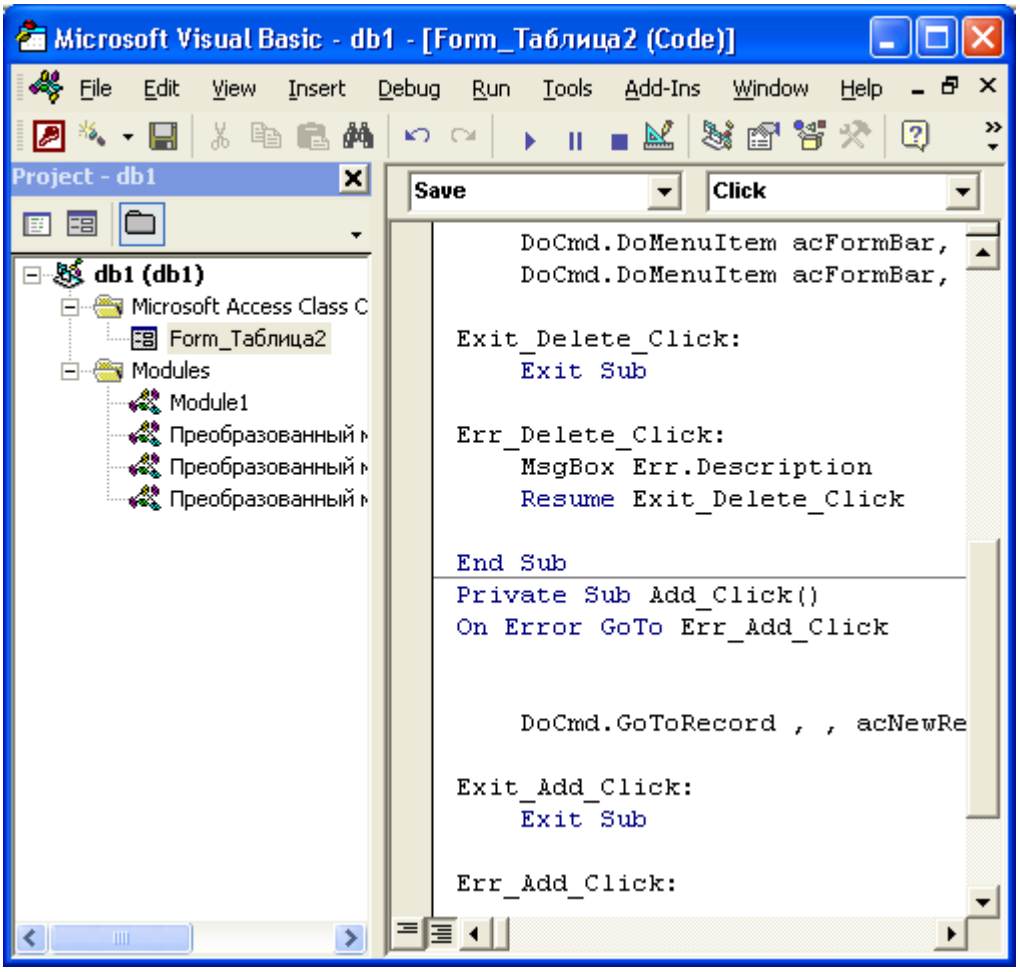

Рис. 7.9. Процедура обработки событий с аргументом **Cancel**

Пример синхронизации данных в двух связанных формах с помощью процедуры обработки события **Текущая запись** (Current) можно увидеть в модуле формы **Поставщики** (Supplier) (база данных «Борей»). В форме есть кнопка Просмотр товаров (Review Products), при нажатии на которую выводится форма **Список товаров** (Product List), показывающая товары данного поставщика. Естественно, что при переходе к новой записи в форме **Поставщики** (Supplier), записи в форме **Список товаров** (Product List) тоже должны быть сменены. Процедура выглядит следующим образом:

```
Private Sub Form_Current()
On Error GoTo Err_Form_Current
' Отображение товаров текущего поставщика при открытии формы "Список ' товаров".
Dim strDocName As String
Dim strLinkCriteria As String
strDocName = "Список товаров"
strLinkCriteria = "[КодПоставщика] = Forms![Поставщики]![КодПоставщика] "
```

```
If IsNull(Me![Название]) Then
Exit Sub
Elself IsLoaded("Список товаров") Then
DoCmd.OpenForm strDocName, , , strLinkCriteria
End if
Exit_Form_Current:
Exit Sub
Err_Form_Current:
MsgBox Err.Description
Resume
Exit_Form_Current
End Sub
```

Если текущая запись в форме **Поставщики** (Supplier) пустая, т. е. добавляется новая запись, то сразу выполняется выход из процедуры. Если текущая запись отображает конкретного поставщика, то проверяется, загружена ли форма **Список товаров** (Product List). Если форма загружена, то меняется набор записей в ней. Делается это с помощью макрокоманды **ОткрытьФорму** (OpenForm) с соответствующим условием отбора записей. При этом на самом деле форма не открывается, просто повторно запрашивается источник данных.

Проверка дублирования значений первичного ключа задается на уровне таблицы, т. е. если поле в таблице определено как первичный ключ, значение свойства **Индексированное поле** (Indexed) автоматически устанавливается равным **Yes** (No Duplicates) – **Да** (Совпадения не допускаются). Однако эта проверка выполняется только тогда, когда запись сохраняется в базе данных. Если значение ключевого поля вводится в форме пользователем, как, например, в таблице **Клиенты** (Customers), то эту проверку лучше выполнить сразу после ввода данных в это поле. Наиболее подходящим событием для этого является событие **До изменения** (Before Update). В форме **Клиенты** (Customers) элемент управления **КодКлиента** (CustomerID) содержит идентификатор клиента. Событие **До изменения** (Before Update) этого поля обрабатывается с помощью макроса **Клиенты** (Customers.ValidateID), который выполняет необходимую проверку. Рассмотрим, как можно обработать это событие с помощью процедуры VBA. Данная процедура может выглядеть следующим образом:

```
Private Sub КодКлиента_BeforeUpdate (Cancel As Integer)
Dim rs As Recordset
Set rs = CurrentDB.Openrecordset("Клиенты", dbOpenTable)
```

```
rs.Index = "PrimaryKey"
rs.Seek "=", Me!КодКлиента
if Not rs.NoMatch Then
MsgBox "Клиент с таким идентификатором уже существует в базе"
Cancel = True
End If
rs .Close
End Sub
```

Поиск записи со значением ключа, совпадающим с введенным значением поля **КодКлиента** (CustomerlD), выполняется с помощью метода **Seek** объекта **Recordset**. Этот метод обеспечивает быстрый поиск необходимой записи. Применить его можно только к набору записей табличного типа, поэтому при создании этого набора записей используется внутренняя константа **dbOpenTable.** Если такая запись найдена, свойство **NoMatch** объекта **Recordset** принимает значение **False**. В этом случае процедура выведет сообщение, что такой пользователь уже существует и присвоит значение **True** аргументу **Cancel**. Это позволяет отменить обновление значения элемента управления. Если значение свойства **NoMatch** объекта **Recordset** равно **True**, процедура закрывает набор записей (рекомендуется не забывать это делать) и завершает свою работу.

Для обработки событий в формах и отчетах используют процедуры типа **Sub** (подпрограммы) или макросы. Однако иногда можно и даже нужно использовать функции. Дело в том, что если в рамках одной формы делается множество однотипных задач, то лучше создать одну процедуру – функцию для выполнения этих задач, описать ее на уровне модуля формы, т. е. в разделе **General**, а потом вызывать из любого места в форме. Если это необходимо, для такой функции определяется один или несколько параметров, которые передаются при вызове данной функции. И хотя значение, возвращаемое функцией, не используется (а обычно и не определяется), применение ее оправдано не только потому, что требуется писать меньше строк кода, а главным образом потому, что минимизация кода в модуле формы ускоряет ее открытие.

Если же идентичные задачи решаются в разных формах, например, одна и та же реакция предусматривается для одинаковых кнопок в разных формах, то такую функцию нужно написать в стандартном модуле базы данных.

Замечание. *Если форма сложная, например, содержит большое количество элементов управления, в том числе поля со списками, то в целях повышения ее быстродействия рекомендуется удалить модуль формы, а все процедуры обработки событий заменить функциями, которые следует при этом вынести в стандартный модуль.*

Допустим, необходимо отобразить в форме список каких-либо объектов, например, счетов клиента. Обычно это делается в подчиненной форме табличного или ленточного вида, в которой отображаются наиболее важные характеристики счета: номер счета, дата выписки, наименование клиента, сумма. Необходимо дать возможность пользователю при желании посмотреть тот счет, который он выбрал, установив курсор на соответствующую строчку. Это можно сделать, создав процедуру открытия формы счета с нужной записью при двойном щелчке мыши на строчке счета. Естественно форма должна открываться при щелчке в любом поле подчиненной формы. Вместо того чтобы в каждом поле подчиненной формы на событие **Двойное нажатие** (DblClick) создавать процедуру обработки событий, можно создать одну функцию и присоединить ее ко всем полям. Функция будет выглядеть так:

```
Private Function Order_DblClick()
Dim strCriteria As String
On Error Goto Err Order DblClick
strCriteria = "КодЗаказа = Forms![Заказы клиента]!_ [Заказы клиента подчиненная].form!КодЗаказа"
DoCmd.OpenForm "Заказы", acNormal, , strCriteria
Exit_Order_DblClick: Exit Function
Err_Order_DblClick: ; MsgBox Err.Description Resume Exit_Order_DblClick
End Function
```

Присоединить эту функцию к событию элемента управления формы можно так, как показано на рис. 7.10. Это сделано для формы **Заказы клиента** (Customer Orders) приложения (NorthWind).

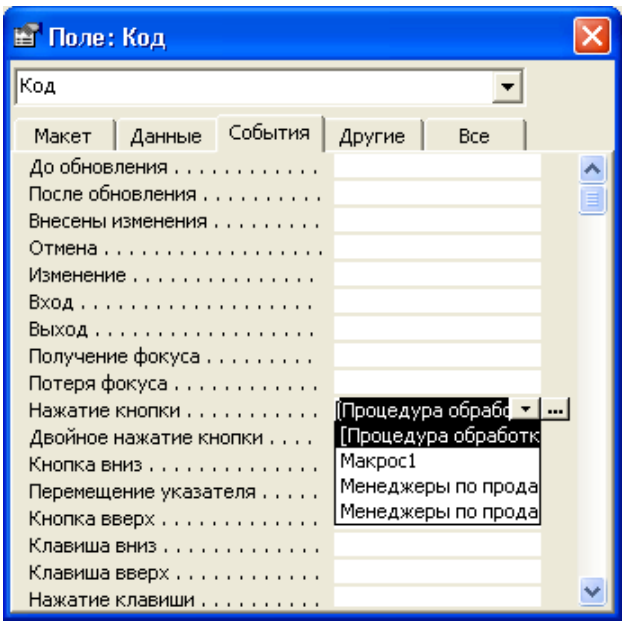

Рис. 7.10. Использование функции при обработке событий в форме

В данном разделе были приведены основные сведения, необходимые для программирования приложений в Access, рассмотрено несколько примеров использования процедур обработки событий в формах. Овладев основным инструментарием и получив необходимые навыки, вы сможете сами создавать процедуры обработки событий в приложениях Access.

8. БАЗЫ ДАННЫХ В ГРАЖДАНСКОЙ АВИАЦИИ

Деятельность авиапредприятий в современных условиях требует разработки новых методов управления, отражающих необходимость реализации потребностей потребителей услуг гражданской авиации (ГА) в обеспечении качества воздушных перевозок, авиационных работ и услуг на всех этапах производственного цикла обеспечения и производства полетов.

К таким методам в настоящее время относится разработка и применение системы качества авиапредприятия, определяющей системный комплексный подход к организации работ по обеспечению и управлению качеством в деятельности авиапредприятия.

Под понятием «качество» понимается совокупность характеристик деятельности, процессов, продукции, организации, системы, отдельного лица или любой комбинации из перечисленных объектов, относящихся к их способности

удовлетворить установленные и предполагаемые требования (ИСО 8402:1994 (E/F/R)).

Под *требованиями к качеству в ГА* принимаются требования по удовлетворению потребностей как внешних, так и внутренних потребителей гражданской авиации (в том числе пассажиров, грузоотправителей, заказчиков на авиационные работы и т. д.) и общества в целом.

Требования к качеству выражаются в определении характера этих потребностей или их переводе в набор количественно или качественно установленных параметров или характеристик объектов для определения возможности их реализации или проверки. Одним из основных критериев обеспечения качества в гражданской авиации является обеспечение безопасности полетов.

Требования стандартов и рекомендуемой практики (SARPS) международной организации гражданской авиации (ИКАО) включают требование к созданию баз данных для регистрации, хранения, анализа и предоставления информации о безопасности авиационной деятельности поставщиков обслуживания. Согласно идеологии SARPS ИКАО, а также действующему воздушному законодательству, указанные требования распространяются также на другие организации, осуществляющие сбор, хранение, обработку, рассылку информации и другие процедуры управления информацией о жизненном цикле гражданской авиационной техники (АТ).

Стандарт (ГОСТ Р 55859-2013. Воздушный транспорт. Система менеджмента безопасности авиационной деятельности. Базы данных. Создание баз данных систем менеджмента безопасности авиационной деятельности поставщиков обслуживания):

– содержит требования к базам данных систем менеджмента безопасности авиационной деятельности (СМБ АД) поставщиков обслуживания и других заинтересованных организаций;

– устанавливает основные принципы разработки баз данных СМБ АД;

– содержит требования к унификации форм представления информации, которые обеспечивают интеграцию баз данных в единое информационное пространство (ЕИП) и информационный обмен между всеми заинтересованными организациями.

Настоящий стандарт в отношении баз данных СМБ АД поставщиков обслуживания содержит описание баз данных и определяет методы управления данными и ресурсы для интеграции баз данных в ЕИП. Описание каждой базы данных включает ее область применения, состав, принципы построения, необходимые идентификаторы. Методы управления базой данных определяются применяемой системой управления, включающей требования к организации базы данных и требования к выполняемым функциям. ЕИП представляет собой интегрированную информационную структуру, в которой циркулирует информация, относящаяся к безопасности авиационной деятельности. Принципы построения ЕИП позволяют осуществлять интеграцию баз данных как поставщиков обслуживания, определенных SARPS ИКАО, так и других заинтересованных субъектов, например, уполномоченных органов и других организаций. Ресурсы для интеграции базы данных в ЕИП предусматривают перевод информации в единые форматы, а также обеспечение доступа к этой информации.

8.1. Требования к базам данных систем менеджмента безопасности авиационной деятельности поставщиков обслуживания

В соответствии с SARPS ИКАО поставщики обслуживания должны разработать и внедрить собственные СМБ АД. В рамках этих систем поставщики обслуживания должны обеспечить:

– сбор информации о фактических и потенциальных недостатках в обеспечении безопасности полетов;

– обязательное и добровольное представление данных об уровне и состоянии безопасности полетов и об инцидентах;

– доступ государственных полномочных органов к данным по безопасности полетов.

Требования к управлению безопасностью полетов включают в себя сбор и анализ информации о безопасности полетов поставщиков обслуживания, оперативный и защищенный обмен этой информацией и сохранение информации в создаваемой базе данных.

Источниками накапливаемой в отраслевых базах данных информации являются результаты непрерывного эксплуатационного мониторинга результатов деятельности поставщиков обслуживания. Эти данные классифицируются по активным угрозам и факторам рисков возникновения неблагоприятных ситуаций и составляют информационную основу для проактивного и прогностического методов управления безопасностью полетов, осуществляемых поставщиками обслуживания.

Все отраслевые базы данных должны создаваться на основе соответствующих отраслевых норм и правил и с использованием совместимой с ADREP системы кодирования и представления данных в соответствии с требованиями к системе ADREP.

Примеры:

1. Автоматизированная система обеспечения безопасности полетов воздушных судов гражданской авиации Российской Федерации (АСОБП).

2. Информационно-аналитическая система мониторинга летной годности воздушных судов (НАС МЛГВС) (ГОСТР S4080).

В целях проведения эффективного анализа информации о недостатках в обеспечении безопасности полетов и определения необходимых действий по обеспечению заданного уровня безопасности полетов создается государственная база данных, которая согласно SARPS ИКАО может включать в себя одну или несколько интегрированных отраслевых баз данных. Источниками информации для государственной базы данных могут быть свои системы представления данных об инцидентах, а также другие системы сбора и обработки данных.

Накапливаемая в государственной базе данных информация составляет информационную основу для создания государственной системы контроля за обеспечением безопасности полетов, которая предназначена для определения и реализации предупредительных и любых других необходимых действий по повышению уровня безопасности полетов.

8.2. Назначение баз данных

Назначение баз данных – обеспечение ввода, сохранения, первичной обработки специализированной информации и вывода данных в соответствии с

запросом. Специализация информации связана с профилем деятельности конкретного поставщика обслуживания.

Хранимая в базах данных информация предназначена для:

– идентификации угроз и факторов риска;

– измерения и ранжирования рисков;

– мониторинга и анализа деятельности поставщиков обслуживания;

– управления рисками для обеспечения заданных уровней безопасности;

– формирования управляющих воздействий на текущие реальные и прогнозируемые возможные результаты деятельности;

– сохранения результатов расследования авиационных происшествий и инцидентов в качестве базы знаний;

– предоставления в текущем времени необходимой информации о безопасности по запросам заинтересованных организаций и/или по каналам телеметрии для анализа;

– документационного обеспечения деятельности поставщика обслуживания;

– анализа данных об авиационном персонале и корректировки обучающих программ;

– анализа операционной (эксплуатационной) деятельности поставщика обслуживания, в том числе производственного планирования, материально-технического обеспечения и т. п.

8.3. Функции баз данных

Функциями баз данных является ввод, хранение и вывод выборки данных, произведенной и отформатированной в соответствии с поступившим запросом.

База данных обеспечивает информационную составляющую выполняемых поставщиком обслуживания производственных функций в задачах управления безопасностью полетов. Состав функций определяется производственными задачами и факторами риска для безопасности полетов и может изменяться в соответствии с изменением количества факторов риска для безопасности полетов, управление которыми осуществляет в процессе своей производственной деятельности конкретный поставщик обслуживания.

Функциями базы данных СМБ АД организаций по техническому обслуживанию и ремонту АТ могут являться:

– создание и актуализация информационного образа воздушного судна;

– актуализация эксплуатационной документации в электронном виде;

– ведение пономерной документации компонентов воздушного судна;

– учет инцидентов и авиационных происшествий;

– мониторинг ресурсного и технического состояния воздушных судов, включая их компоненты, в том числе: оценка аутентичности компонентов воздушных судов, учет информации о функциональных отказах систем воздушных судов, отказах компонентов воздушных судов и других неисправностей, учет наработки воздушных судов и их компонентов и т. д.

С целью обеспечения полноты, достоверности и качества информации базы данных должны включать системы входного и логического контроля. Необходимо, чтобы алгоритмы контроля информации обеспечивали ввод в базу данных только прошедшей контроль информации.

8.4. Задачи, решаемые с использованием баз данных

Системы менеджмента безопасности авиационной деятельности с использованием информации баз данных должны в реальном времени решать следующие задачи:

– непрерывный мониторинг производственной деятельности поставщика обслуживания и сбор статистической информации о результатах этой деятельности;

– анализ накопленной информации для выявления факторов опасности на любом этапе производственной деятельности поставщика обслуживания;

– оперативный доступ персонала поставщика обслуживания к информационным системам и их ресурсам, взаимодействие и консультации с другими заинтересованными организациями;

– информационное обеспечение решений по планированию и формированию предупреждающих мер на основе установленных методов управления безопасностью полетов поставщика обслуживания.

8.5. Требования к отказоустойчивости

Отказоустойчивость БД обеспечивается программными методами защиты данных от потерь и искажений, а также резервированием компонентов серверного оборудования.

Для предотвращения потерь данных должно быть настроено автоматическое создание резервных копий БД на внешние носители.

Основным программным методом, гарантирующим ссылочную целостность данных в случае сбоев, является использование транзакций, поддержка которых должна быть реализована на уровне СУБД. В случае отказа аппаратного обеспечения журнал транзакций позволяет также восстановить данные, измененные с момента создания последней резервной копии.

Аппаратное резервирование в простейшем случае должно обеспечиваться применением избыточного массива независимых жестких дисков (RAID). Если невозможно обеспечить восстановление сервера БД после отказа за допустимое время простоя БД, то необходимо обеспечить дублирование серверов БД с использованием общей системы хранения данных или настройкой репликации данных на сервер, находящийся в горячем резерве.

Для минимизации времени восстановления БД после сбоев и отказов необходимо наличие разработанного регламента восстановления данных из резервных копий и журнала транзакций.

8.6. Требования к физической безопасности

Помещения, в которых располагается сервер БД и хранятся резервные копии данных, должны исключать возможность доступа неавторизованных лиц и быть оборудованы охранно-пожарной сигнализацией и средствами видеонаблюдения. В целях защиты данных от утраты при пожаре и других стихийных бедствиях, сервер БД и хранилище резервных копий должны быть территориально разнесены.

Сервер БД должен обеспечиваться резервным электропитанием на время, достаточное для корректного завершения его работы. На сервере должно быть установлено программное обеспечение, выполняющее его остановку с сохране-

нием хранимых в оперативной памяти данных в случае пропадания централизованного электропитания.

8.7. Требования к информационной безопасности

БД СМБ АД должна иметь разграничение доступа на основе проверки прав пользователя, реализованного как на уровне СУБД, так и на уровне информационной системы в целом.

Защита от несанкционированного доступа должна обеспечиваться не только для пользовательского интерфейса, но и для всех систем нижнего уровня: сервера приложений, веб-сервера, СУБД, операционной системы, локальной сети оператора БД.

Приложение, взаимодействующее с БД, должно обеспечивать фильтрацию запросов для предотвращения внедрения SQL кода (SQL injection).

Локальная сеть оператора БД должна быть защищена межсетевыми экранами. В случае передачи данных между сервером приложений, сервером БД и устройствами резервного копирования по открытым каналам следует обеспечить шифрование трафика.

Для предотвращения утечки данных необходимо осуществлять контроль над созданием и хранением резервных копий, а также над использованием вспомогательных и временных БД, задействованных при отладке и тестировании приложений.

В качестве примера системы менеджмента качества рассмотрим систему управления безопасностью полетов.

Структурная схема системы управления безопасностью полетов (СУБП) приведена на рис. 8.1.

На вход системы поступают информационные потоки, характеризующие состояние воздушного судна (ВС), экипажа и персонала, внешней среды.

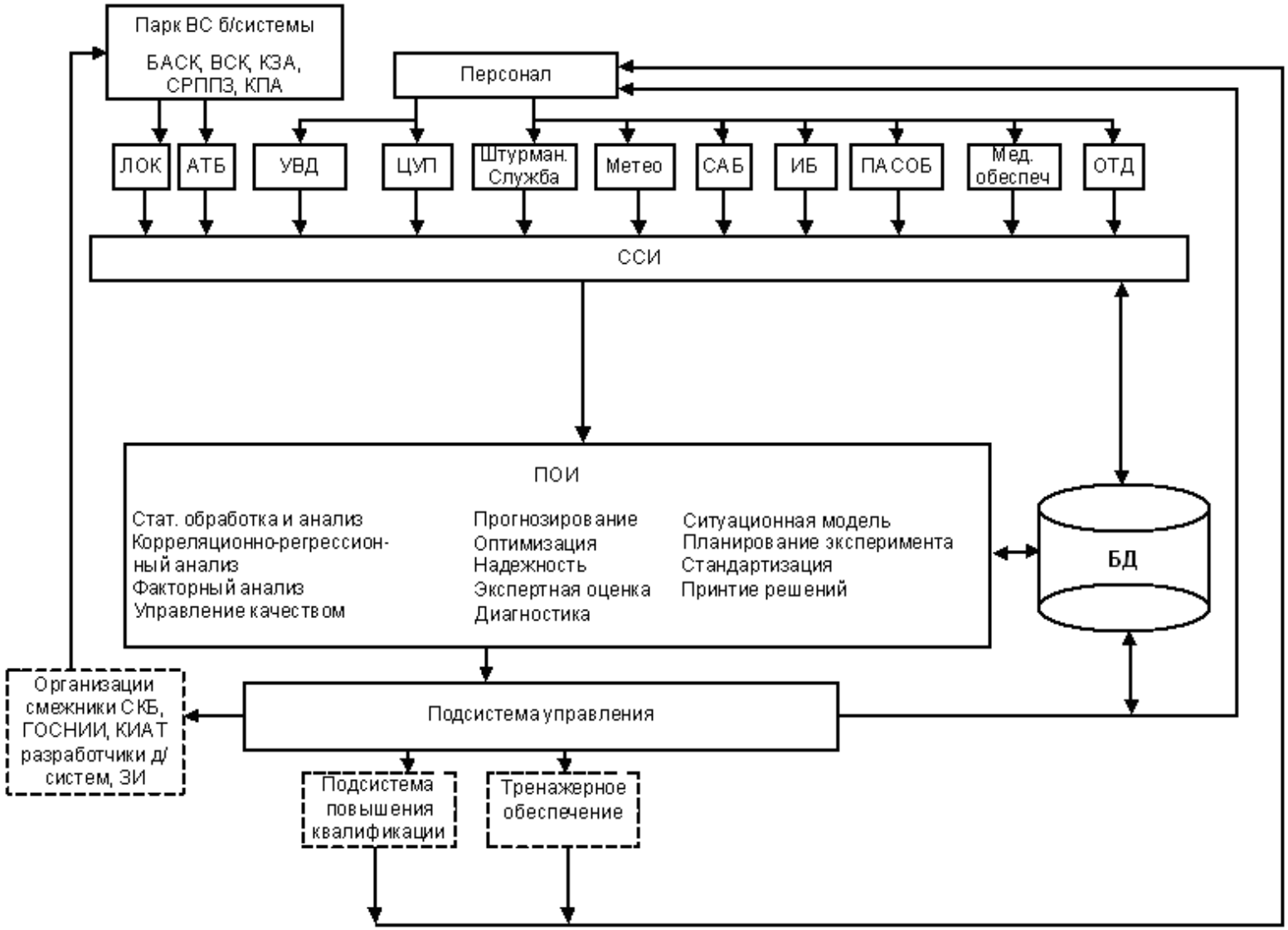

Рис. 8.1. Структурная схема системы управления безопасностью полетов

Рассмотрим источники поступающей информации.

На борту ВС информация о полете записывается комплексом записывающей аппаратуры (КЗА). По результатам каждого полета ВС лаборатория объективного контроля (ЛОК) расшифровывает бортовые регистраторы (черные ящики) и анализирует техническое состояние ВС и технику пилотирования каждого экипажа.

Авиационно-техническая база (АТБ) проводит обслуживание и ремонт парка ВС. Персонал АТБ анализирует показания системы встроенного контроля (ВСК) бортовых систем. В случае необходимости могут проводиться наземные испытания для определения конкретного отказа. Все работы на АТБ выполняются в наземных условиях.

Отдел технической документации (ОТД) контролирует состояние технической документации ВС.

Источником информации о выполненном полете является центр управления полетом (ЦУП) или командно-диспетчерский пункт (КДП). В ЦУП и КДП проводится разбор полетов, анализируются все особые ситуации. Здесь же анализируется работа штурманской службы и службы управления воздушным движением (УВД). Проверке подвергается правильность совместных действий операторов УВД и экипажа ВС.

В СУБП поступает информация о состоянии здоровья и о психологическом состоянии экипажа от службы медицинского обеспечения (МО). По этой информации планируются профилактические мероприятия с летным составом и персоналом авиационного предприятия.

В случае возникновения аварийных происшествий в СУБП поступает информация от службы поиска, аварийного спасания и обеспечения безопасности.

В качестве состояния внешней среды в СУБП поступает информация от службы авиационной безопасности и метеослужбы (МЕТЕО).

Весь объем информации собирается подсистемой сбора информации (ССИ) и вносится в БД для хранения и выдачи подсистеме обработки информации (ПОИ).

Задачи, которые решает ПОИ: статистическая обработка и анализ данных, корреляционно-регрессионный анализ, факторный анализ, управление качеством по критерию, управление рисками, прогнозирование, оптимизация, оценка надежности, экспертные оценки, диагностика, ситуационное моделирование.

Подсистема управления осуществляет планирование эксперимента (активное и пассивное), стандартизацию, принятие решений.

Персонал подсистемы управления готовит мероприятия по выдаче предложений на конструктивные изменения ВС для завода-изготовителя ВС, для опытно-конструкторского бюро (разработчика ВС), для научно-исследовательского института авиационной технологии, ГОСНИИ ГА, для разработчиков бортовых систем и других смежных организаций.

В случае, если подсистема управления и входящий в нее персонал выявляют некоторые типичные ошибки конкретного пилота, то даются рекомендации, по которым пилот должен пройти курсы повышения квалификации или тренировочное обучение. В необходимых случаях такое же повышение квалификации может пройти любой сотрудник.

Мероприятия по поддержанию требуемого уровня работоспособности выполняются постоянно в течение всей эксплуатации данного типа ВС.

На основе рассмотренной структуры возможно построение информационной системы, основу которой составляет база данных. Большое внимание при проектировании базы данных следует уделить изучению документооборота, выделению необходимых объектов предметной области и их атрибутов, что позволит построить инфологическую модель предметной области, провести нормализацию и реализовать базу данных с помощью выбранной СУБД.

ЗАКЛЮЧЕНИЕ

В данном пособии рассмотрены основы проектирования и создания баз данных. Необходимо отметить, что важным этапом проектирования является анализ предметной области, который может занять достаточно продолжительное время и потребовать привлечения специалистов в рассматриваемой предметной области. От того, насколько тщательно проведен анализ, зависит практическая ценность созданного приложения и время разработки. При небрежно проведенном анализе велика вероятность возврата от уже реализованной базы данных к начальному этапу.

В пособии не рассматривались вопросы разделения баз данных и создания клиент-серверных приложений, что является следующим шагом создания полноценных сетевых приложений.

Авторы надеются, что рассмотренный материал поможет всем желающим в первоначальном освоении СУБД Access.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бекаревич, Ю. Б. СУБД Access для Windows 95 в примерах / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб. : BHV – Санкт-Петербург, 1997. – 400 с.

2. Дженнинг, Р. Использование Microsoft Access 2000 : учеб. пособие / Р. Дженнинг ; пер. с англ. – М. : Вильямс, 2000. – 1152 с.

3. Андерсен, В. Microsoft Office 2002 / В. Андерсен ; пер. с англ. – М. : АСТ : Астрель, 2005. – 641 с.

4. ГОСТ Р 55859-2013. Воздушный транспорт. Система менеджмента безопасности авиационной деятельности. Базы данных. Создание баз данных систем менеджмента безопасности авиационной деятельности поставщиков обслуживания. – Введ. 2015–01–01.

Учебное пособие

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ, БАЗЫ ДАННЫХ

РАБОТА С СУБД ACCESS

Составители: ТОЛСТОВ
Константин Анатольевич
ЛЕБЕДЕВ
Алексей Михайлович

ISBN 978-5-7514-0230-3

Редактор Е. А. Нестерова
Компьютерная верстка Н. П. Красильниковой

Подписано в печать 14.05.2015. Формат 60×90/16. Бумага офсетная.
Печать трафаретная. Усл. печ. л. 6,81. Уч.-изд. л. 6,18.
Тираж 30 экз. Заказ № 400.

РИО и типография УВАУ ГА(И). 432071, Ульяновск, ул. Можайского, 8/8.