**ЛЕКЦИОННОЕ ЗАНЯТИЕ**

**Тема: СУБД** **Access.** **Объекты БД. Создание проекта БД. Типы данных в Access. Организация межтабличных связей в БД. СУБД Access. Объекты БД. Создание проекта БД. Типы данных в Access. Организация межтабличных связей в БД.**

1. **Основные понятия баз данных**
	1. **Базы данных и системы управления базами данных**

*База данных — это организованная структура, предназначенная для хранения информации*. Поскольку данные и информация — понятия взаимосвязанные, но не тождественные, то следует заметить некоторое несоответствие в этом определении. Его причины чисто исторические. В те годы, когда формировалось понятие баз данных, в них действительно хранились только данные. Однако сегодня большинство систем управления базами данных (СУБД) позволяют размещать в своих структурах не только данные, но и методы (то есть программный код), с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами. Таким образом, мы можем говорить, что в современных базах данных хранятся отнюдь не только данные, но и информация.

Это утверждение легко пояснить, если, например, рассмотреть базу данных крупного банка. В ней есть все необходимые сведения о клиентах, об их адресах, кредитной истории, состоянии расчетных счетов, финансовых операциях и т. д. Доступ к этой базе имеется у достаточно большого количества сотрудников банка, но среди них вряд ли найдется такое лицо, которое имеет доступ ко всей базе полностью и при этом способно единолично вносить в нее произвольные изменения. Кроме данных, база содержит методы, и средства, позволяющие каждому из сотрудников оперировать только с теми данными, которые входят в его компетенцию. В результате взаимодействия данных, содержащихся в базе, с методами, доступными конкретным сотрудникам, образуется информация, которую они потребляют и на основании которой в пределах собственной компетенции производят ввод и редактирование данных.

С понятием базы данных тесно связано понятие *системы управления базой данных.* Это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации. Под *визуализацией информации* базы понимается отбор отображаемых данных в соответствии с заданным критерием, их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

В мире существует множество систем управления базами данных. Несмотря на то что они могут по-разному работать с разными объектами и предоставляют пользователю различные функции и средства, большинство СУБД опираются на единый устоявшийся комплекс основных понятий. Это дает нам возможность рассмотреть одну систему и обобщить ее понятия, приемы и методы на весь класс СУБД. В качестве такого учебного объекта мы выберем СУБД Microsoft Access, входящую в пакет Microsoft Office наряду с пакетами Microsoft Word и Microsoft Excel. В тех случаях, когда конкретные приемы операций зависят от используемой версии программы, мы будем опираться на последнюю версию Microsoft Access 2000, хотя в основном речь будет идти о таких обобщенных понятиях и методах, для которых различия между конкретными версиями программ второстепенны.

* 1. **Структура простейшей базы данных**

Сразу поясним, что если в базе нет никаких данных (пустая база), то это все равно полноценная база данных. Этот факт имеет методическое значение. Хотя данных в базе и нет, но информация в ней все-таки есть — это *структура базы*. Она определяет методы занесения данных и хранения их в базе. Простейший "некомпьютерный" вариант базы данных — деловой ежедневник, в котором каждому календарному дню выделено по странице. Даже если в нем не записано ни строки, он не перестает быть ежедневником, поскольку имеет структуру, четко отличающую его от записных книжек, рабочих тетрадей и прочей писчебумажной продукции.

Базы данных могут содержать различные объекты, но, забегая вперед, скажем, что основными объектами любой базы данных являются ее таблицы. Простейшая база данных имеет хотя бы одну таблицу. Соответственно, структура простейшей базы данных тождественно равна структуре ее таблицы, рис. 11.1.

Мы знаем, что структуру двумерной таблицы образуют столбцы и строки. Их аналогами в структуре простейшей базы данных являются *поля* и *записи*. Если записей в таблице пока нет, значит, ее структура образована только набором полей. Изменив состав полей базовой таблицы (или их свойства), мы изменяем структуру базы данных и, соответственно, получаем новую базу данных.



Рис. 1.1. Простая таблица базы данных

* 1. **Свойства полей базы данных**

Поля базы данных не просто определяют структуру базы — они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Ниже перечислены основные свойства полей таблиц баз данных на примере СУБД Microsoft Access.

* **Имя поля** — определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц).
* **Тип поля** — определяет тип данных, которые могут содержаться в данном поле.
* **Размер поля** — определяет предельную длину (в символах) данных, которые могут размещаться в данном поле.
* **Формат поля** — определяет способ форматирования данных в ячейках, принадлежащих полю.
* **Маска ввода** — определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных).
* **Подпись** — определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство **Имя поля**).
* **Значение по умолчанию** — то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных).
* **Условие на значение** — ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты).
* **Сообщение об ошибке** — текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство Условие на значение).
* **Обязательное поле** — свойство, определяющее обязательность заполнения данного поля при наполнении базы;
* **Пустые строки** — свойство, разрешающее ввод пустых строковых данных (от свойства Обязательное поле отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым).
* **Индексированное поле** — если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

Здесь мы должны обратить особое внимание читателя на то, что поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных. Так, например, список вышеуказанные свойств полей относится в основном к полям текстового типа. Поля других типов могут иметь или не иметь эти свойства, но могут добавлять к ним и свои. Например для данных, представляющих действительные числа, важным свойством является количество знаков после десятичной запятой. С другой стороны, для полей, используемых для хранения рисунков, звукозаписей, видеоклипов и других объектов OLE, большинство вышеуказанных свойств не имеют смысла.

* 1. **Типы данных**
* Таблицы баз данных, как правило, допускают работу с большим количеством разных типов данных. Так, например, базы данных Microsoft Access работают со следующими типами данных.
* **Текстовый** — тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).
* **Мемо** — специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он хранится в другом месте базы данных, а в поле хранится указатель на него, но для пользователя такое разделение заметно не всегда.
* **Числовой** — тип данных для хранения действительных чисел.
* **Дата/время** — тип данных для хранения календарных дат и текущего времени.
* **Денежный** — тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.
* **Счетчик** — специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование — для порядковой нумерации записей
* **Логический** — тип для хранения логических данных (могут принимать только два значения, например Да или Нет).
* **Поле объекта OLE** — специальный тип данных, предназначенный для хранения объектов OLE, например мультимедийных. Реально, конечно, такие объекты в таблице не хранятся. Как и в случае полей МЕМО, они хранятся в другом месте внутренней структуры файла базы данных, а в таблице хранятся только указатели на них (иначе работа с таблицами была бы чрезвычайно замедленной).
* **Гиперссылка** — специальное поле для хранения адресов URL Web-объектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.
* **Мастер подстановок** — это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод в данных поле так, чтобы не вводить их вручную, а выбирать из раскрывающегося списка.
	1. **Безопасность баз данных**

Базы данных — это тоже файлы, но работа с ними отличается от работы с файлами других типов, создаваемых прочими приложениями. Выше мы видели, что всю работу по обслуживанию файловой структуры берет на себя операционная система. Для баз данных предъявляются особые требования с точки зрения безопасности, поэтому в них реализован другой подход к сохранению данных.

При работе с обычными приложениями для сохранения данных мы выдаем соответствующую команду, задаем имя файла и доверяемся операционной системе. Если мы закроем файл, не сохранив его, то вся работа по созданию или редактированию файла пропадет безвозвратно.

Базы данных — это особые структуры. Информация, которая в них содержится, очень часто имеет общественную ценность. Нередко с одной и той же базой (например, с базой регистрации автомобилей в ГИБДД) работают тысячи людей по всей стране. От информации, которая содержится в некоторых базах, может зависеть благополучие множества людей. Поэтому целостность содержимого базы не может и не должна зависеть ни от конкретных действий некоего пользователя, забывшего сохранить файл перед выключением компьютера, ни от перебоев в электросети.

Проблема безопасности баз данных решается тем, что в СУБД для сохранения информации используется двойной подход. В части операций, как обычно, участвует операционная система компьютера, но некоторые операции сохранения происходят в обход операционной системы.

Операции изменения структуры базы данных, создания новых таблиц или иных объектов происходят при сохранении файла базы данных. Об этих операциях СУБД предупреждает пользователя. Это, так сказать, глобальные операции. Их никогда не проводят с базой данных, находящейся в коммерческой эксплуатации, — только с ее копией. В этом случае любые сбои в работе вычислительных систем не страшны.

С другой стороны, операции по изменению содержания данных, не затрагивающие структуру базы, максимально автоматизированы и выполняются без предупреждения. Если работая с таблицей данных мы что-то в ней меняем в составе данных, то изменения сохраняются немедленно и автоматически.

Обычно, решив отказаться от изменений в документе, его просто закрывают без сохранения и вновь открывают предыдущую копию. Этот прием работает почти во всех приложениях, но только не в СУБД. Все изменения, вносимые в таблицы базы, сохраняются на диске без нашего ведома, поэтому попытка закрыть базу "без сохранения" ничего не даст, так как все уже сохранено. Таким образом, редактируя таблицы баз данных, создавая новые записи и удаляя старые, мы как бы работаем с жестким диском напрямую, минуя операционную систему.

**Предупреждение**. По указанным выше причинам нельзя заниматься учебными экспериментами на базах данных, находящихся в эксплуатации. Для этого следует создавать специальные учебные базы или выполнять копии структуры реальных баз (без фактического наполнения данными).

1. **Проектирование баз данных**
	1. **Режимы работы с базами данных**

Обычно с базами данных работают две категории исполнителей. Первая категория — *проектировщики*. Их задача состоит в разработке структуры таблиц базы данных и согласовании ее с заказчиком. Кроме таблиц проектировщики разрабатывают и другие объекты базы данных, предназначенные, с одной стороны, для автоматизации работы с базой, а с другой стороны — для ограничения функциональных возможностей работы с базой (если это необходимо из соображений безопасности). Проектировщики не наполняют базу конкретными данными (заказчик может считать их конфиденциальными и не предоставлять посторонним лицам). Исключение составляет экспериментальное наполнение модельными данными на этапе отладки объектов базы.

Вторая категория исполнителей, работающих с базами данных, — *пользователи*. Они получают исходную базу данных от проектировщиков и занимаются ее наполнением и обслуживанием. В общем случае пользователи не имеют средств доступа к управлению структурой базы — только к данным, да и то не ко всем, а к тем, работа с которыми предусмотрена на конкретном рабочем месте.

Соответственно, система управления базами данных имеет два режима работы: *проектировочный* и *пользовательский*. Первый режим предназначен для создания или изменения структуры базы и создания ее объектов. Во втором режиме происходит использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

* 1. **Объекты базы данных**

Мы уже упомянули о том, что кроме таблиц база данных может содержать и другие типы объектов. Привести полную классификацию возможных объектов баз данных затруднительно, поскольку каждая система управления базами данных может реализовать свои типы объектов. Однако основные типы объектов мы можем рассмотреть на примере СУБД Microsoft Access. В версии Microsoft Access 2000 эта СУБД позволяет создавать и использовать объекты семи различных типов.

**Таблицы**. Как мы уже говорили, это основные объекты любой базы данных. Во-первых, в таблицах хранятся все данные, имеющиеся в базе, а во-вторых, таблицы хранят и структуру базы (поля, их типы и свойства).

**Запросы**. Эти объекты служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции как отбор данных, их сортировку и фильтрацию. С помощью запросов можно выполнять преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое наполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое, рис. 11.2.

Начинающие пользователи не сразу понимают роль запросов, поскольку все те же операции можно делать и с таблицами. Да, действительно, это так, но есть соображения удобства (в первую очередь быстродействия) и соображения безопасности.

Из соображений безопасности, чем меньше доступа к базовым таблицам имеют конечные пользователи, тем лучше. Во-первых, снижается риск того, что неумелыми действиями они повредят данные в таблицах. Во-вторых, предоставив разным пользователям разные запросы, можно эффективно разграничить их доступ к данным в строгом соответствии с кругом персональных обязанностей. В банках, например, одни сотрудники имеют доступ к таблицам данных о клиентах, другие - к их расчетным счетам, третьи — к таблицам активов банка. Если и есть специальные службы, имеющие доступ ко всем информационным ресурсам банка (с целью контроля и анализа), то они лишены средств для внесения изменений — все сделано так, чтобы один человек не мог совершить фиктивную операцию, независимо от того, какую должность он занимает. В базе данных, имеющей правильно организованную структуру, для совершения противоправных действий необходим сговор нескольких участников, а такие действия пресекаются не программными, а традиционными средствами обеспечения безопасности.



Рис. 1.2. Структура простого запроса

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную *результирующую таблицу*. Если хотят подчеркнутьфакт “временности” этой таблицы, то ее еще называют *моментальным снимком*. Когда мы работаем с основными таблицами базы, мы физически имеем дело с жестким диском, то есть с очень медленным устройством (напомним, что это связано с особенностью сохранения данных, описанной выше). Когда же на основании запроса мы получаем результирующую таблицу, то имеем дело с электронной таблицей, не имеющей аналога на жестком диске, — это только *образ* отобранных полей и записей. Разумеется, работа с “образом” происходит гораздо быстрее и эффективнее - это еще одно основание для того, чтобы широко использовать запросы.

Ранее при обсуждении основных структур данных отмечалось, что недостатком упорядоченных табличных структур является сложность их обновления, поскольку при внесении новых записей нарушается упорядоченность — приходится переделывать всю таблицу. В системах управления базами данных и эта проблема решается благодаря запросам.

Основной принцип состоит в том, что от базовых таблиц никакой упорядоченности не требуется. Все записи в основные таблицы вносятся только в естественном порядке по мере их поступления, то есть в неупорядоченном виде. Если же пользователю надо видеть данные, отсортированные или отфильтрованные по тому или иному принципу, он просто использует соответствующий запрос. Если нужного запроса нет, он обращается к проектировщику и просит его такой запрос сделать и предоставить.

**Формы**. Если запросы — это специальные средства для отбора и анализа данных, то формы — это средства для ввода данных. Смысл их тот же — предоставить пользователю средства для заполнения только тех полей, которые ему заполнять положено. Одновременно с этим в формеможно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочие) для автоматизации ввода. Преимущества форм раскрываются особенно наглядно, когда происходит ввод данных с заполненных бланков. В этом случае форму делают графическими средствами так, чтобы она повторяла оформление бланка — это заметно упрощает работу наборщика, снижает его утомление и предотвращает появление печатных ошибок. На сопроводительном рисунке приведен пример простейшей формы для ввода данных.

С помощью форм данные можно не только вводить, но и отображать. Запросы тоже отображают данные, но делают это в виде результирующей таблицы, не имеющей почти никаких средств оформления. При выводе данных с помощью форм можно применять специальные средства оформления (рис. 11.3).



Рис. 1.3. Пример формы

**Отчеты**. По своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и дли вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т. п.) (рис. 11.4).

**Страницы**. Это специальные объекты баз данных, реализованные в последней версии СУБД Microsoft Access 2000. Правда, более корректно их называть *страницами доступа к данным*. Физически это особый объект, выполненный в коде HTML, размещаемый на Web-странице и передаваемый клиенту вместе с ней. Сам по себе этот объект не является базой данных, но содержит компоненты, через которые осуществляется связь переданной Web-страницы с базой данных, остающейся на сервере. Пользуясь этими компонентами, посетитель Web-узла может просматривать записи базы в полях страницы доступа. Таким образом, страницы доступа к данным осуществляют интерфейс между клиентом, сервером и базой данных, размещенной на сервере. Эта база данных не обязательно должна быть базой данных Microsoft Access. Страницы доступа, созданные средствами Microsoft Access, позволяют работать также с базами данных Microsoft SQL Server.

**Макросы и модули**. Эти категории объектов предназначены как для автоматизации повторяющихся операций при работе с системой управления базами данных, так и для создания новых функций путем программирования. В СУБД Microsoft Access макросы состоят из последовательности внутренних команд СУБД и являются одним из средств автоматизации работы с базой. Модули создаются средствами внешнего языка программирования, в данном случае языка Visual Basic for Applications. Это одно из средств, с помощью которых разработчик базы может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, а также уровень ее защищенности.



Рис. 1.4. Пример простого отчета

* 1. **Проектирование базы данных**

Мы лишь рассмотрели основные понятия баз данных и еще не знакомы ни с одной системой управления базами данных, но, тем не менее, поднимаем вопрос о проектировании базы. Это не случайно. Методически правильно начинать работу с карандашом и листом бумаги в руках, не используя компьютер. На данном этапе он просто не нужен. Неоптимальные решения и прямые ошибки, заложенные на этапе проектирования, впоследствии очень трудно устраняются, поэтому этот этап является основополагающим.

**Разработка технического задания**. Техническое задание на проектирование базы данных должен предоставить заказчик. Однако для этого он должен владеть соответствующей терминологией и знать, хотя бы в общих чертах, технические возможности основных систем управления базами данных. К сожалению, на практике такое положение встречается не всегда. Поэтому обычно используют следующие подходы:

* демонстрируют заказчику работу аналогичной базы данных, после чего согласовывают спецификацию отличий;
* если аналога нет, выясняют круг задач и потребностей заказчика, после чего помогают ему подготовить техническое задание.

При подготовке технического задания составляют:

* список исходных данных, с которыми работает заказчик;
* список выходных данных, которые необходимы заказчику для управления структурой своего предприятия;
* список выходных данных, которые не являются необходимыми для заказчика, но которые он должен предоставлять в другие организации (в вышестоящие структуры, в органы статистического учета, прочие административные и контролирующие организации).

При этом очень важно не ограничиваться взаимодействием с головным подразделением заказчика, а провести обсуждение со всеми службами и подразделениями, которые могут оказаться поставщиками данных в базу или их потребителями. Так, например, при подготовке базы данных для учета абитуриентов и студентов в высшем учебном заведении, необходимо не только изучить документооборот ректората и всех деканатов, но и понять, что хотели бы получить от базы данных службы. Следует изучить работу подразделений, распределяющих учебную нагрузку преподавателей, отвечающих за распределение аудиторного фонда, за проживание студентов в общежитии и других. В расчет должны приниматься и такие службы, как библиотека, отдел кадров и прочие. В любой момент может выясниться, например, что администрация библиотеки должна периодически поставлять кому-то отчеты, характеризующие читательскую активность студентов в зависимости от пола, возраста и социального положения. К возможным пожеланиям заказчика следует готовиться на этапе проектирования, до создания базы.

**Разработка структуры базы данных**. Выяснив основную часть данных, которые заказчик потребляет или поставляет, можно приступать к созданию структуры базой, то есть структуры ее основных таблиц.

1. Работа начинается с составления генерального списка полей — он может насчитывать десятки и даже сотни позиций.

2. В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.

3. Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель — обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше — на одном рабочем месте.

Наметив столько таблиц, сколько подразделений охватывает база данных, приступают к дальнейшему делению таблиц. Критерием необходимости деления является факт множественного повтора данных в соседних записях. Это явное свидетельство того, что таблицу надо поделить на две взаимосвязанные таблицы.

4. В каждой из таблиц намечают *ключевое поле*. В качестве такового выбирают поле, данные в котором повторяться не могут. Например, для таблицы данных о студентах таким полем может служить индивидуальный шифр студента. Для таблицы, в которой содержатся расписания занятий, такого поля можно и не найти, но его можно создать искусственным комбинированием полей “Время занятия” и “Номер аудитории”. Эта комбинация неповторима, так как в одной аудитории в одно и то же время не принято проводить два различных занятия.

Если в таблице вообще нет никаких полей, которые можно было бы использовать как ключевые, всегда можно ввести дополнительное поле типа **Счетчик** — оно не может содержать повторяющихся данных по определению.

5. С помощью карандаша и бумаги расчерчивают связи между таблицами. На рис. 11.5 показан пример взаимосвязи между группой таблиц, составляющих одну базу данных. Такой чертеж называется *схемой данных*.

Существует несколько типов возможных связей между таблицами. Наиболее распространенными являются связи “один ко многим” и “один к одному”. Связь между таблицами организуется на основе общего поля, причем в одной из таблиц оно обязательно должно быть ключевым, то есть на стороне “один” должно выступать ключевое поле, содержащее уникальные, неповторяющиеся значения. Значения на стороне “многие” могут повторяться.



Рис. 2.5. Схема связей между таблицами

Рассмотрим таблицу **Клиенты** (рис. 11.5). Здесь поле **Код клиента** является ключевым. Это понятно, поскольку у каждого клиента должен быть свой уникальный код, идентифицирующий его однозначно. Если мы рассмотрим таблицу **Заказы**, то увидим, что в ней код клиента не может быть уникальным, поскольку каждый клиент мог сделать сколь угодно много заказов. На схеме данных эти поля соединены линией связи. С одной стороны эта линия маркирована знаком “1”, с другой стороны — значком “бесконечность”. Это графический метод изображения связи “один ко многим”.

Ключевым полем в таблице заказов является **Код заказа** — он однозначно идентифицирует, кто, когда, что заказал и на какую сумму. Здесь же можно узнать, какой сотрудник принял заказ к исполнению. Поскольку один сотрудник может принять множество заказов, поле **Код сотрудника** в таблице заказов не является ни уникальным, ни ключевым, зато в таблице **Сотрудники** это поле уникально.

Про подобные таблицы говорят, что они связаны *реляционными отношениями*, Соответственно, системы управления, способные работать со связанными таблицами, называют *системами управления реляционными базами данных*, а схему данных в технической литературе могут называть *схемой реляционных отношений*.

6. Разработкой схемы данных заканчивается “бумажный” этап работы над техническим предложением. Эту схему можно согласовать с заказчиком, после чего приступать к непосредственному созданию базы данных.

Следует помнить, что по ходу разработки проекта заказчику непременно будут приходить в голову новые идеи. На всех этапах проектирования он стремится охватить единой системой все новые и новые подразделения и службы предприятия. Возможность гибкого исполнения его пожеланий во многом определяется квалификацией разработчика базы данных. Если схема данных составлена правильно, подключать к базе новые таблицы нетрудно. Если структура базы нерациональна, разработчик может испытать серьезные трудности и войти в противоречия с заказчиком.

Противоречия исполнителя с заказчиком всегда свидетельствуют о недостаточной квалификации исполнителя. Именно поэтому этап предварительного проектирования базы данных следует считать основным. От его успеха зависит, насколько база данных станет удобной и будут ли с ней работать пользователи. Если отмечается, что пользователи базы “саботируют” ее эксплуатацию и предпочитают работать традиционными методами, это говорит не о низкой квалификации пользователей, а о недостаточной квалификации разработчика базы.

На этом этапе завершается предварительное проектирование базы данных, и на следующем этапе начинается ее непосредственная разработка. С этого момента следует начать работу с системой управления базами данных. В нашем примере мы рассмотрим СУБД Microsoft Access 2000.

1. **Работа с СУБД Microsoft Access 2000**
	1. **Общие замечания**

Ниже мы рассмотрим, как в программе Microsoft Access 2000 реализованы средства разработки основных объектов базы данных, и в упражнениях познакомимся с конкретными приемами работы. Однако прежде чем приступать к освоению системы, следует учесть ряд важных замечаний, связанных с особенностями ее автоматизации.

СУБД Microsoft Access 2000 предоставляет несколько средств создания каждого из основных объектов базы. Эти средства можно классифицировать как:

* ручные (разработка объектов в режиме **Конструктора**);
* автоматизированные (разработка с помощью программ-мастеров);
* автоматические — средства ускоренной разработки простейших объектов.

Соотношения между этими средствами понятны: ручные средства являются наиболее трудоемкими, но обеспечивают максимальную гибкость; автоматизированные и автоматические средства являются наиболее производительными, но и наименее гибкими. Методической особенностью изучения программы Microsoft Access является тот факт, что в учебных целях для создания разных объектов целесообразно пользоваться разными средствами.

1. При разработке учебных таблиц и запросов рекомендуется использовать ручные средства — работать в режиме **Конструктора**. Использование мастеров ускоряет работу, но не способствует освоению понятий и методов.

2. При разработке учебных форм, отчетов и страниц доступа наоборот лучше пользоваться автоматизированными средствами, предоставляемыми мастерами. Это связано с тем, что для данных объектов большую роль играет внешний вид. Дизайн этих объектов весьма трудоемок, поэтому его лучше поручить программе, а учащемуся сосредоточиться на содержательной части работы.

3. Разработку макросов и модулей в данном пособии мы не рассматриваем. Эти средства ориентированы на профессиональных разработчиков баз данных, поэтому в рамках общетехнического курса “Информатики” для них недостаточно места.

**3.2. Работа с таблицами**

**Создание таблиц**. Работа с любыми объектами начинается с окна **База данных** (рис. 11.6). На левой панели данного окна сосредоточены элементы управления для вызова всех семи типов объектов программы. Создание таблиц начинается с выбора элемента управления **Таблицы**.



Рис. 3. 6. Окно База данных

На правой панели представлен список таблиц, уже имеющихся в составе базы, и приведены элементы управления для создания новой таблицы. Чтобы создать таблицу вручную, следует использовать значок **Создание таблицы в режиме конструктора**.

Окно **Конструктора таблиц** представлено на рис. 11.7. То, что мы видим в этом режиме, фактически является графическим бланком для создания и редактирования структуры таблиц. В первом столбце вводят имена полей. Если свойство **Подпись** для поля не задано, то **Имя поля** станет одновременно и именем столбца будущей таблицы. Тип для каждого поля выбирают из раскрывающегося списка, открываемого кнопкой выбора типа данных (см. рис. 11.7). Эта кнопка — скрытый элемент управления. Она отображается только после щелчка на поле бланка. Это надо иметь в виду — в Microsoft Access очень много таких скрытых элементов управления, которые не отображаются, пока ввод данных не начат.

**Совет**. При изучении приемов работы с программой Microsoft Access целесообразно специально “прощелкивать” пустые поля ее бланков левой кнопкой мыши в поисках “скрытых” элементов управления.

Нижняя часть бланка содержит список свойств поля, выделенного в верхней части. Некоторые из свойств уже заданы по умолчанию. Свойства полей не являются обязательными. Их можно настраивать по желанию, а можно и не трогать.

При создании таблицы целесообразно (хотя и не обязательно) задать ключевое поле. Это поможет впоследствии, при организации связей между таблицами. Для задания ключевого поля достаточно щелкнуть на его имени правой кнопкой мыши и в открывшемся контекстном меню выбрать пункт **Ключевое поле**.

Выше мы говорили о том, что если первичный ключ необходим для связи с другими таблицами, но ни одно из полей не является уникальным, то первичный ключ можно создать на базе двух (или более полей). Эта операция выполняется точно так же, через контекстное меню, надо только уметь выделить сразу несколько полей. Групповое выделение выполняют при нажатой клавише Shift щелчками на квадратных маркерах слева от имен полей.

Закончив создание структуры таблицы, бланк закрывают (при этом система выдает запрос на сохранение таблицы), после чего дают таблице имя, и с этого момента она доступна в числе прочих таблиц в основном окне **База данных**. Оттуда ее и можно открыть в случае необходимости.



Рис. 3. 7. Окно таблицы в режиме Конструктора

Созданную таблицу открывают в окне **База данных** двойным щелчком на ее значке. Новая таблица не имеет записей — только названия столбцов, характеризующие структуру таблицы (рис. 11.8). Заполнение таблицы данными производится обычным порядком. Курсор ввода устанавливается в нужную ячейку указателем мыши. Переход к следующей ячейке можно выполнить клавишей Tab. Переход к очередной записи выполняется после заполнения последней ячейки.

В нижней части таблицы расположена Панель кнопок перехода. Ее элементами управления удобно пользоваться при навигации по таблице, имеющей большое число записей.

Начинающим пользователям Microsoft Access доставляет неудобство тот факт, что данные не всегда умещаются в ячейках таблицы. Шириной столбцов можно управлять методом перетаскивания их границ. Удобно использовать автоматическое форматирование столбцов “по содержимому”. Для этого надо установить указатель мыши на границу между столбцами (в строке заголовков столбцов), дождаться, когда указатель сменит форму, и выполнить двойной щелчок. Это общесистемный прием Windows 98, и им можно пользоваться в данной программе, как и во многих других.



Рис. 3. 8. Пример новой таблицы

После наполнения таблицы данными сохранять их не надо — все сохраняется автоматически. Однако если при работе с таблицей произошло редактирование ее макета (например, изменялась ширина столбцов), СУБД попросит подтвердить сохранение этих изменений.

Если возникнет необходимость изменить структуру таблицы (состав полей или их свойства), таблицу надо открыть в режиме Конструктора. Для этого ее следует выделить в окне **База данных** и щелкнуть на кнопке **Конструктор**

Если на этапе проектирования базы данных была четко разработана структура таблиц, то создание таблиц с помощью Конструктора происходит очень быстро и эффективно. Даже без использования автоматизированных средств создание основы для достаточно крупных проектов происходит в считанные минуты — это ценное свойство СУБД Microsoft Access, но оно реализуется при непременном условии тщательной предварительной подготовки.

**Создание межтабличных связей**. Если структура базы данных продумана заранее, а связи между таблицами намечены, то создание реляционных отношений между таблицами выполняется очень просто. Вся необходимая работа происходит в специальном окне **Схема данных** и выполняется с помощью мыши. Окно **Схема данных** открывают кнопкой на панели инструментов или командой **Сервис -> Схема данных** (если в меню **Сервис** не видно соответствующего пункта, следует раскрыть расширенную часть меню). Порядок создания межтабличных связей рассмотрен подробно в упражнении 11.2.

Образовавшаяся межтабличная связь отображается в окне **Схема данных** в виде линии, соединяющей два поля разных таблиц. При этом одна из таблиц считается главной, а другая — связанной. Главная — это та таблица, которая участвует в связи своим ключевым полем (название этого поля на схеме данных отображается полужирным шрифтом).

Здесь мы подходим к важному вопросу: “А зачем вообще нужна связь между таблицами?” У связи два основных назначения. Первое — обеспечение целостности данных, а второе — автоматизация задач обслуживания базы. Представим себе, что в таблице Клиенты, где каждый клиент уникален, кто-то удалит запись для одного из клиентов, но не сделает этого в таблице Заказы. Получится, что согласно таблице Заказы некто, не имеющий ни имени, ни адреса, а только абстрактный код, делал заказы. Узнать по коду, кто же это был на самом деле, будет невозможно — произошло нарушение целостности данных.

В данном случае владелец базы может применить три подхода: либо вообще ничего не делать для защиты целостности данных, либо запретить удаление данных из ключевых полей главных таблиц, либо разрешить его, но при этом адекватно обработать и связанные таблицы. Вручную сделать это чрезвычайно трудно, поэтому и нужны средства автоматизации.

Связь между таблицами позволяет:

* либо исключить возможность удаления или изменения данных в ключевом поле главной таблицы, если с этим полем связаны какие-либо поля других таблиц;
* либо сделать так, что при удалении (или изменении) данных в ключевом поле главной таблицы автоматически (и абсолютно корректно) произойдет удаление или изменение соответствующих данных в полях связанных таблиц.

Для настройки свойств связи надо в окне **Схема данных** выделить линию, соединяющую поля двух таблиц, щелкнуть на ней правой кнопкой мыши и открыть контекстное меню связи, после чего выбрать в нем пункт **Изменить связь** — откроется диалоговое окно **Изменение связи,**рис.11.9. В нем показаны названия связанных таблиц и имена полей, участвующих в связи (здесь же их можно изменить), а также приведены элементы управления для обеспечения условий целостности данных.

Если установлен только флажок **Обеспечение целостности данных**, то удалять данные из ключевого поля главной таблицы нельзя. Если вместе с ним включены флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей**, то, соответственно, операции редактирования и удаления данных в ключевом поле главной таблицы разрешены, но сопровождаются автоматическими изменениями в связанной таблице.

Таким образом, смысл создания реляционных связей между таблицами состоит, с одной стороны, в защите данных, а с другой стороны — в автоматизации внесения изменений сразу в несколько таблиц при изменениях в одной таблице.



Рис. 3.9. Окно настройки межтабличной связи

* 1. **Работа с запросами**

Если структура базы данных предприятия хорошо продумана, то исполнители, работающие с базой, должны навсегда забыть о том, что в базе есть таблицы, а еще лучше, если они об этом вообще ничего не знают. Таблицы — слишком ценные объекты базы, чтобы с ними имел дело кто-либо, кроме разработчика базы.

Если исполнителю надо получить данные из базы, он должен использовать специальные объекты — запросы. Все необходимые запросы разработчик базы должен подготовить заранее. Если запрос подготовлен, надо открыть панель Запросы в окне **База данных**, выбрать его и открыть двойным щелчком на значке — откроется результирующая таблица, в которой исполнитель найдет то, что его интересует.

В общем случае результирующая таблица может не соответствовать ни одной из базовых таблиц базы данных. Ее поля могут представлять набор из полей разных таблиц, а ее записи могут содержать отфильтрованные и отсортированные записи таблиц, на основе которых формировался запрос. Лишь в тех случаях, когда исполнитель не находит нужных данных в результирующей таблице, возникает необходимость готовить новый запрос — это задача разработчика базы.

В учебных целях запросы лучше готовить вручную, с помощью **Конструктора**. Как и в случае с таблицами, для этого есть специальный значок в окне **База данных**. Он называется **Создание запроса в режиме конструктора** и открывает специальный бланк, называемый бланком запроса по образцу. За этим длинным названием скрывается тот приятный факт, что, хотя запросы к таблицам баз данных пишутся на специальном языке программирования — SQL, пользователям Microsoft Access изучать его не обязательно, а большинство операций можно выполнить щелчками кнопок мыши и приемом перетаскивания в бланке.

Бланк запроса по образцу представлен на рис. 11.10. Как видно, он состоит из двух областей. В верхней отображается структура таблиц, к которым запрос адресован, а нижняя область разбита на столбцы — по одному столбцу на каждое поле будущей результирующей таблицы.



Рис. 3.10. Бланк запроса по образцу

Идея формирования запроса по образцу чрезвычайно проста. С помощью контекстного меню на верхней половине бланка открывают те таблицы, к которым обращен запрос. Затем в них щелкают двойными щелчками на названиях тех полей, которые, должны войти в результирующую таблицу. При этом автоматически заполняются столбцы в нижней части бланка. Сформировав структуру запроса, его закрывают, дают ему имя и в дальнейшем запускают двойным щелчком на значке в окне **База данных**.

Порядок действий, рассмотренный выше, позволяет создать простейший запрос, называемый запросом на выборку. Он позволяет выбрать данные из полей таблиц, на основе которых запрос сформирован.

**Упорядочение записей в результирующей таблице**. Если необходимо, чтобы данные, отобранные в результате работы запроса на выборку, были упорядочены по какому-либо полю, применяют сортировку. В нижней части бланка имеется специальная строка **Сортировка**. При щелчке на этой строке открывается кнопка раскрывающегося списка, в котором можно выбрать метод сортировки: по возрастанию или по убыванию. В результирующей таблице данные будут отсортированы по тому полю, для которого задан порядок сортировки.

Возможна
многоуровневая сортировка — сразу по нескольким полям. В этом случае данные сначала сортируются по тому полю, которое в бланке запроса по образцу находится левее, затем по следующему полю, для которого включена сортировка, и так далее слева направо. Соответственно, при формировании запроса надо располагать поля результирующей таблицы не как попало, а с учетом будущей сортировки. В крайнем случае, если запрос уже сформирован и надо изменить порядок следования столбцов, пользуются следующим приемом:

* выделяют столбец щелчком на его заголовке (кнопку мыши отпускают);
* еще раз щелкают на заголовке уже выделенного столбца (но кнопку не отпускают);
* перетаскивают столбец в другое место.

**Управление отображением данных в результирующей таблице**. В нижней части бланка запроса по образцу имеется строка **Вывод на экран**. По умолчанию предполагается, что все поля, включенные в запрос, должны выводиться на экран, но это не всегда целесообразно. Например, бывают случаи, когда некое поле необходимо включить в запрос, например потому, что оно является полем сортировки, но, в то же время, нежелательно, чтобы пользователь базы видел его содержание. В таких случаях отображение содержимого на экране подавляют сбросом флажка **Вывод на экран**. Примером может быть запрос на вывод списка сотрудников предприятия, отсортированный по количеству дней, пропущенных по болезни. Он позволит каждому оценить свое положение в этом списке, но не позволит точно узнать, кто и сколько дней болел.

**Использование условия отбора**. Дополнительным средством, обеспечивающим отбор данных по заданному критерию, является так называемое **Условие отбора**. Соответствующая строка имеется в нижней части бланка запроса по образцу. Для каждого поля в этой строке можно задать индивидуальное условие. В качестве примеров можно привести следующие условия:

>100 – отбираются записи, у которых в поле КодЗаказа значения превышают 100.

Like “Па\*” - отбираются записи, у которых в поле Название значения начинаются с букв “Па”.

**Другие виды запросов**. Мы рассмотрели запроси на выборку. Это самые простые и, в то же время, наиболее распространенные виды запросов. Однако существуют и другие виды запросов, некоторые их которых выполняются на базе предварительно созданного запроса на выборку. К ним относятся прежде всего:

* **запросы с параметром** (интересны тем, что критерий отбора может задать сам пользователь, введя нужный параметр при вызове запроса);
* **итоговые запросы**, назначение которых отдаленно напоминает итоговые функций электронных таблиц (производят математические вычисления по заданном полю и выдают результат;
* **запросы на изменение** — позволяют автоматизировать заполнение полей таблиц;
* **перекрестные запросы**, позволяющие создавать результирующие таблицы на основе результатов расчетов, полученных при анализе группы таблиц;
* **специфические запросы SQL** — запросы к серверу базы данных, написанные на языке запросов SQL.

С некоторыми видами запросов мы познакомимся в упражнениях.

* 1. **Работа с формами**

С одной стороны, формы позволяют пользователям вводить данные в таблицы базы данных без непосредственного доступа к самим таблицам. С другой стороны, они позволяют выводить результаты работы запросов не в виде скупых результирующих таблиц, а в виде красиво оформленных форм. В связи с таким разделением существует два вида формирования структуры форм: на основе таблицы и на основе запроса, хотя возможен и комбинированный подход, — это вопрос творчества.

**Автоформы**. В отличие от таблиц и запросов, которые мы формировали вручную, формы удобнее готовить с помощью средств автоматизации. Полностью автоматическими являются средства, называемые *автоформами*. Существует три вида автоформ: “в столбец” (рис. 11.11), ленточные (рис. 11.12) и табличные (рис. 11.13).



Рис. 3.11. Автоформа "в столбец"



Рис. 3.12. Автоформа ленточная



Рис. 3.13. Автоформа табличная

Как видно из рисунков, автоформа “в столбец” отображает все поля одной записи — она удобна для ввода и редактирования данных. Ленточная автоформа отображает одновременно группу записей — ее удобно использовать для оформления вывода данных. Табличная автоформа по внешнему виду ничем не отличается от таблицы, на которой она основана.

Для создания автоформы следует открыть панель формы в окне **База данных** и воспользоваться командной кнопкой **Создать**. В открывшемся диалоговом окне **Новая форма** выбирают тип автоформы и таблицу (или запрос), на которой она основывается. После щелчка на кнопке ОК автоформа формируется автоматически и немедленно готова к работе, то есть к вводу или отображению данных.

Обратите внимание на то, что автоформа основывается только на одном объекте. Иные средства создания форм позволяют закладывать в основу структуры формы поля нескольких таблиц или запросов. Если форма основывается только на одном объекте, она называется простой формой. Если форма основывается на полях из нескольких связанных таблиц, то она называется сложной и представляет собой композицию из нескольких форм.

**Создание форм с помощью мастера**. **Мастер форм** — специальное программное средство, создающее структуру формы в режиме диалога с разработчиком. Мастер форм можно запустить из окна **База данных** щелчком на значке **Создание формы** с помощью мастера на панели **Формы**.

1. На первом этапе работы Мастера форм выбирают таблицы и поля, которые войдут в будущую форму.

2. На втором этапе выбирается внешний вид формы.

3. На третьем этапе выбирается стиль оформления формы.

4. На последнем этапе выполняется сохранение формы под заданным именем. Здесь же можно включить переключатель **Изменить макет формы**, который открывает только что созданную форму в режиме **Конструктора**. Этим удобно воспользоваться в учебных целях, чтобы рассмотреть структуру формы на готовом примере.

**Структура формы**. Как видно из рисунка 11.14, форма имеет три основных раздела: область заголовка, область данных и область примечания. Линии, разделяющие разделы, перетаскиваются по вертикали с помощью мыши — это позволяет изменять размеры разделов так, как требуется.



Рис. 3.14. Структура формы, созданной Мастером форм

Разделы заголовка и примечания имеют чисто оформительское назначение — их содержимое напрямую не связано с таблицей или запросом, на котором основана форма. Раздел данных имеет содержательное значение — в нем представлены элементы управления, с помощью которых выполняется отображение данных или их ввод. Разработчик формы может разместить здесь дополнительные элементы управления для автоматизации ввода данных (переключатели, флажки, списки и другие, типичные для приложений Windows).

**Элементы управления формы**. Элементы управления, которыми может пользоваться разработчик, представлены на **Панели элементов**, рис.11.15. Ее открывают щелчком на соответствующей кнопке панели инструментов Microsoft Access или командой **Вид -> Панель элементов**.



Рис. 3.15. Панель элементов

Выбор элемента управления выполняется одним щелчком на его значке в **Панели элементов**, после чего следующим щелчком в поле формы отмечается место, куда он должен быть поставлен. Вместе с элементом в поле формы вставляется его присоединенная надпись. По умолчанию эта надпись стандартная, например для переключателей это Переключатель1, Переключатель2 и т. д. Редактированием свойства элемента управления (доступ к свойствам открывается через контекстное меню) можно дать элементу управления более содержательную подпись.

Основными элементами оформления формы являются текстовые надписи и рисунки. Для создания в форме текстовых надписей служат два элемента управления — **Надпись** и **Поле**. В качестве надписи можно задать произвольный текст. Элемент **Поле** отличается тем, что в нем отображается содержимое одного из полей таблицы, на которой основана форма, то есть при переходе от записи к записи текст может меняться.

Для создания графических элементов оформления служат элементы управления **Рисунок**, **Свободная рамка объекта** и **Присоединенная рамка объекта**. Рисунок выбирается из графического файла и вставляется в форму. Элемент **Свободная рамка объекта** отличается тем, что это не обязательно рисунок — это может быть любой другой объект OLE, например мультимедийный. Элемент **Присоединенная рамка объекта** тоже в какой-то степени может служить для оформления формы, но его содержимое берется не из назначенного файла, а непосредственно из таблицы базы данных (если она имеет поле объекта OLE). Естественно, что при переходе между записями содержимое этого элемента будет меняться.

**Дизайн формы**. В то время как таблицы базы данных глубоко скрыты от посторонних глаз, формы базы данных — это средства, с помощью которых с ней общаются люди. Поэтому к формам предъявляются повышенные требования по дизайну.

В первую очередь, все элементы управления форм должны быть аккуратно выровнены. Это обеспечивается командой **Формат -> Выровнять**. Если нужно равномерно распределить элементы управления по полю формы, используют средства меню **Формат** ->**Интервал по горизонтали** или **Формат -> Интервал по вертикали**.

Ручное изменение размеров и положения элементов управления тоже возможно, но редко приводит к качественным результатам. При работе вручную используют перетаскивание маркеров, которые видны вокруг элемента управления в тот момент, когда он выделен. Особый статус имеет маркер левого верхнего угла. Обычно элементы управления перетаскиваются вместе с присоединенными к ним надписями. Перетаскивание с помощью этого маркера позволяет оторвать присоединенную надпись от элемента.

Существенную помощь при разработке дизайна формы оказывает вспомогательная сетка. Ее отображение включают командой **Вид -> Сетка**. Автоматическую привязку элементов к узлам сетки включают командой **Формат -> Привязать к сетке**.

**Управление последовательностью перехода**. Пользователь, для которого, собственно, и разрабатывается форма, ожидает, что ввод данных в нее должен происходить по элементам управления слева направо и сверху вниз. Однако при проектировании сложных форм, когда в процессе дизайна элементы управления многократно' перемещаются с места на место, очень легко перепутать их последовательность и создать неудобный порядок ввода данных.

Физически последовательность перехода — это порядок перехода к следующему полю по окончании работы с предыдущим. Она легко проверяется с помощью клавиши Tab. Если при последовательных нажатиях этой клавиши фокус ввода “мечется” по всей форме, значит последовательность перехода нерациональна и ее надо править.

Для управления последовательностью перехода служит диалоговое окно **Последовательность перехода**. В нем представлен список элементов управления формы. Порядок элементов в списке соответствует текущему порядку перехода. Изменение порядка перехода выполняется перетаскиванием в два приема:

* щелчком на кнопке маркера слева от названия выделяется элемент управления (кнопка мыши отпускается);
* после повторного щелчка с перетаскиванием элемент перемещается на новое место.

Закончив разработку макета формы, ее следует закрыть и сохранить под заданным именем. После открытия формы в окне **База данных**, с ней можно работать: просматривать или редактировать данные из базовой таблицы. Проверку последовательности перехода выполняют клавишей Tab.

* 1. **Работа с отчетами**

Отчеты во многом похожи на формы и страницы доступа к данным, но имеют иное функциональное назначение — они служат для форматированного вывода данных на печатающие устройства и, соответственно, при этом должны учитывать параметры принтера и параметры используемой бумаги.

Большая часть того, что было сказано о формах, относится и к отчетам. Здесь также существуют средства автоматического, автоматизированного и ручного проектирования. Средства автоматического проектирования реализованы автоотчетами (**Базаданных -> Создать -> Новый отчет -> Автоотчет в столбец**). Кроме автоотчетов “в столбец” существуют “ленточные” автоотчеты. Разницу между ними нетрудно увидеть, поставив эксперимент.

Средством автоматизированного создания отчетов является Мастер отчетов. Он запускается двойным щелчком на значке Создание отчета с помощью мастера в окне База данных. Мастер отчетов работает в шесть этапов. При его работе выполняется выбор базовых таблиц или запросов, на которых отчет базируется, выбор полей, отображаемых в отчете, выбор полей группировки, выбор полей и методов сортировки, выбор формы печатного макета и стиля оформления.

Структура готового отчета отличается от структуры формы только увеличенным количеством разделов. Кроме разделов заголовка, примечания и данных, отчет может содержать разделы верхнего и нижнего колонтитулов. Если отчет занимает более одной страницы, эти разделы необходимы для печати служебной информации, например номеров страниц. Чем больше страниц занимает отчет, тем важнее роль данных, выводимых на печать через эти разделы. Если для каких-то полей отчета применена группировка, количество разделов отчета увеличивается, поскольку оформление заголовков групп выполняется в отдельных разделах.

Редактирование структуры отчета выполняют в режиме Конструктора (режим запускается кнопкой **Конструктор** в окне **База данных**). Приемы редактирования те же, что и для форм. Элементы управления в данном случае выполняют функции элементов оформления, поскольку печатный отчет не интерактивный объект, в отличие от электронных форм и Web-страниц. Размещение элементов управления выполняют с помощью Панели элементов (**Вид -> Панель элементов**), которая по составу практически не отличается от Панели элементов формы. Важной особенностью отчетов является наличие средства для вставки в область верхнего или нижнего колонтитула текущего номера страницы и полного количества страниц. Эту операцию выполняют с помощью диалогового окна **Номера страниц** (**Вставка -> Номера страниц**).

**ПРАКТИЧЕСКОЕ ЗАДАНИЕ**

**Упражнение 1. Создание базовых таблиц**.

Руководитель малого предприятия, выполняющего сборку персональных компьютеров из готовых компонентов, заказал разработку базы данных, основанной на двух таблицах. Одна таблица содержит данные, которые могут отображаться для клиентов при согласовании спецификации изделия, — в ней указаны розничные цены на компоненты. Вторая таблица предназначена для анализа результатов деятельности предприятия — в ней содержатся оптовые цены на компоненты и краткая информация о поставщиках (клиенты предприятия не имеют доступа к данным этой таблицы).

1. Запустите программу Microsoft Access 2000 (**Пуск -> Программы -> Microsoft** **Access**).

2. В окне Microsoft Access включите переключатель **Новая база данных** и щелкните на кнопке ОК.

3. В окне **Файл** новой базы данных выберите папку \Мои документы и дайте файлу имя: **Комплектующие**. Убедитесь, что в качестве типа файла выбрано **Базы данных Microsoft Access**, и щелкните на кнопке **Создать**. Откроется окно новой базы — **Комплектующие: база данных**.

4. Откройте панель **Таблицы**.

5. Дважды щелкните на значке **Создание таблицы в режиме конструктора** — откроется бланк создания структуры таблицы.

6. Для первой таблицы введите следующие поля:

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Компонент | Текстовый |
| Модель | Текстовый |
| Основной параметр | Числовой |
| Цена | Числовой |

Обратите внимание на то, что в данном случае поле **Цена** задано не денежным типом, а числовым. Данные, относящиеся к денежному типу, имеют размерность, выраженную в рублях (если работа выполняется с версией Microsoft Access, локализованной в России). Но стоимость компонентов вычислительной техники выражать в этой единице измерения не принято. Для сравнимости цен разных поставщиков обычно используют “условные единицы”. В таких случаях удобно использовать поле числового типа, чтобы не перенастраивать всю СУБД.

7. Щелкните на поле **Цена**. В нижней части бланка задайте свойство **Число десятичных знаков**, равным 2.

8. Для связи с будущей таблицей поставщиков надо задать ключевое поле. Поскольку здесь ни одно поле явно не претендует на “уникальность”, используем комбинацию полей **Компонент** и **Модель**. Выделите оба поля в верхней части бланка (при нажатой клавише Shift). Щелчком правой кнопки мыши откройте контекстное меню и выберите в нем пункт **Ключевое поле**.

9. Закройте окно Конструктора. При закрытии окна дайте таблице имя **Комплектующие**.



Рис.16. Таблица Комплектующие в режиме Конструктора

10. Повторив действия пунктов 5-9, создайте таблицу **Поставщики**, в которую входят следующие поля.

Обратите внимание на то, что поле номера телефона является текстовым, несмотря на то, что обычно номера телефонов записывают цифрами. Это связано с тем, что они не имеют числового содержания. Номера телефонов не сравнивают по величине, не вычитают из одного номера другой и т. д. Это типичное текстовое поле.

Ключевое поле можно не задавать — для текущей задачи оно не требуется.

11. В окне **Комплектующие: база данных** откройте по очереди созданные таблицы и наполните их экспериментальным содержанием (3-4 записи). Закончив работу, закройте таблицы и завершите работу с программой.

**Упражнение 2. Создание межтабличных связей**.

1. Запустите программу Microsoft Access 2000 (**Пуск -> Программы -> Microsoft** **Access**).

2. В окне Microsoft Access включите переключатель **Открыть базу данных**, выберите ранее созданную базу **Комплектующие** и щелкните на кнопке ОК.

3. В окне **Комплектующие: база данных** откройте панель **Таблицы**. Убедитесь, что на ней присутствуют значки ранее созданных таблиц **Комплектующие** и **Поставщики**.

4. Разыщите на панели инструментов кнопку **Схема данных**. Если есть сложности, найдите команду строки меню: **Сервис -> Схема данных**. Воспользуйтесь любым из этих средств, чтобы открыть окно **Схема данных**. Одновременно с открытием этого окна открывается диалоговое окно **Добавление таблицы**, на вкладке **Таблицы** которого можно выбрать таблицы, между которыми создаются связи.

5. Щелчком на кнопке **Добавить** выберите таблицы **Комплектующие** и **Поставщики** — в окне **Схема данных** откроются списки полей этих таблиц.

6. При нажатой клавише Shift выделите в таблице **Комплектующие** два поля — **Компонент** и **Модель**.

7. Перетащите эти поля на список полей таблицы **Поставщики**. При отпускании кнопки мыши автоматически откроется диалоговое окно **Изменение связей**.

8. На правой панели окна **Изменение связей** выберите поля **Компонент** и **Модель** таблицы **Поставщики**, включаемые в связь. Не устанавливайте флажок **Обеспечение целостности данных** — в данном упражнении это не требуется, но может препятствовать постановке учебных опытов с таблицами.

9. Закройте диалоговое окно **Изменение связей** и в окне **Схема данных** рассмотрите образовавшуюся связь. Убедитесь в том, что линию связи можно выделить щелчком левой кнопки мыши, а щелчком правой кнопки мыши открывается контекстное меню, позволяющее разорвать связь или отредактировать ее.

10. Закройте окно **Схема данных**. Закройте программу Microsoft Access.

**Упражнение 3. Создание запроса на выборку.**

В этом упражнении мы создадим запрос на выборку информации о жестких дисков, имеющих емкость не менее 40 Гбайт при цене менее 100 условных единиц. Результирующая таблица должна содержать также адрес поставщика и номер его телефона.

1. Запустите программу Microsoft Access 2000 (**Пуск -> Программы -> Microsoft** **Access**).

2. В окне Microsoft Access включите переключатель **Открыть базу данных**, выберите ранее созданную базу **Комплектующие** и щелкните на кнопке ОК.

3. В окне **Комплектующие: база данных** откройте панель **Запросы**. Дважды щелкните на значке **Создание запросав режиме Конструктора** — откроется бланк запроса по образцу. Одновременно с ним откроется диалоговое окно **Добавление таблицы**.

4. В окне **Добавление таблицы** выберите таблицу **Поставщики** и щелкните на кнопке **Добавить**. Закройте окно **Добавление таблицы**.

5. В списке полей таблицы **Поставщики** выберите поля, включаемые в результирующую таблицу: **Компонент**, **Модель**, **Цена оптовая**, **Поставщик**, **Телефон**. Выбор производите двойными щелчками на именах полей.

6. Задайте условие отбора для поля **Компонент**. В соответствующую строку введите: **Жесткий диск**. Из таблицы будут выбираться не все изделия, а только жесткие диски.

7. Задайте условие отбора для поля **Цена оптовая**. В соответствующую строку введите: **< 100**. Из таблицы будут выбираться только изделия, имеющие цену менее 150 условных единиц.

8. Нам еще надо задать условие отбора по основному потребительскому параметру — емкости жесткого диска. Однако в таблице **Поставщики** такого поля нет. С другой стороны, в ней есть поле **Модель**, которое однозначно определяет параметры изделия. Благодаря тому, что по полю **Модель** у нас установлена связь с таблицей **Комплектующие**, мы получаем возможность ввести в запрос поле **Основной параметр**, взяв его из другой таблицы.

Добавьте список полей таблицы **Комплектующие** в верхнюю часть бланка запроса по образцу. Для этого щелкните правой кнопкой мыши в верхней области бланка и в открывшемся контекстном меню выберите пункт **Добавить таблицу** — откроется уже знакомое нам окно **Добавление таблицы**. Выберите в нем таблицу **Комплектующие**.

9. Двойным щелчком на поле **Основной параметр** в списке полей таблицы **Комплектующие** введите это поле в бланк запроса по образцу.

10. В строке **Условие отбора** столбца **Основной параметр** введите условие **> 40** (емкость диска более сорока гигабайт).

11. Закройте бланк запроса по образцу. При закрытии запроса введите его имя — **Выбор комплектующих**.

12. В окне **Комплектующие: база данных** откройте только что созданный запрос и рассмотрите результирующую таблицу. Ее содержательность зависит от того, что было введено в таблицы **Комплектующие** и **Поставщики** при их наполнении в упражнении 13.1. Если ни одно изделие не соответствует условию отбора и получившаяся результирующая таблица не имеет данных, откройте базовые таблицы и наполните их модельными данными, позволяющими проверить работу запроса.

13. По окончании исследований закройте все открытые объекты и завершите работу с программой Microsoft Access.

ДАННОЕ ПРАКТИЧЕСКОЕ ЗАДАНИЕ ОТСЫЛАТЬ НА МОЮ ПОЧТУ

в виде программы сделанной в Microsoft Access:

apulicheva@gmail.com

**СРОКИ ВЫПОЛНЕНИЕ ЗАДАНИЕ 06.04.2020 ДО 13:00**