Урок №2

Теория баз данных

Содержание

Модель реляционной базы данных. Первичные ключи

На прошлой паре мы рассматривали такое понятие, как проектирование и построение баз данных и ее основных объектов — таблиц. Итак, строить таблицы Вы уже умеете, но одного такого умения недостаточно. Ваша база данных должна быть эффективной и содержать такое количество таблиц, которое необходима. Кроме того, каждая таблица должна быть построена в соответствии с основными правилами построения таблиц, поскольку только в этом случае разработаная база данных будет реляционной и продуктивной.

Согласно упомянутым выше правилам построения, каждая таблица в реляционной базе данных должна состоять из полей, которые содержат характеристики данных, и записей, содержащих непосредственно сами значения. Каждая таблица должна иметь ***первичный ключ*** (*primary key*) — значение, которое уникально идентифицирует каждую запись в пределах таблицы. Первичный ключ, кроме того, что указывает на уникальность значений, позволяет обеспечить быстрый доступ к конкретной записи таблицы.

Значение первичного ключа размещается в отдельном поле или группе полей таблицы и, как правило, в имени содержится префикс или суффикс ID. Например, ProductID, IdGroup и тому подобное. Как правило, поле, содержащее

значение первичного ключа, размещают самым первым в списке полей.

Рассмотрим практический пример построения таблицы средствами СУБД MS Access с первичным ключом. Наша таблица будет содержать информацию о работниках предприятия и носить название ***Employee***. Учитывая факт, что первым полем (ключевым полем, то есть полем, что содержит первичный ключ) должен быть уникальный идентификатор, ее структура приобретет следующий вид:

EmployeeId	FName	LName	Position	Depart-ment	Phone-Department
01	Вася	Пупкин	Инженерр	115	21-21-21
02	Вова	Петров	Инженер	126	36-32-21
03	Оксана	Головач	Админи-стратор	126	36-32-21

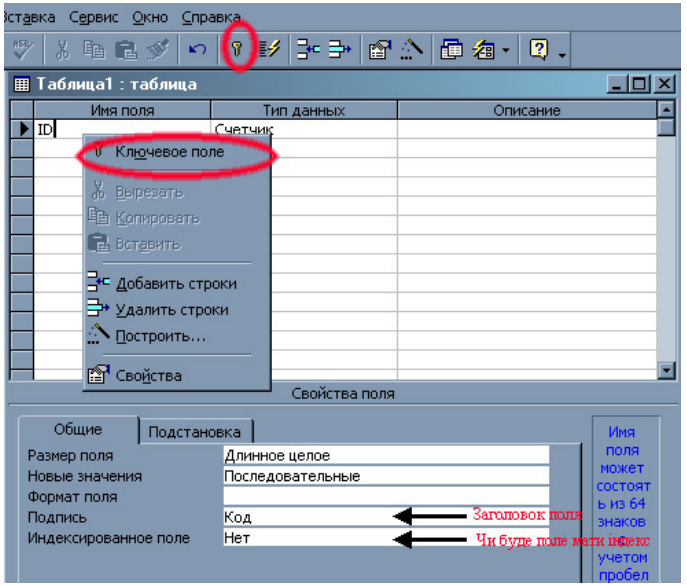


Рис. 1.

При создании поля ***EmployeeId***, необходимо добиться уникальности значений. Здесь можно пойти ***двумя путями***: сделать поле числового типа и позволить пользователю самому, при внесении новых значений в таблицу, указывать уникальные значения первичного ключа, или же позволить системе генерировать эти значения автоматически. В первом случае, гарантировать уникальность трудно, поскольку пользователи могут делать ошибки и присваивать одно и то же значение двум работникам (особенно в большой системе). Поэтому, лучше положиться на инструменты самой СУБД, которые гарантированно дадут необходимый результат — уникальные значения.

В MS Access для автоматизации процесса генерации уникальных значений в пределах таблиц используется тип «***Счетчик***». Следует отметить, что такой тип существует только в MS Access, в других СУБД для этих целей используются другие средства. Чтобы знать, какие именно, перед созданием таблицы нужно прочитать соответствующую документацию.

Но установить для поля тип счетчика недостаточно. Чтобы сделать его первичным ключом, нужно выбрать в контекстном меню данного поля пункт ***«Ключевое поле»*** или воспользоваться кнопкой с ключиком на панели инструментов СУБД.

По поводу закладки ***«Общие»***, то на ней мы можем задать размер поля, механизм генерации чисел (последовательные, то есть с шагом 1, или произвольные), формат поля, заголовок поля в режиме таблицы (по умолчанию в качестве заголовка используется имя поля в конструкторе), а также указать на то, будет ли данное поле индексом. Об индексах мы поговорим несколько позже, но, для информации, можно заметить, что они используются для быстрого доступа к данным в базе данных.

Создание других полей таблицы у Вас не должно вызвать затруднений, поскольку таблицы Вы уже создавали на прошлой паре. Обратим внимание лишь на следующее замечание, которое стоит взять себе за правило при построении таблиц базы данных: ***названия полей и самих таблиц нужно писать латинскими буквами, без пробелов и различных символов (кроме символа нижнего подчеркивания (_))***. Это правило связано с рекомендацией, чтобы при построении прикладного интерфейса к Вашей базе данных на определенном языке программирования у Вас не возникало трудностей в создании запросов с помощью языка SQL. Если поля Ваших таблиц или сами названия таблиц не будут соответствовать вышеописанным критериям, то при доступе к ним, в лучшем случае, Вам придется их названия брать в угловые скобки или другие управляющие символы. В худшем — такой способ не пройдет и придется полностью переделывать базу данных (☹).

Связи

Понятие связей между таблицами. Внешние ключи

Как Вы уже знаете, база данных может состоять из многих таблиц, которые содержат определенную информацию. В основном, информация, которую Вы в ней (базе данных) храните, имеет одно направление, например, содержит информацию о работе предприятия, данные о студентах и преподавателей академии и др. Эту информацию Вы логично можете разместить в разных таблицах, но каждая из этих таблиц будет автономной и не зависеть от других. Как же их связать?

Как Вы уже знаете, каждая реляционная таблица должна содержать первичный ключ, который уникально характеризует каждую ее запись. Но, кроме этой функции, он позволяет реализовать и связи между таблицами, благодаря которым данные из одной таблицы становятся доступны для другой. В таком случае, если база данных содержит несколько таблиц, работа в ней становится более эффективной, облегчается ввод данных в таблицу, снижается вероятность ошибок.

Связь реализуется за счет того, что в каждой из таблиц содержится по одному полю, которое имеет одинаковое значение. Кстати, эти поля могут иметь разные имена, достаточно лишь соблюдать однозначное соответствие данных. Для одной из таблиц таким полем является первичный ключ, а для второй таблицы — этим полем будет поле ***внешнего ключа*** (*foreign key*).

Итак, ***внешний ключ*** — это поле таблицы, значения которой совпадают со значением поля, которое является первичным ключом другой таблицы.

Следовательно, за счет пары первичный ключ — внешний ключ происходит связь данных между двумя таблицами.

Типы связей

Связь — это способ объяснить СУБД, каким образом следует выполнять выборку информации из таблиц баз данных. Между таблицами базы данных существуют 4 типа связей:

1. **ОДИН-К-ОДНОМУ (1:1):** каждой записи таблицы А соответствует только одна запись таблицы Б (или наоборот). Такой тип связей используется редко, поскольку практически все данные могут быть размещены в одной таблице. Она может быть полезна в случае, когда, например, целесообразно логично разделить одну громоздкую таблицу. Схематично это можно изобразить следующим образом:

Рис. 2.

Пример: существуют две таблицы, одна из которых содержит данные о работниках предприятия, а вторая — профессиональные сведения о них. В таком

случае, можно сказать, что между этими таблицами существует отношение один-к-одному, поскольку для одного человека (из первой таблицы) может существовать только одна запись, содержащий профессиональные сведения во второй таблице.

2. **ОДИН-КО-МНОГИМ (1: М):** одна запись таблицы А связана со многими записями таблицы Б, но одной записи таблицы В не может соответствовать несколько записей таблицы А. Схематическое изображение связи:

Рис. 3.

Пример: квартира может быть пустой, в ней может жить один или несколько жильцов.

3. **МНОГИЕ-К-ОДНОМУ (М: 1)** — обратное к предыдущему. Тип отношения между объектами зависит от Вашей точки зрения. Если рассматривать отношение жильцов к квартире, тогда формируется связь многие-к-одному.

4. **МНОГИЕ-КО-МНОГИМ (М: М)**. Возникает между двумя таблицами в случаях, когда:
   - одна запись из первой таблицы может быть связана более чем с одной записью из второй таблицы;
   - одна запись из второй таблицы может быть связана более чем с одной записью из первой таблицы.

Такая связь мало распространена, но если она существует, то обычно вводится промежуточная таблица (таблица-связь), которая состоит только из внешних ключей и связывает две таблицы между собой.

Пример: предприятие, на котором работают работники (один или более). Кроме того, вполне вероятна ситуация, что работники работают на нескольких предприятиях и эта информация хранится в базе данных. В таком случае, между этими таблицами (Работа и Работники) существует связь многие-ко-многим.

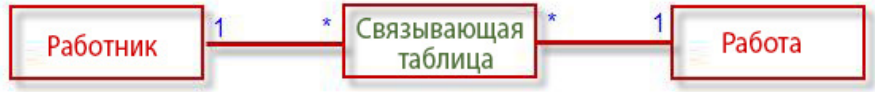


Рис. 4.

Существуют следующие ***правила, которым должны соответствовать типы данных связанных полей***:

- общие и связанные поля должны быть одинакового типа;
- если оба связанных поля имеют числовой тип, то они должны иметь и одинаковые значения свойства «***Размер поля***»;
- поле с типом данных «***Счетчик***» можно связать с числовым полем, у которого «***Размер поля***» имеет значение «***Длинное целое***».

Нормализация таблиц

Базовые основы нормализации. Нормальные формы

На прошлой паре мы рассматривали с Вами такое понятие как проектирование базы данных, но мы не останавливались на том, что же будет происходить после того, как Вы ее спроектируете. То есть, будет ли она оптимально спроектированной, иметь эффективную структуру, не существует ли избыточности (например, в двух таблицах повторяется одна и та же информация или нет), обеспечивается ли целостность данных, и тому подобное. Чтобы проверить ее на оптимальность не стоит ломать голову над придумыванием своих собственных методов анализа. Для этого достаточно при проектировании придерживаться такой процедуры как нормализация, которая позволит улучшить структуру Вашей базы данных и сделать ее максимально эффективной.

Нормализация — это пошаговый процесс замены одной таблицы (или набора таблиц) другими, имеющими более простую структуру. Другими словами — это процесс разбиения таблицы со сложной громоздкой структурой на две и более простых. Это приводит к исключению избыточности информации, ее упорядочению, экономии памяти, увеличению скорости доступа к данным, и тому подобное. На каждом этапе нормализации таблицы сводятся к определенному виду, что называют ***нормальной формой***.

Чтобы лучше разобраться с нормализацией и тем, когда таблицы будут являться нормальными формами, рассмотрим пример. Допустим, у нас существует таблица Employee, которая имеет следующую структуру и содержит определенный набор данных:

Employeeld	FullName	Position	Salary	Department	Phone-Department
1	Вася Пупкин	Инженер	4 500	115	21-21-21
2	Вова Петров	Инженер	5 000	127	21-17-87
3	Оксана Головач	Програмист	3 000	126	21-32-21
4	Оксана Головач	Администратор	2 000	126	21-32-21
5	Вова Маякін	Програмист	5 000	127	21-17-87

Проанализируем:

1. В данной таблице дублируется информация о должности и отделы работников.
2. Если изменится номер отдела, то небходимо менять его у всех работников данного отдела. Аналогичная ситуация и по телефонам.
3. Невозможно включить данные о новом отделе, пока не будут набраны данные о его работниках (парадоксально, но факт).

Поэтому, согласно правилам нормализации, нужно выполнить декомпозицию данной таблицы ***Employee*** и разбить ее на меньшие, которые имеют более оптимальную структуру. Каждая из таблиц должна быть нормальной формой, то есть должна отвечать трем основным правилам нормализации (или трем нормальным формам). Данные правила применяются последовательно, начиная с первой нормальной формы. Следует также учитывать тот факт, что

каждая следующая форма, начиная с первой, является более совершенной, чем предыдущая с точки зрения избыточности.

Начнем процесс. Сначала нужно выделить ***первую нормальную форму***. Для этого проанализируем наши данные на наличие в одном поле данных, которые можно разместить в разных полях, поскольку они приводят к избыточности. В нашем примере, поле FullName содержит неоптимальные данные, поскольку довольно часто необходимо осуществлять поиск данных по фамилии работника, а его имя неизвестно. В таком случае эти данные следует разместить в отдельных полях. Результатом таких преобразований будет таблица следующего вида:

EmployeeId	FName	LName	Position	Salary	Department	Phone-Department
1	Вася	Пупкин	Инженер	4 500	115	21-21-21
2	Вова	Петров	Инженер	5 000	127	21-17-87
3	Оксана	Головач	Програмист	6 500	126	21-32-21
4	Оксана	Головач	Администратор	6 500	126	21-32-21
5	Вова	Маякин	Програмист	5 000	127	21-17-87

Теперь поля нашей таблицы содержат атомарные значения, то есть такие, которые нельзя разделить на меньшие составляющие. Процесс поиска работника упрощен. Кроме того, данная таблица находится в первой нормальной форме (1НФ), поскольку каждый атрибут каждой записи таблицы (то есть каждая ячейка таблицы) содержит только атомарные (неделимые) значения. Иными словами, при 1НФ каждое поле должно содержать не более одного значения, которое нельзя разделить на части.

Второй этап нормализации — это выделение ***второй нормальной формы***. Если проанализировать более

подробно нашу таблицу, то видно, что значение многих полей повторяются. Так, один работник может работать на нескольких должностях, и одну должность могут занимать несколько работников. При этом человеку, который будет вносить данные в базу данных, придется часто вводить одинаковые значения, повышает вероятность ошибок при вводе, на поиск и исправление которых уйдет время. Чтобы этого избежать и еще больше повысить поиск данных по базе данных, нужно разбить таблицу на несколько меньших. Следует также не забывать, что каждая из таблиц должна содержать атомарные значения, иначе они не будут соответствовать 1НФ. В таком случае, нужно для каждой из таблиц повторять первый этап нормализации.

Итак, в результате упрощения мы получим 3 таблицы: ***Employee, Position и Department***. Каждая из этих таблиц имеет свой собственный первичный ключ и они независимы друг от друга.

Employee

EmployeeId	FName	LName	Salary
1	Вася	Пупкин	4 500
2	Вова	Петров	5 000
3	Оксана	Головач	6 500
4	Вова	Маякин	5 000

Position

PositionId	Name
1	Инженер
2	Администратор
3	Програмист

Department

DepartmentId	NumberDepartment	PhoneDepartment
1	115	21-21-21
2	126	21-32-21
3	127	21-17-87

Правило гласит, что таблица находится во второй нормальной форме (2НФ), если:

- она находится в первой нормальной форме;
- все данные, которые можно логически отделить от других, выделенны в отдельную таблицу;
- все поля, которые логически относятся к выделенным данным, вынесенны в эту таблицу;
- для каждой таблицы введен уникальный идентификатор.

Итак, наши таблицы находятся во второй нормальной форме, поскольку теперь по значению идентификатора любой таблицы можно найти все значения, которые ему соответствуют. Например, по значению поля DepartmentId мы получим информацию о его названии и телефон. Кроме этого, выделив отдельные таблицы, мы можем с легкостью получить списки всех отделов и работников.

Третий уровень нормализации — это выделение ***третьей нормальной формы***. Таблица находится в третьей нормальной форме (3НФ), если она удовлетворяет требования 2НФ и не одно из ее неключевых полей не зависит функционально от других неключевых полей. Простыми словами, определены все связи между таблицами и введены дополнительные поля для этого.

Чтобы наши таблицы находились в третьей нормальной форме нужно их связать и в случае необходимости ввести дополнительные таблицы. Для связи таблиц между собой вводим внешние ключи в таблицу Employee, ведь две другие с ней связанны.

EmployeeId	FName	LName	Salary	PositionId	DepartmentId
1	Вася	Пупкин	4 500	1	1
2	Вова	Петров	5 000	1	3
3	Оксана	Головач	6 500	3	2
4	Оксана	Головач	6 500	2	2
5	Вова	Маякин	5 000	3	3

Но этого мало, поскольку при установлении связей в нашей таблице присутствует дублирование данных, которое касается работников, то есть их имена и фамилии, а это не допустимо. Это связано с тем, что один работник может работать на нескольких должностях и одновременно с тем на одной должности может работать несколько сотрудников. Фактически, у нас образовался связь многие-ко-многим, в связи с чем, возникает необходимость ввести промежуточную таблицу-связь.

Employee

EmployeeId	FName	LName	Salary	DepartmentId
1	Вася	Пупкин	4 500	1
2	Вова	Петров	5 000	3
3	Оксана	Головач	6 500	2
4	Вова	Маякин	5 000	3

Position

PositionId	Name
1	Інженер
2	Адміністратор
3	Програміст

EmployeePosition

Id	EmployeeId	PositionId
1	1	1
2	2	1
3	3	3
4	3	2
5	4	3

Кстати, первичный ключ в промежуточной таблице не является обязательным.

После таких преобразований наши, таблицы будут находиться в третьей нормальной форме, поскольку все они содержат атомарные значения (требование 1НФ), данные не дублируются (требование 2 НФ) и содержат внешние ключи для связывания (требование 3НФ).

Таким образом, мы нормализовали нашу базу данных. Следует отметить, что все осуществленные нами преобразования, хотя и выполняются в строгом соответствии с теорией нормализации, интуитивно понятны и очевидны. С опытом процесс проектирования баз данных, выбора необходимой структуры таблиц и определение оптимальных связей между ними уже не будет для вас таким загадочным и сложным, как на первых этапах освоения теории баз данных.

И наш рассказ был бы неполным, если бы мы не сказали, что на самом деле третья нормальная форма — это не последний этап нормализации, существуют формы

и более высокого уровня: 4НФ, 5НФ и 6НФ. Однако ситуации, требующие форм такого уровня, очень редкий случай и поэтому мы их рассматривать не будем.

Кроме того, теоретики реляционных систем Кодд и Бойс предложили более жесткое определение для 3НФ, которое учитывает, что в таблице может существовать несколько возможных ключей. Таблица находится в нормальной форме Бойса-Кодда (НФБК), если любая функциональная зависимость между ее полями сводится к полной функциональной зависимости от возможного ключа.

Анализ и нормализация данных средствами СУБД MS Access

Кроме того, что у нас есть возможность собственноручно организовать нормальные формы, в СУБД MS Access 2000/2003/XP существует инструмент «***Мастер анализа таблиц***», который позволяет проверить структуру таблицы и, в случае необходимости, разделить ее на

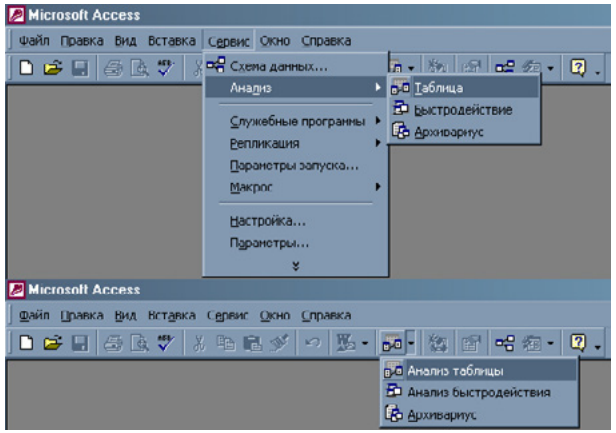


Рис. 5.

новые связанные таблицы, что приводит к эффективности хранения данных. Данный инструмент можно вызвать через пункт меню ***Анализ->Таблица*** или аналогичной кнопки на панели инструментов (Рис. 5).

Принцип работы анализатора очень прост. Для примера возьмем нашу еще не нормализованную первичную таблицу с информацией о работниках некоторой компании. Выбираем пункт панели инструментов ***"Анализ таблиц"*** и у нас автоматически запускается визард анализатора, на первых двух шагах которого рассказывают о том, для чего необходимо проводить анализ таблиц и как он осуществляется:

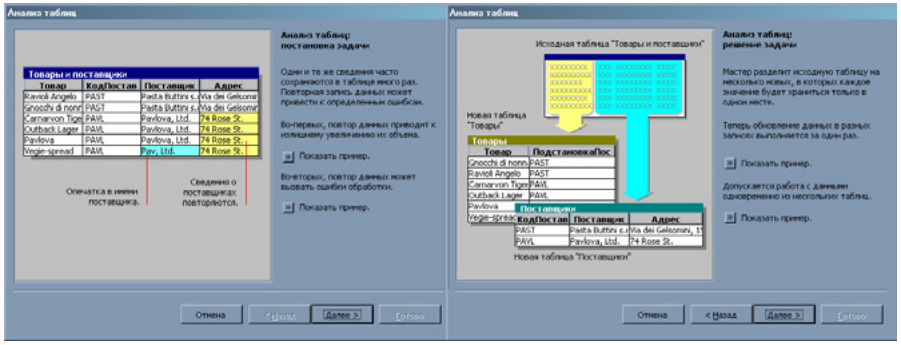
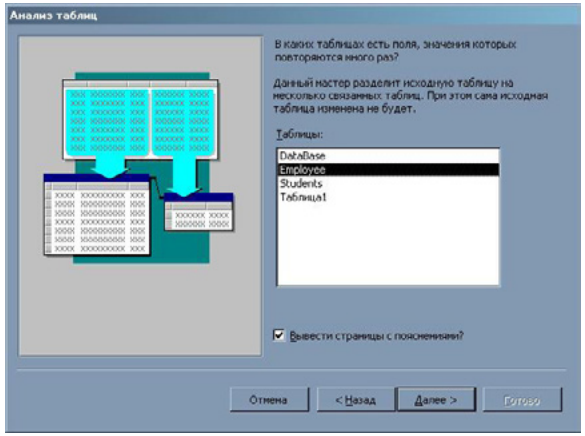


Рис. 6, 7

Далее нужно выбрать или положиться на опыт Визард в плане осуществления нормализации или сделать все вручную самому.



В случае выбора первого варианта мы имеем следующую ситуацию: Вам предложат сначала посмотреть на результат распределения с представлением возможности переименования таблицы. После этого предложат собственноручно добавить первичные ключи в тех таблицах, которые визарду неподвластны, то есть начальных.

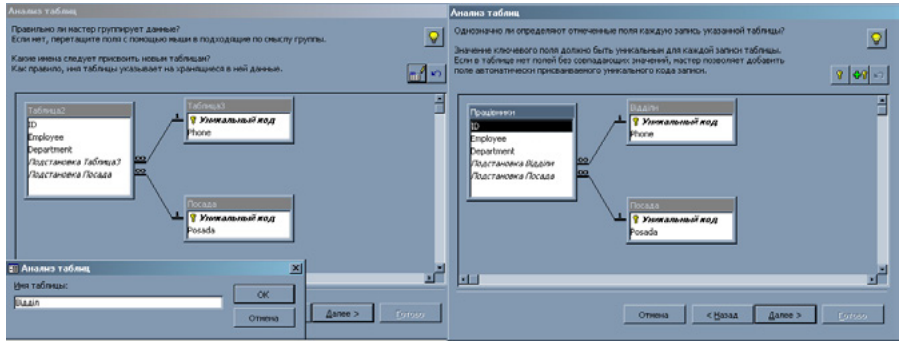


Рис. 8

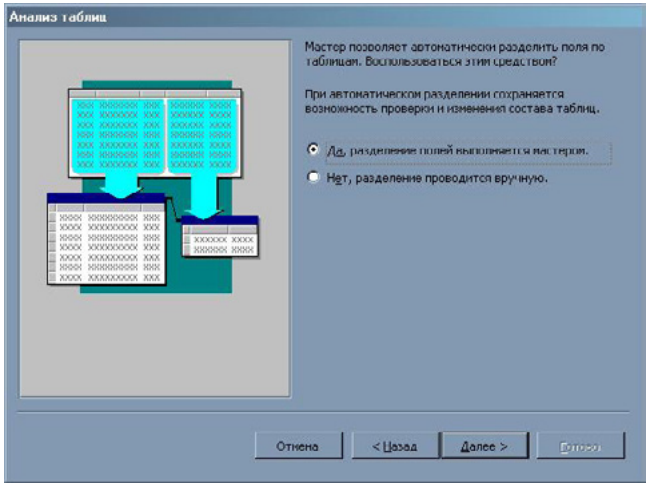


Рис. 9

В случае выбора второго варианта откроется похожее окно, в котором Вы должны собственноручно создать таблицы простым движением мышки. Для начала Вы

выбираете поле, которое необходимо вынести в другую таблицу, а затем кликаете на этом поле и, удерживая левую кнопку мыши, перетаскиваем его на пустую рабочую область. Переместив поле, отпускаем кнопку мыши. После этого у Вас образуется новая таблица и визард предложит ввести ее новое название. При желании добавить к новосозданной таблице еще одно поле, Вы проделываете почти подобные действия, то есть перетаскиваете желаемое поле с помощью мыши в зарезервированное для него место в другой таблице. После завершения всех действий по созданию оптимальной структуры таблиц сразу предоставляется возможность определения ключевых полей:

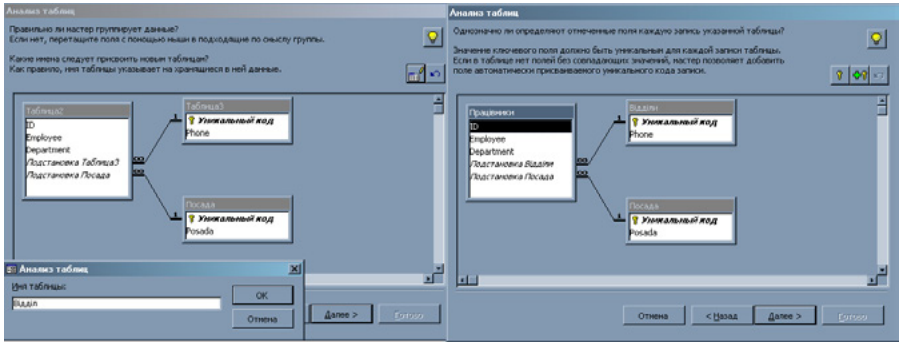


Рис. 10

В конечном итоге, мы получаем определенное на этапе анализа количество таблиц с уже перенесенными данными:

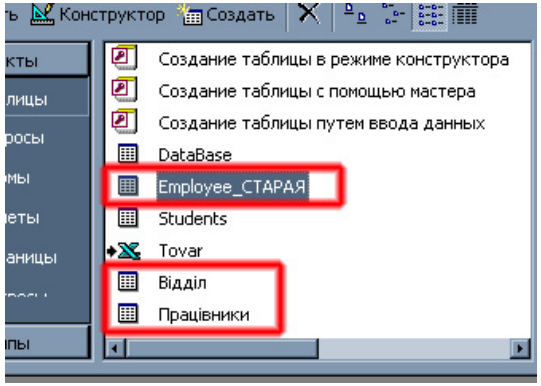


Рис. 11

К встроенным средствам автоматизации следует также отнести и «***Анализатор быстодействия***», анализирующий производительность БД и ее объектов, давая рекомендации по их улучшению.

Для примера возьмем базу данных и проанализируем ее с помощью данного инструмента

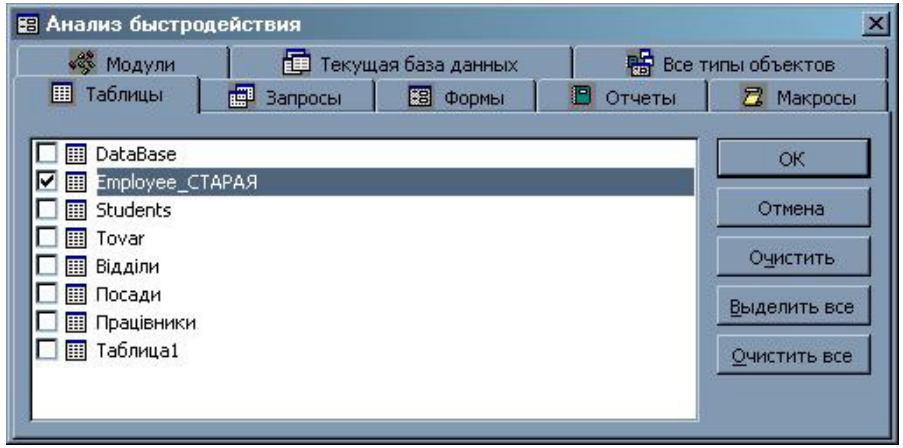


Рис. 12

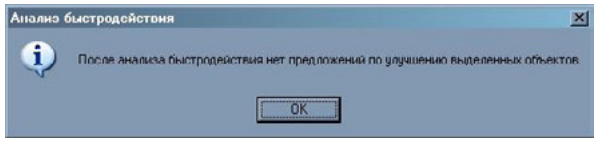


Рис. 13

Для этого в окне анализатора выбираем ту таблицу, запрос, отчет и т.д., мы хотим проанализировать на быстродействие. После нажатия кнопки «ОК», появится окно с рекомендациями.

Как видим, рекомендаций по нашей таблице нет, все у нас хорошо. Пойдем дальше и подвергнем анализу всю базу данных, а именно схему данных:

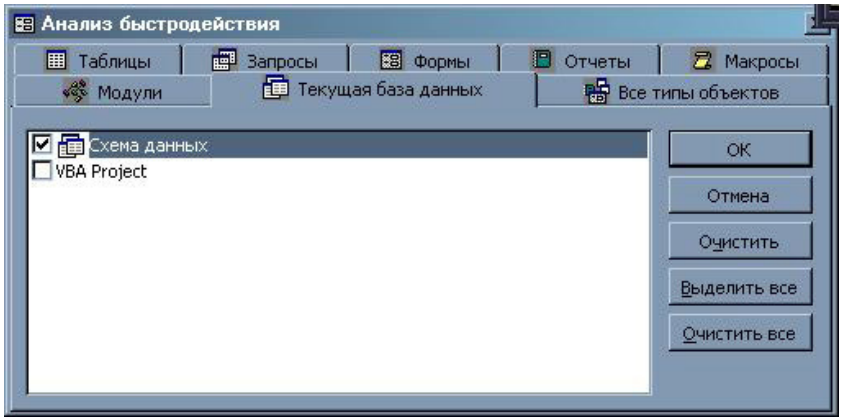


Рис. 14

А здесь не все так хорошо. В принципе, это и так понятно, поскольку наша база данных экспериментальная и большинство таблиц в ней не имеют друг к другу никакого отношения, и поэтому о связи между ними нет смысла и говорить.

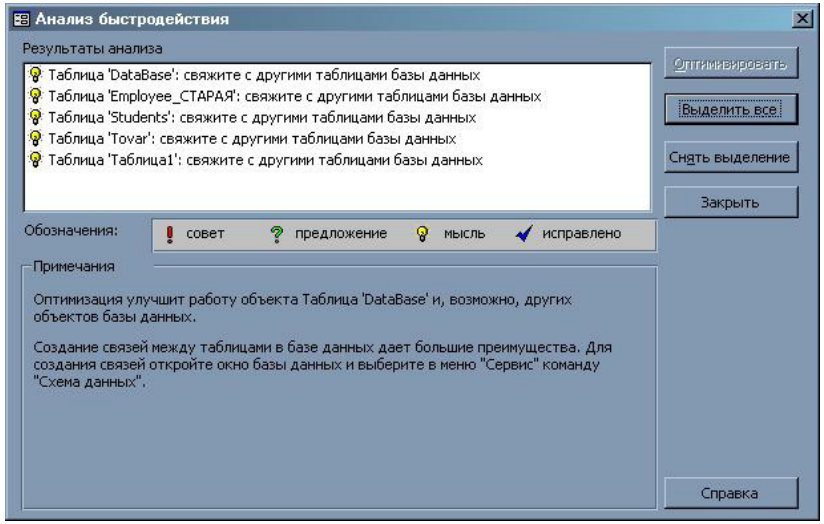


Рис. 15

Обеспечение целостности данных. Схема данных

Кроме того, что необходимо осуществить нормализацию данных, выделить нормальные формы, установить связи между ними, необходимо также обеспечить целостность данных, которые связаны. ***Целостность данных*** — это система правил, регулирующих взаимодействие между связанными таблицами и обеспечивающих корректность хранимых в таких таблицах данных. База данных представляет собой динамический механизм, в котором постоянно вносятся новые данные, удаляются ненужные и модифицируются уже существующие. В связи с этим, главная задача средств обеспечения целостности данных заключается в том, чтобы БД постоянно содержала точную и актуальную информацию.

Целостность данных говорит о том, что данные, введенные в общее поле двух связанных таблиц, должны совпадать. Из двух связанных таблиц одна называется родительской, а другая — дочерней, поэтому корректнее будет сказать, что данные, которые вводятся в поле внешнего ключа дочерней таблицы, должны совпадать с данными, которые хранятся в поле первичного ключа родительской таблицы. При любых попытках их модификации СУБД MS Access автоматически проверяет их значение. Если такое изменение нарушает установленную между таблицами связь, выдается сообщение о нарушении целостности данных.

Следовательно, при проектировании и создании базы данных необходимо предусмотреть все возможные случаи работы с ней и обеспечить целостность ее данных.

Для этого необходимо осуществить следующие действия:

1. Открываем инструмент «***Схема данных***». Для этого выбираем команду главного меню ***Сервис-> Схема данных*** или нажимаем кнопку «***Схема данных***» панели инструментов.

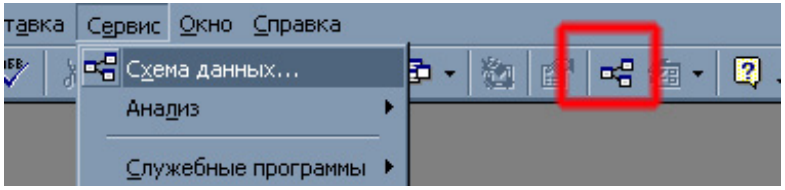


Рис. 16

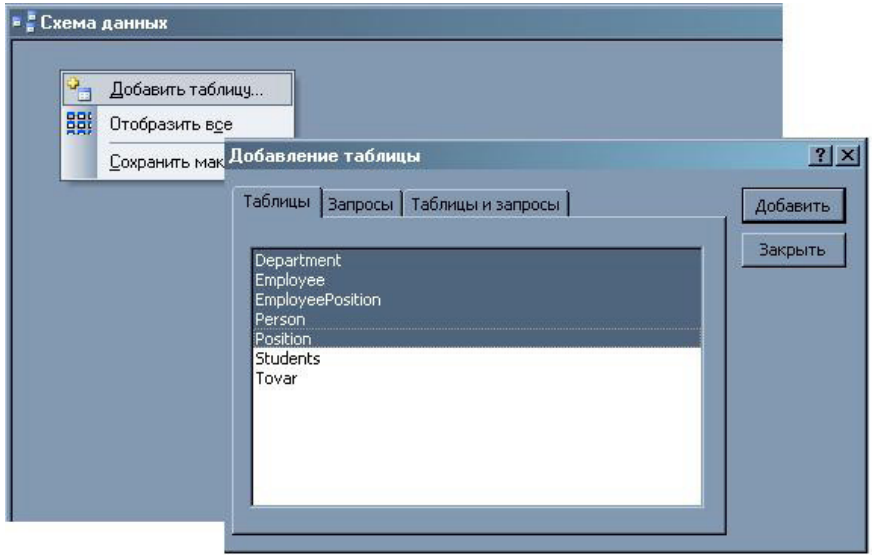


Рис. 17

Данный инструмент позволяет очень удобно просматривать структуру всех объектов базы данных, определять тип связей между ними и установить параметры целостности данных между таблицами.

2. Если в окне схемы данных еще не присутствуют таблицы или необходимые нам объекты, следует открыть диалоговое окно «***Добавление таблицы***» через контекстное меню схемы данных (рис. 17).
3. Создаем связи между добавленными в схему таблицами. Для этого выбираем таблицу, в которой общее поле выполняет роль первичного ключа (для нас это таблица ***Employee***). Кликаем на этом поле и, удерживая левую кнопку мыши, перетаскиваем его на другую таблицу, поместив точно над тем полем, которое является общим (в таблице ***Department*** это поле ***DepartmentId***). Переместив на поле, отпускаем кнопку мыши. Аналогичные действия проделывают и для других таблиц.

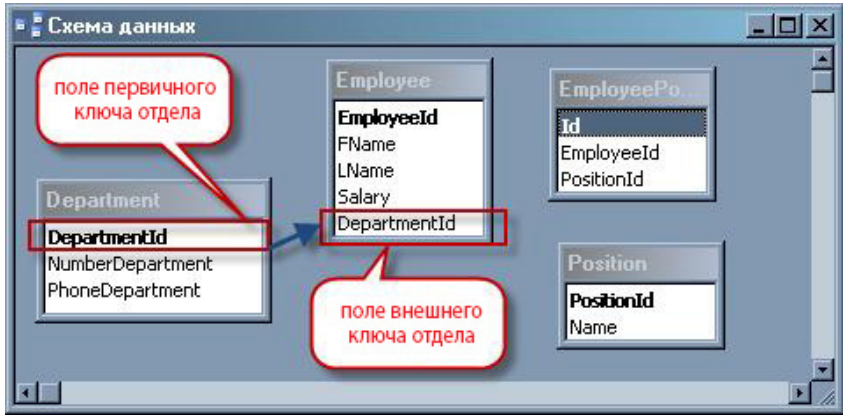


Рис. 18

4. После того как поле будет перенесено перед Вами появится диалоговое окно ***«Изменение связей»***.

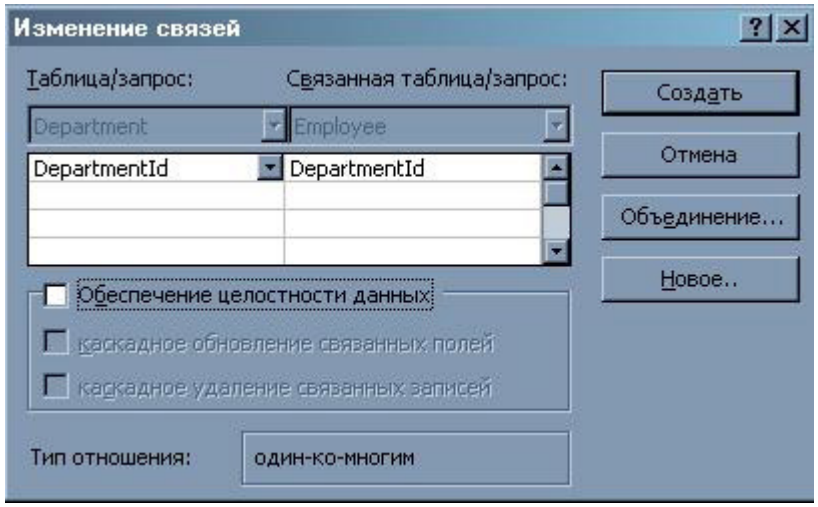


Рис. 19

В этом окне представлено несколько параметров. В верхней части окна отражены имена связываемых таблиц и их общие поля. Также доступна опция «Обеспечение целосности даннях», с помощью которой обеспечивается корректность связи между таблицами. Убедившись в том, что параметры заданы верно, установите связь между таблицами нажав кнопку ***«Создать»***. После этого между двумя связанными полями появится ***линия объединения*** (*линия связи*) (Рис. 20).

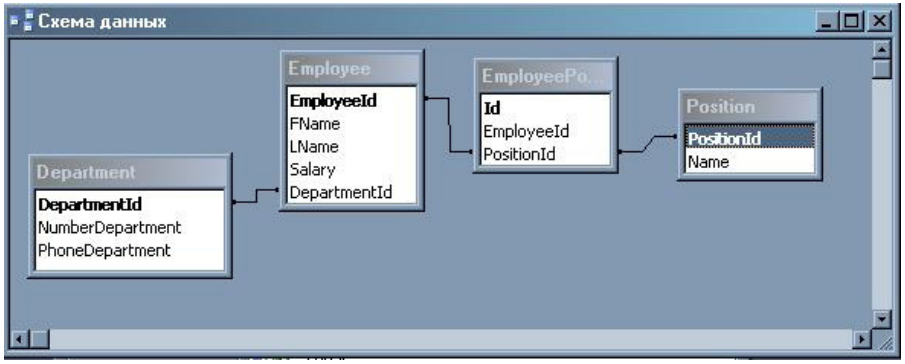


Рис. 20

5 Но это еще не все. У Вас образовались простые связи, которые просто объединяют две таблицы. Посмотрим внимательно на диалоговое окно ***«Изменение свиязи»***, которое, кстати, можно повторно вызвать с помощью пункта контекстного меню (или двойной клик по нему) ***«Изменить связь»*** линии связи или с помощью двойного клика по связи. Данное окно содержит еще две опции, которые позволяют активизировать автоматическое выполнение каскадного удаления и каскадного обновления данных.

Каскадное обновление связанных полей говорит о том, что при любом изменении данных первичного ключа в родительской таблице автоматически будут обновляться значения в соответствующем поле связанной таблицы. Если эта опция не задана, изменить значение ключевого поля первичной таблицы не удастся. Важно отметить, что если ключевое поле родительской таблицы — это поле типа «Счетчик», то нет смысла устанавливать данную опцию, так как значение поля счетчика изменить невозможно. Если же поле первичного ключа таблицы принимает участие сразу в нескольких связях с другими таблицами, то для корректной работы базы данных данную опцию следует установить для всех связанных таблиц.

Каскадное удаление связанных записей говорит о том, что при удалении записей в родительской таблице автоматически удаляются все соответствующие записи в дочерней связанной таблице. Если данная опция не установлена, то MS Access не позволит удалить записи родительской таблицы при наличии связанных записей в дочерних таблицах. Чтобы решить эту проблему, придется сначала удалить все записи в дочерних таблицах и только потом — запись в главной таблице.

Форматирование данных при вставке в таблицу. Ввод данных средствами СУБД

Итак, с принципами построения таблиц разобрались, переходим к ознакомлению с детальной информацией про форматирование отдельных полей, то есть того, как будут отображаться в них данные для пользователя.

Для демонстрации разработаем таблицу ***Person***, которая будет следующую структуру:

Id
Name
Age
Married
Salary
E-mail
Photo

Поле Id является ключевым и создается по стандартному принципу. Его настройки мы уже рассматривали сегодня ранее (см. Раздел 1).

Следующим полем будет Name, которое будет хранить имена лиц и будет иметь тип поля — «***Текстовый***». Как уже отмечалось, для более детальной настройки поля используется закладка «***Общие***», которая приобрела для тектового поля следующий вид (Рис. 21).

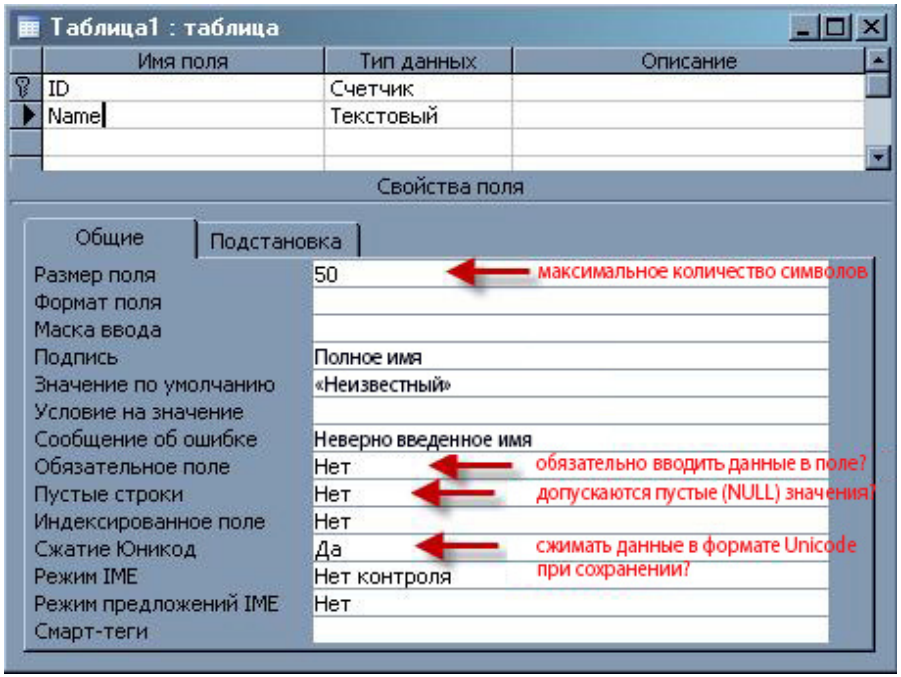


Рис. 21

Описание характеристик такого поля продемонстрировано на рисунке, поэтому остановимся лишь на тех, которые требуют более детального рассмотрения.

Таким образом, в поле ***«Значение поля по умолчанию»*** мы можем ввести те данные, которые будут в поле, если пользователь его не заполнит. Это значение нужно учитывать, поскольку необязательно все Ваши данные в Таблице будут заполнены.

В поле ***«Сообщение об ошибке»*** Вы можете задать текст, который будет размещен на диалоговом окне сообщения об ошибке в случае неверного ввода пользователем данных в поле.

Поле ***«Режим IME»***. IME (*Input Method Editor*) — это программа, которая позволяет распознавать символы азиатских алфавитов (китайского, корейского и японского), которые вводятся с помощью специальных кодов. Данное поле позволяет указать режим конвертирования, которое происходит при вводе данных в поле.

Поле ***«Режим предложений IME»*** позволяет определить режим приложений IME, которые будут применяться при введении данных.

Поля ***«Формат поля»***, ***«Маска ввода»*** и ***«Условие на значение»*** связаны и схожи между собой, поскольку все они предназначены для форматирования данных при вводе. Различия между ними заключаются в следующем:

1. ***Формат поля*** — данные проверяются после их ввода на соответствие определенному правилу.
2. ***Маска ввода*** — аналогично формату поля, но данные проверяются еще во время введения.
3. ***Условие на значение*** — данные проверяются также на этапе ввода, но на соответствие определенному условию. Например, чтобы пользователь не ввел отрицательную цену, или же дату, которая позже текущей и тому подобное.

Для форматирования ввода данных с помощью поля «Формат поля» используются следующие управляющие символы:

1. ***Для текстовых и МЕМО-полей*** используются следующие знаки:

- **>** — все что вводится, будет выводиться в верхнем регистре.
- **<** — все что вводится, будет выводиться в нижнем регистре.
- **@** — в поле нужно обязательно ввести обычный символ или символ пробела (space). Например, при формате поля (@@@) @@@ — @@ — @@ введенное значение 0551234567 приобретет следующий вид: (055) 123-45-67. Следует также учитывать, что заполнение осуществляется с младшего разряда, иначе вставляется пустой символ. Например, задан формат поля вида "@@@@@". При введении в данное поле имени "Оля" мы получим следующий результат "Оля".
- **&** — необязательный текстовый символ (формат по умолчанию). Так же, как и символ @ относится к каждому символа поля.

2. ***Числовые и денежные поля.*** Формат поля может состоять из четырех разделов, каждый из которых разделяется точкой с запятой (;):
   - 1-й раздел определяет формат положительных чисел;
   - 2-й — отрицательных чисел;
   - 3-й — нулевых значений;
   - 4-й — пустых (NULL) полей.

   Например, +0,0; -0,0; 0,0; "Не заполнено".
   Формат данных полей может задаваться следующими символами:

- **.** и **,** — Десятичный разделитель.
- **,** и **" "** (пробел) — Разделитель групп разрядов.
- **#**, **0** — Прототип разряда: **#** — необязательный, **0** — обязательный.
- **$**, **грн.**, **р.р.** — Текстовая константа. Например: **$ ## ###**, **00** или **## ###, 00 грн**.
- **%** — Процентный формат. Все введенные числа в данное поле будут умножаться на 100 и дописываться символ процента (%). Например, при формате поля # 0,00% вводится значение 0,34567. В результате мы получим в поле значение вида 34,57%.
- **E-** или **e-** — Экспонента, в которой положительная степень отображается без знака, а отрицательная с знаком. Например, при формате поля #, #### E-00 и введении числа 3456,7 Вы получите значение вида: 3,4567E03.
- **E +** или **e +** — Экспонента, в которой и положительная, и отрицательная степень отображается со знаком. Например, при формате поля **#**, **#### E + 00** и введении числа 3456,7 Вы получите значение вида: 3,4567E + 03.

3. ***Поля даты и времени.*** Для их форматирования используются следующие символы:
   - (**:**) — разделитель компонентов времени;
   - (**.**), (**/**) — разделители компонентов даты;
   - **d** — день месяца (1-31);
   - **dd** — день месяца (01-31);

- **ddd** — сокращенное название дня недели (Пн. -Вс.);
- **dddd** — полное наименование дня недели (понедельник — воскресенье);
- **ddddd** — встроенный короткий формат даты;
- **dddddd** — встроенный длинный формат даты;
- **w** — порядковый номер дня недели (1-7);
- **ww** — порядковый номер дня недели в году (1-53) ;
- **m** — порядковый номер месяца (1-12);
- **mm** — порядковый номер месяца (01-12);
- **mmm** — первые три буквы месяца (Янв–Дек);
- **mmmm** — полное название месяца (Январь-Декабрь);
- **q** — порядковый номер квартала в году;
- **y** — номер дня в году (1-366);
- **yy** — год без века (01-99);
- **yyyy** — год с веком (1999);
- **h**, **hh** — часы (1-24 / 01-24);
- **n**, **nn** — минуты (1-59 / 01-59);
- **s**, **ss** — секунды (1-59 / 01-59);
- **AM/PM** — 12-часовой формат времени с указанием времени до полудня буквами AM и после полудня — PM;
- **am/pm** — аналогично предыдущему, но обозначение будет прописными буквами: am, pm;

4. ***Логический тип.*** Формат поля состоит из трех разделов, каждый из которых разделяется точкой с запятой (;):

- 1-й раздел является пустым;
- 2-й раздел предназначен для отображения строчного значения, заменяет true;
- 3-й раздел предназначен для отображения строчного значения, заменяет false.

Формат данных полей может задаваться следующими символами:

- **(Пропуск)** — Выводит пропуск как текстовую константу.
- ****, **"Текст"** — Все символы в кавычках или после символа (\) являются текстовой константой.
- **!** — Выравнивание содержимого по левому краю.
- ***** — Выравнивание содержимого по правому краю. Причем, символ после звездочки выступает в качестве заполнителя.
- **[цвет]** — Задает цвет текста: Красный, Зеленый, Синий, Черный, Белый, Пурпурный, Желтый, Бирюзовый.

Маска ввода может содержать до трех разделов, которые также разделяются точкой с запятой:

1. Непосредственно сама маска ввода.
2. Раздел, который указывает на то, будет ли СУБД MS Access хранить при введении промежуточные символы в таблице, или нет. Значение 0 (ноль) указывает на то, что текстовые константы (например, дефисы или скобки в маске ввода телефонных номеров) будут храниться вместе с данными; значение 1 или пустое значение разряда указывает на то, что сохраняются только данные.

3. Раздел, определяющий символ, используемый для отображения пустых позиций в маске ввода, то есть куда должны помещаться данные, введенные пользователем. По умолчанию используется символ нижнего подчеркивания (_).

Символы, которые могут использоваться ***для создания маски ввода***:

- **0** — обязательнае цифра;
- **9** — необязательная цифра;
- **#** — необязательная цифра или символ;
- **L** — обязательная буква;
- **?** — необязательная буква;
- **A** — обязательная цифра или буква;
- **a** — необязательная цифра или буква;
- **&** — что угодно и необязательно;
- **C** — что угодно и обязательно;
- **!** — Заполнение данных справа налево;
- **\, "текст"** — все символы в кавычках или после символа (\) являются текстовой константой.

Чтобы проверить результат, которого удалось достичь путем настройки данного поля, нужно ввести в него данные. Для этого нужно перейти в режим таблицы с помощью кнопки на панели инструментов «Вид» (*см. Предыдущий урок*).

Следующим шагом добавляем поле, которое будет хранить возраст лица — ***Age.*** Вставим ему числовой тип поля. Закладка ***«Общие»*** несколько изменилась: некоторые

характеристики исчезли, так как необходимость в них отпала, и добавилась дополнительная — ***«Число десятичных знаков»***, которая позволяет настроить формат вывода данных для действительных значений.

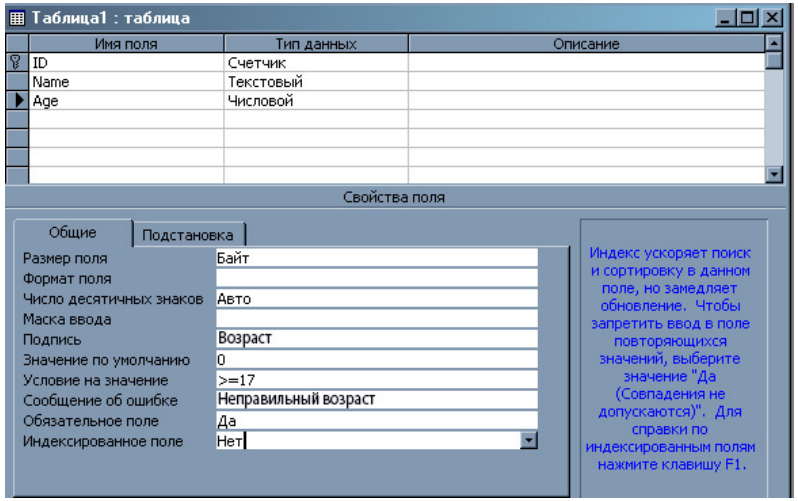


Рис. 22

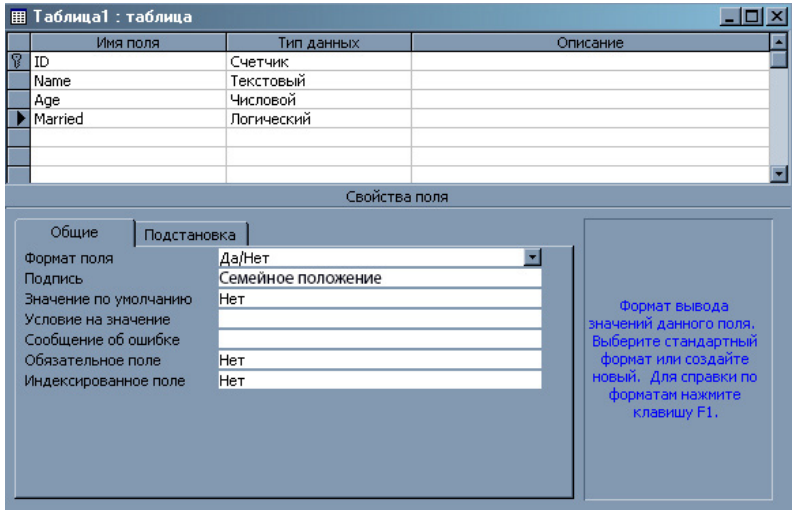


Рис. 23

Поле, содержащее информацию о семейном положении (***Married***) уместно сделать логического типа, поскольку оно может принимать только одно из двух значений: да или нет, женат или холост (Рис. 23).

Если после настройки данного поля перейти в режим таблицы, то в данном поле появился переключатель (***checkbox***), который контролирует ввод необходимых данных (женат или нет).

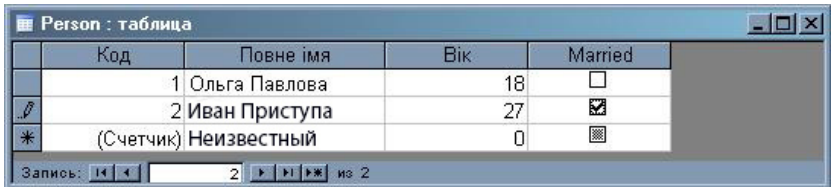


Рис. 24

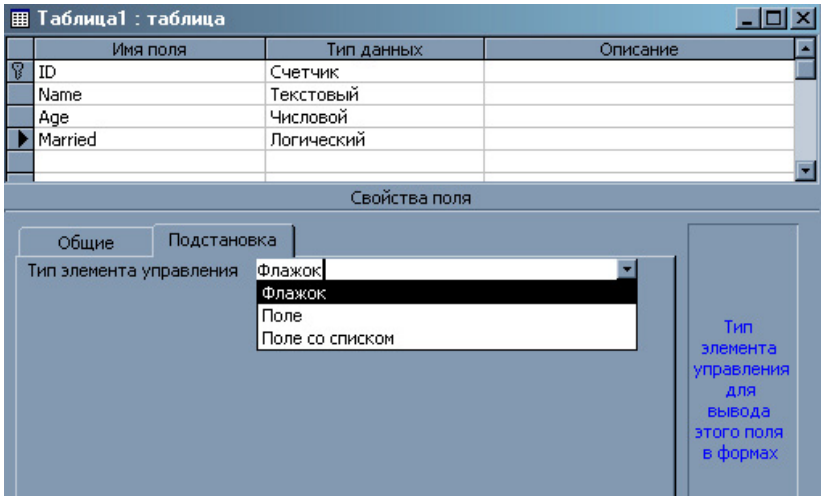


Рис. 25

Но если Вас не устраивает поле с переключателем, это можно изменить. Например, предоставить пользователю

возможность выбирать необходимые данные из выпадающего списка или вводить с клавиатуры. Для того чтобы это сделать нужно снова перейти в режим конструктора и перейти на закладку «***Параметры***».

На этой закладке есть только одно поле с характеристикой, которая определяет тип элемента управления, используемого для представления данных, содержащихся в данном поле (рис. 25).

Хотя вариантов действий — три, но для логического типа данных будут работать только 2 первых. Для установления своих значений нужно настроить «Формат поля», а при вводе данных вводить значение –1 (для истинного значения — true) и 0 (для false).

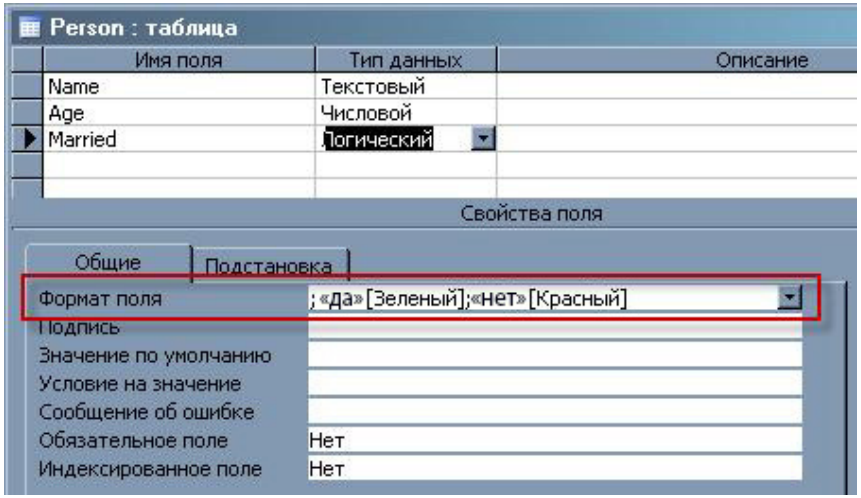


Рис. 26

Если Вы все же хотите использовать поле со списком, тогда придется прибегнуть к хитрости. Во-первых, изменить тип поля на «Текстовый». Затем на закладке

«Общие» установить условие на значение следующего содержания: "да" Or "нет". на закладке «Подстановка» при выборе элемента управления «Поле со списком» появятся дополнительные поля, с помощью которых можно его настроить.

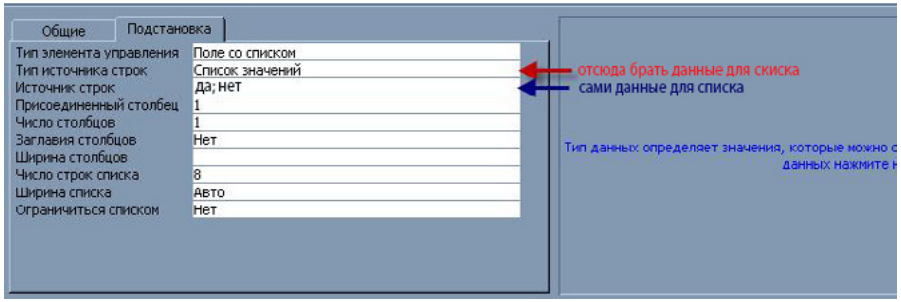


Рис. 27

Самыми основными характеристиками являются:

- «***Тип источника строк***», в котором указывается тип источника данных: другая таблица или запрос, список полей или список значений.
- «***Источник строк***» позволяет конкретизировать данную информацию: написать запрос на выборку данных или указать набор значений, которые будут в выпадающем списке.

Все остальные — вспомогательные, то есть позволяют собственно настроить вывод данных: указать количество элементов списка, его ширину и тому подобное.

Следующее поле — поле ***DateOfBirth*** (*дата рождения*) с типом поля ***Дата/Время***. Нового в списке характеристик закладки «Общие» ничего нет, новизна заключается только в изменении формата поля.

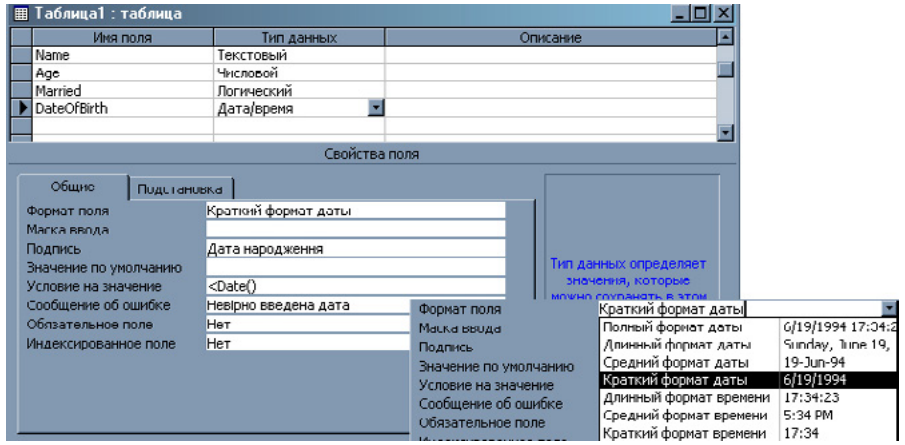


Рис. 28

Поле ***Salary*** (*Оклад, заработная плата*) сделаем «денежным» и отформатируем его.

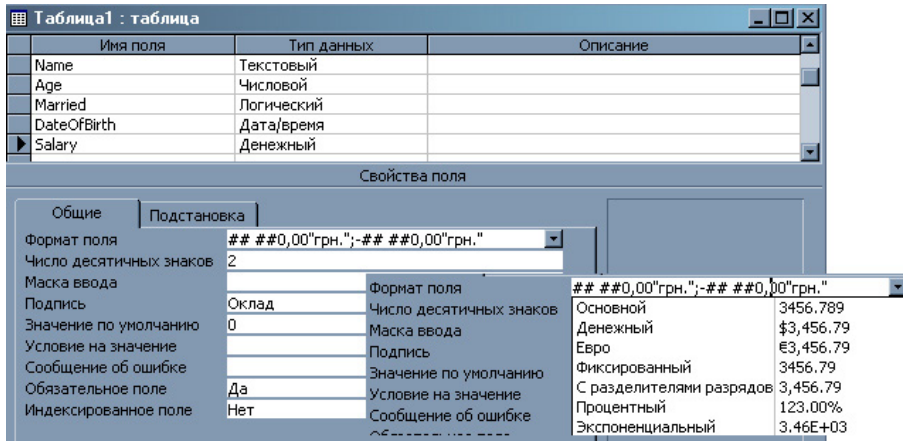


Рис. 29

Как видно из рисунка, необходимого формата данных (для указания величины денежного оклада в гривнях) у нас не нашлось, но мы уже познакомились с тем, как создать свой собственный формат, поэтому создадим его сами.

Для поля, которое содержит ***E-mail адрес*** лица, если таковой у него есть, лучше сделать формат типа ***«Гиперссылка»***.

Поексперементируем с данным типом поля.

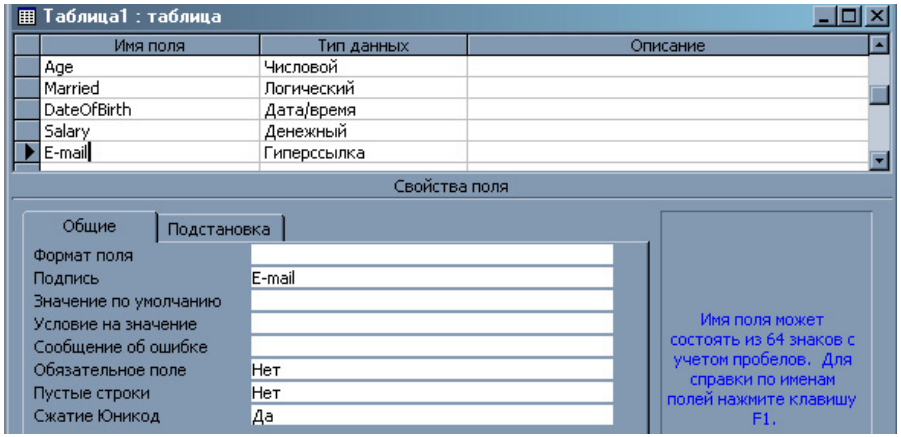


Рис. 30.

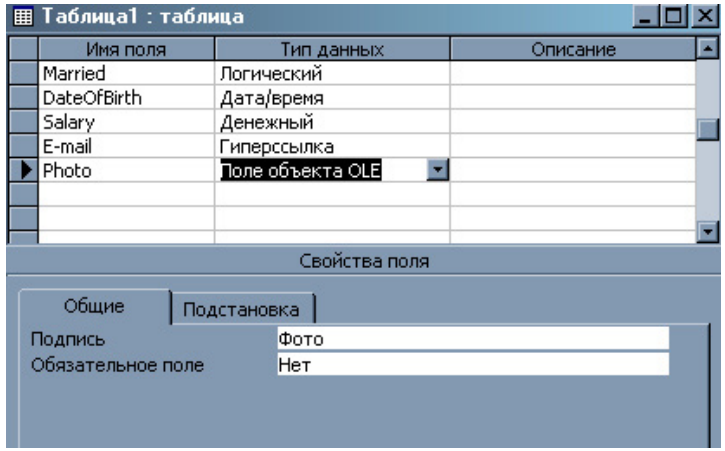


Рис. 31.

Поле Photo, содержащее фото работника. У нас осталось только два простых типа полей, которые мы еще не

рассматривали подробно: ***поле MEMO*** (характеристики которого ничем не отличаются от характеристик обычного текстового поля; единственная маленькое отличие — это максимальное количество допустимых символов, о чем уже говорилось на первом уроке) и ***поле объекта OLE***. Последнее идеально подходит для сохранения необходимых нам данных, поэтому его и выберем (Рис. 31).

Закладка «Общие» для такого типа полей не получила разнообразия — здесь только две характеристики, причем, уже хорошо Вам знакомых. После этого перейдем в режим таблицы и посмотрим на это поле со стороны пользователя: как в такое поле вводить данные? Для этого нужно в контекстном меню данного поля выбрать пункт "Добавить объект".

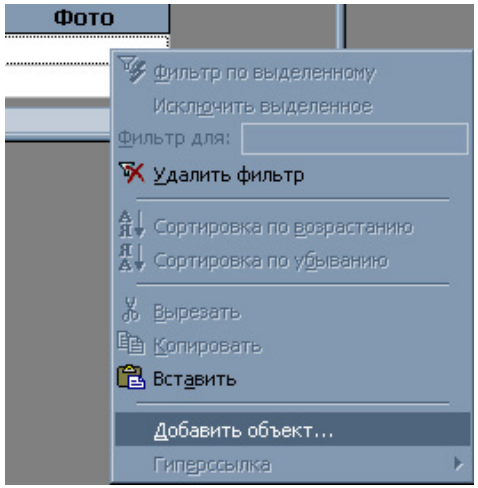


Рис. 32.

После этого появится диалоговое окно с возможностью выбора фотографии или можно на ходу создать фотопортрет человека (☺):

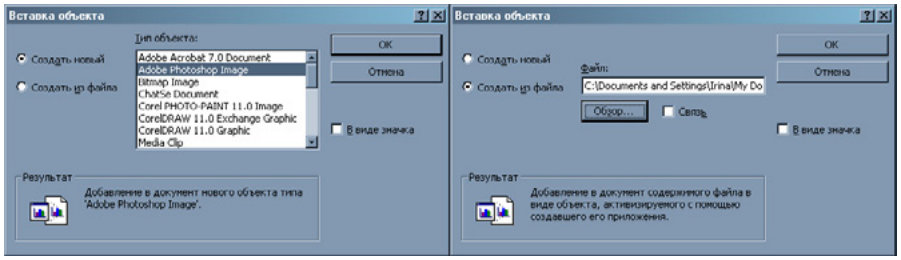


Рис. 33

Вы выбираете то, что Вам больше подходит и закрываете это окно.

Импорт и связывание таблиц

Импорт электронных таблиц

Возникают различные ситуации в жизни и вполне вероятно, что возникнет необходимость импортировать данные из других форматов в формат базы данных MS Access. Очень часто такая ситуация возникает, если создать интерфейс к базе данных другого формата. В частности, поддерживается импорт данных из формата электронных таблиц и текстовых документив. В этих случаях данные должны быть правильно оформлены, иначе импорт будет невозможен.

Схематично такое взаимодействие можно изобразить следующим образом:

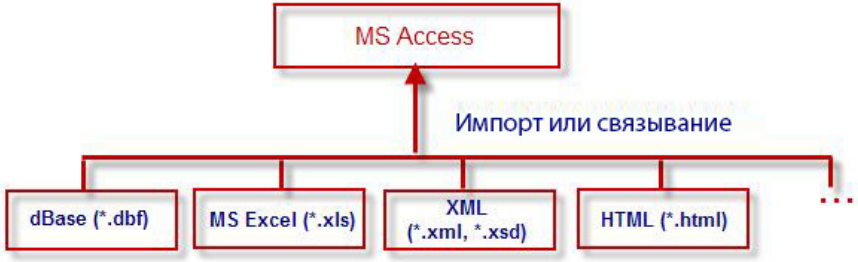


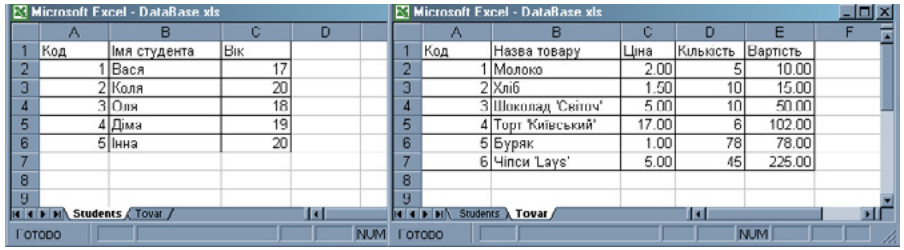
Рис. 34

Для лучшей наглядности рассмотрим импорт таблиц с данными нескольких форматов, а начнем с получения данных в формате MS Excel. Для этого создадим небольшую

БД с таблицами в MS Excel, оформленных в соответствии со следующими требованиями:

- в таблице должны быть только данные;
- каждая таблица должна размещаться на отдельном листе;
- формулы не копируются, поэтому следует помнить, что поля с ними будут после импорта пустыми. Но для того, чтобы такой ситуации не случилось, перед импортом Вы можете перевести формулы в числа.

В результате наши электронные таблицы для импорта будут иметь следующий вид:



Microsoft Excel - DataBase.xls

	A	B	C	D
1	Код	Ім'я студента	Вік	
2	1	Вася	17	
3	2	Коля	20	
4	3	Оля	18	
5	4	Діма	19	
6	5	Інна	20	
7				
8				
9				

Students | Tovar

Microsoft Excel - DataBase.xls

	A	B	C	D	E	F
1	Код	Назва товару	Ціна	Кількість	Вартість	
2	1	Молоко	2.00	5	10.00	
3	2	Хліб	1.50	10	15.00	
4	3	Шоколад 'Світоч'	5.00	10	50.00	
5	4	Торт 'Київський'	17.00	6	102.00	
6	5	Буряк	1.00	78	78.00	
7	6	Чіпси 'Lays'	5.00	45	225.00	
8						
9						

Students | Tovar

Готово NUM

Готово NUM

Рис. 35

Как видно на картинке, наша электронная база данных содержит два листка: на первом размещается информация о студентах, а на втором — о товаре: его название, количество, цена за единицу или килограмм и общая стоимость имеющегося товара той или иной категории, исчисляемая по формуле.

Импортируем ее в базу данных MS Access. Для этого нужно вызвать контекстное меню в окне объекта «Таблицы» и выбрать пункт меню ***«Импорт»*** и необходимую базу данных, то есть наш файл в формате * .xls.

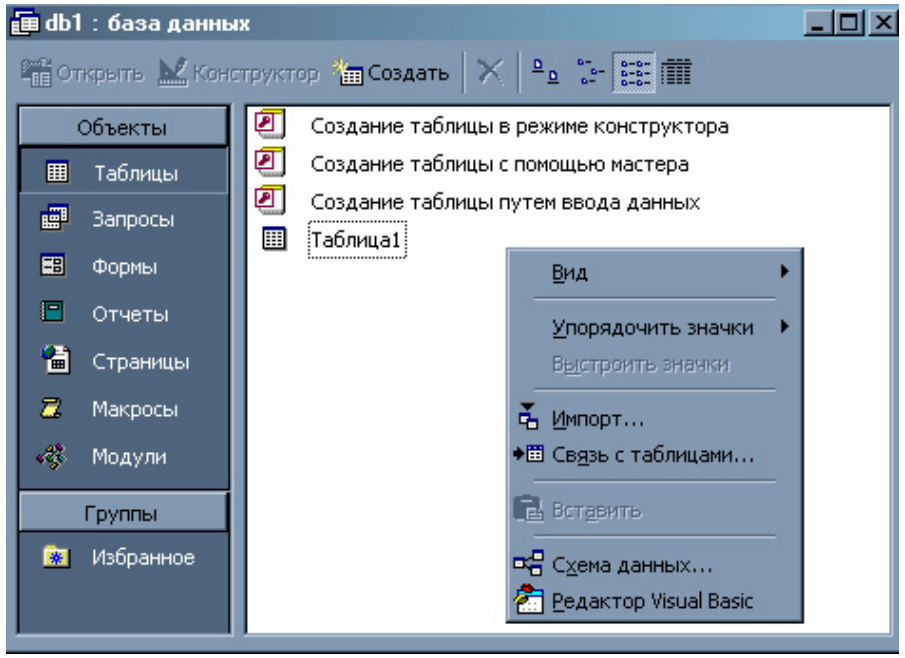


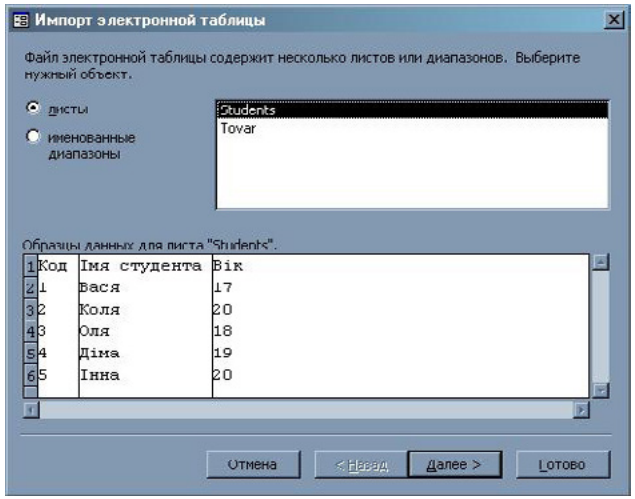
Рис. 36.

Стоит отметить, что при импорте таблица помещается в базу данных навсегда, то есть все изменения останутся только в MDB-файле, в файле MS Excel они отображаться не будут.

После избрания необходимого документа для импорта MS Access запустит визард, который позволит настроить импортируемые данные.

Поскольку MS Access понимает каждый лист нашего файла как отдельную таблицу БД, а имя листа — как имя этой таблицы, то на первом шаге Вам предложат выбрать ту таблицу, которую Вы хотите импортировать. Кстати, если в таблице есть именованные диапазоны, то они также будут восприняты как отдельные таблицы.

Рис. 37



После этого, необходимо указать, откуда нужно взять имена полей: из таблицы или будут созданы автоматически.

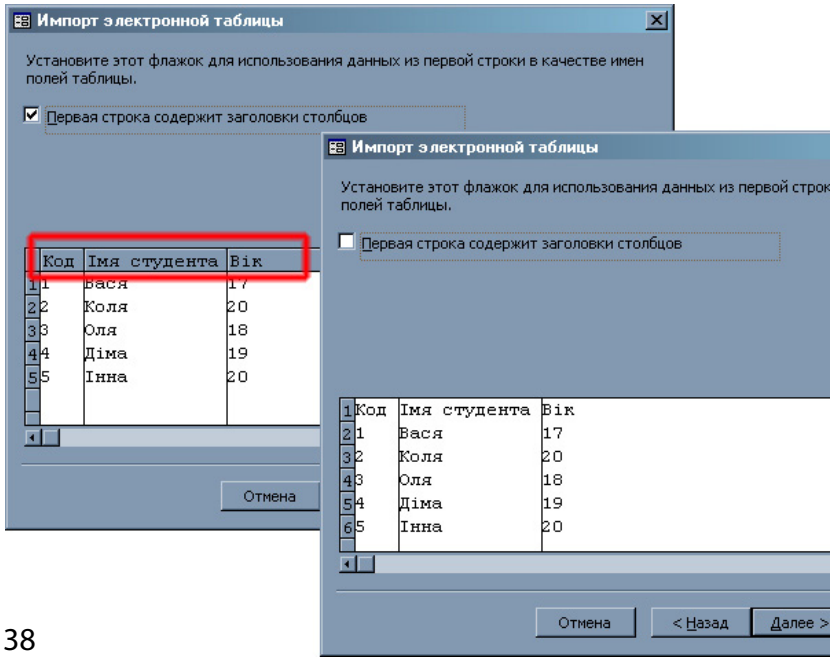


Рис. 38

На этом обязательные шаги завершаются, и Вы можете нажимать на кнопку «Готово», но поскольку существует еще ряд шагов, то стоит все же их рассмотреть на всякий случай.

Итак, следующий шаг позволяет определиться с тем, куда добавлять импортируемых данные: в уже существующую таблицу (если, конечно, формат совместим) или создать новую.

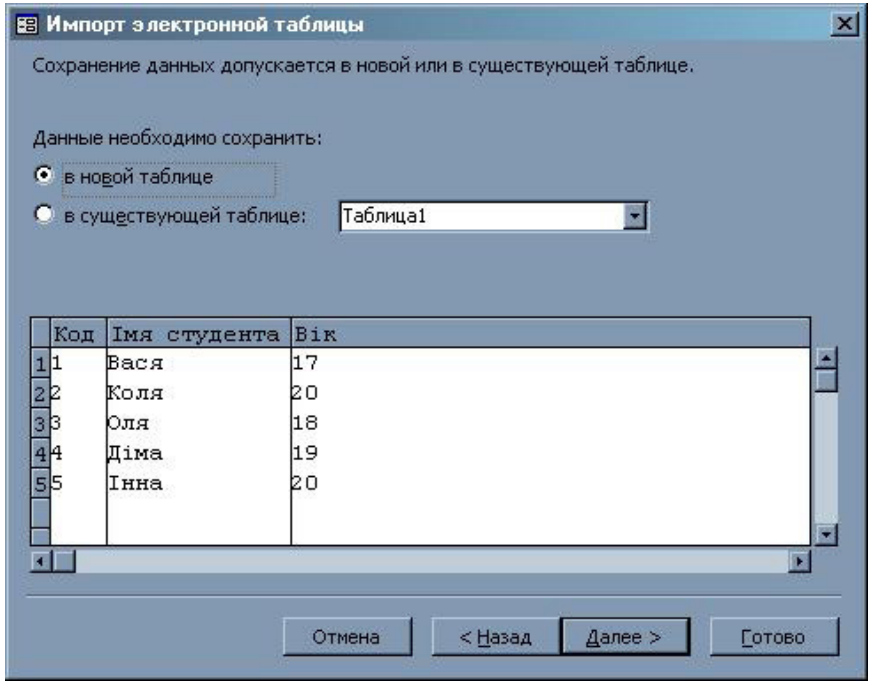


Рис. 39

При выборе второго варианта и перехода к новому шагу, данные импортируются в выбранную из списка таблицу. При выборе же первого — необходимо пошагово создать новую таблицу.

Далее нужно описать каждое поле таблицы, после чего указать ее первичный ключ, то есть, какое поле будет ключевым.

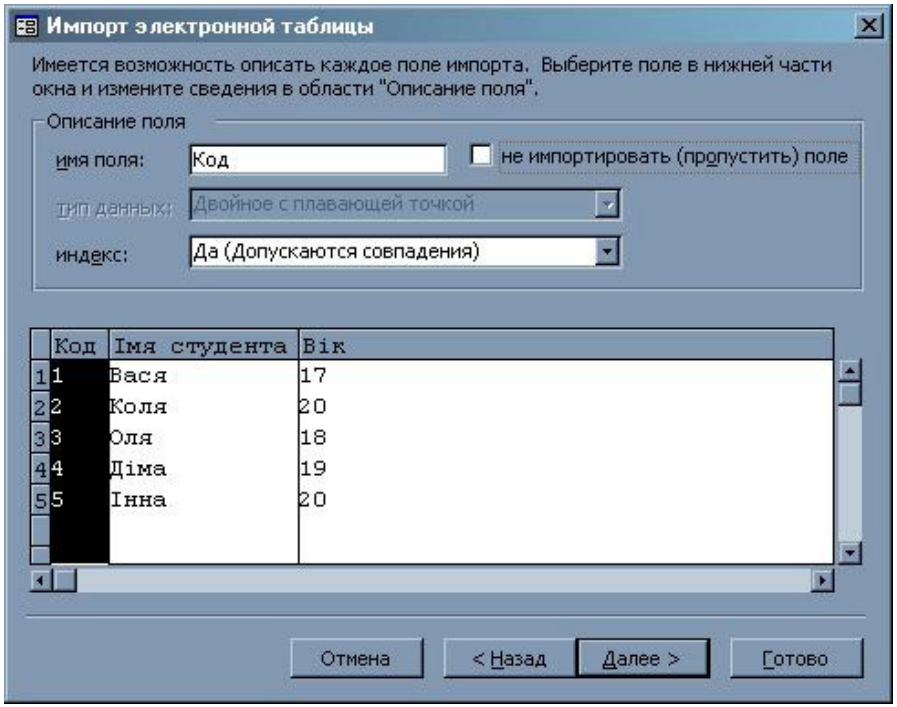


Рис. 40

В самом конце вводим новое название таблицы, если Вы не согласны с предложенной, и с уверенностью завершаем работу Визард.

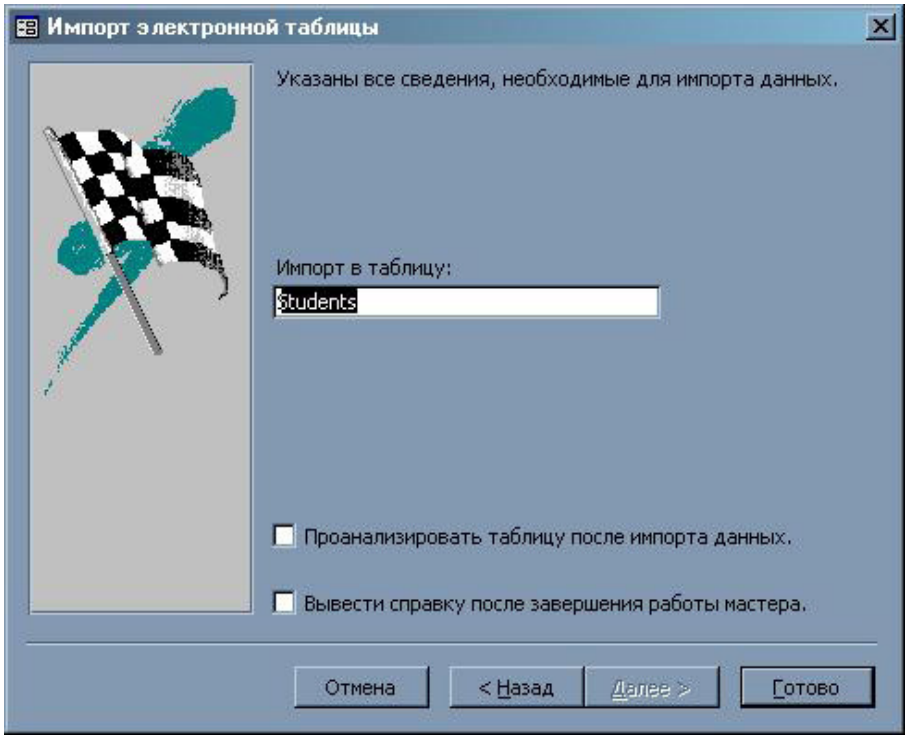


Рис. 41

Работа завершена. Таблица импортирована.

6.2. Импорт текстовых файлов

Следующим шагом рассмотрим импорт текстовых файлов, поскольку такие возможности дают много программ. При импорте текстовых файлов выделяют несколько понятий:

1. ***Сепаратор*** (*разделитель*) — это символ, отделяющий одно поле от другого.
2. ***Ограничитель*** — это символ, в середине которого содержатся поля.

Сам импорт (и экспорт тоже) может осуществляться в одном из ***двух вариантов***:

1. Фиксированной (постоянной) ширины
2. С сепараторами

Итак, начнем. Создадим условную таблицу со значениями в текстовом файле. Для этого воспользуемся многофункциональным текстовым редактором «Блокнот» (☺).

После этого импортируем созданный файл, выбрав пункт меню «Импорт». На первом этапе необходимо указать, как необходимо осуществить импорт:

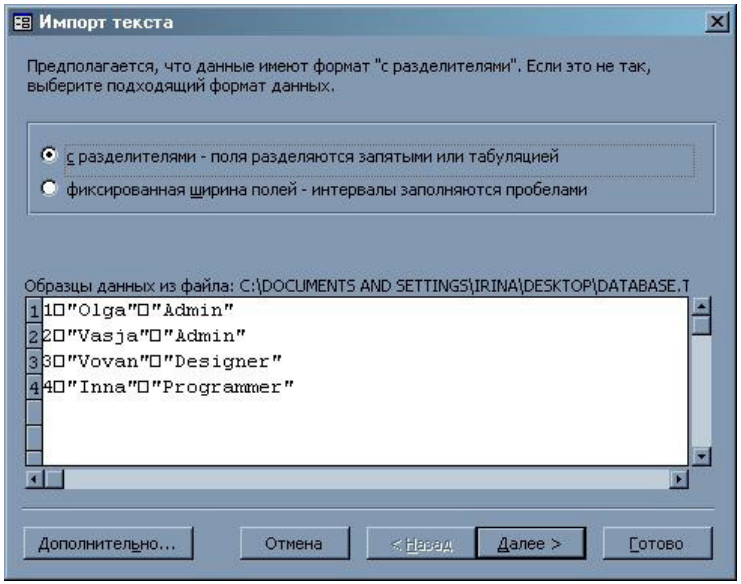


Рис. 43

Разница в следующих действиях будет видна только на следующих шагах, поэтому попробуем и один, и второй способ:

1. ***С разделителем***:

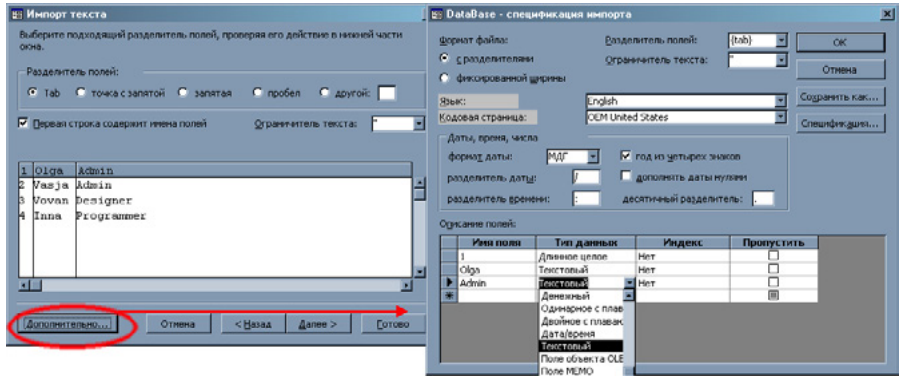


Рис. 44

2. С ***фиксированной шириной полей***:

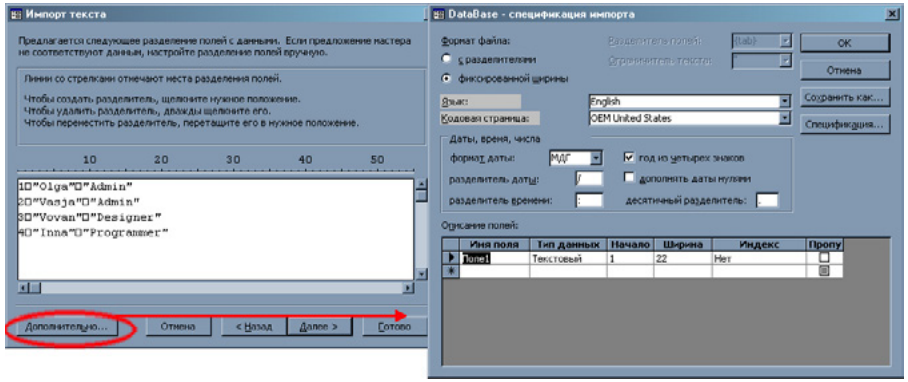


Рис. 45

Остановимся на создании полей с разделителем и идем дальше. А дальше нам предлагают один из вариантов сохранения импортируемых информации: создать новую

таблицу или поместить новые данные в уже существующую. Выбираем для сохранения новую таблицу, поскольку пока в нашей БД отсутствует таблица с похожей структурой:

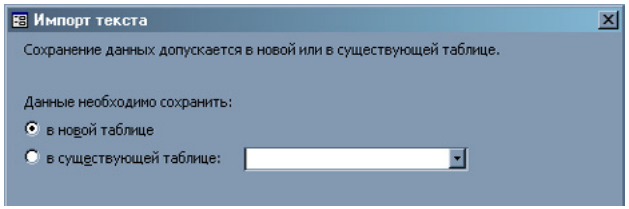


Рис. 46

Следующие шаги Вам уже знакомы: сначала описываем каждое поле импорта, затем задаем первичный ключ и завершаем работу над импортом наших данных:

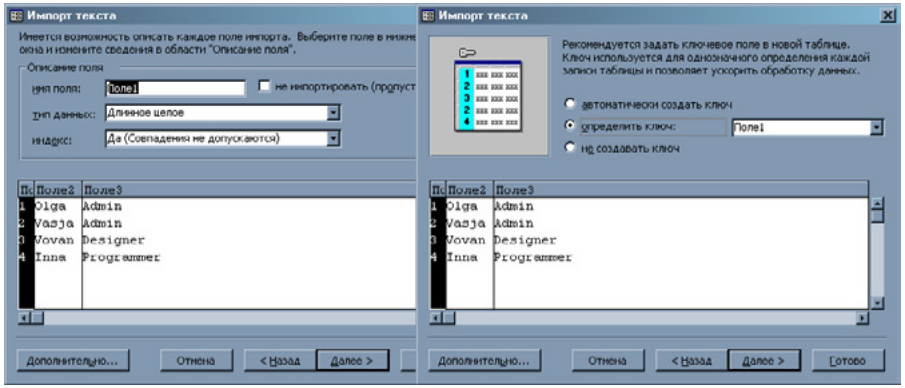
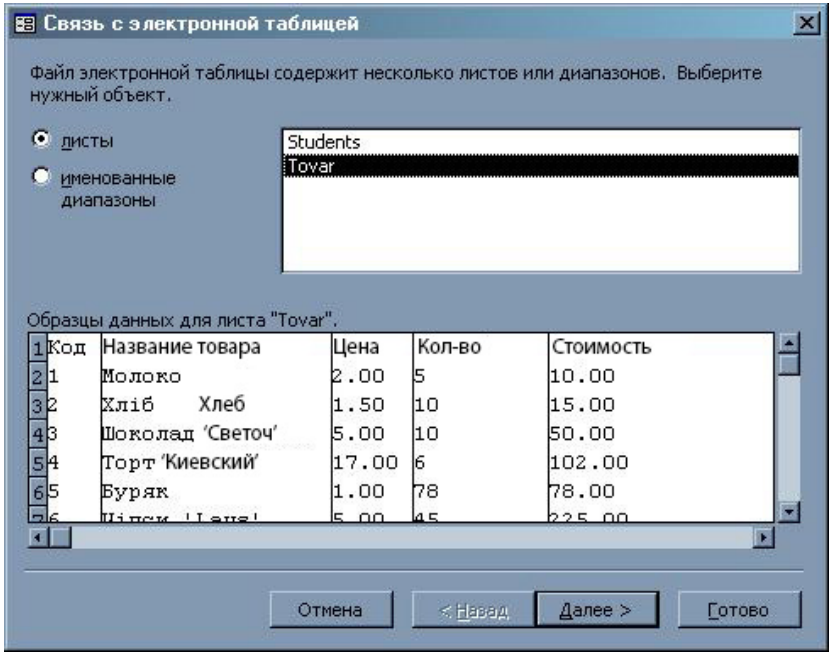


Рис. 47

1.2. Связывание с таблицами других баз данных

Кроме импорта существует и другая возможность работы с базами данных, организованных в другой формат (отличный от MS Access) — это ***связь с таблицами***. При связывании таблицы, изменения, которые Вы будете осуществлять в MDB-файле, будут отображаться и в связанной базе, и наоборот. Этот вариант более распространен, в отличие от импорта, поскольку MS Access неэффективно работает с данными большого объема, но позволяет создать для них удобный интерфейс, не зная при этом языка программирования.

Связывание на этапе реализации ничем существенно не отличается от импорта, разница только в конечном результате. Для демонстрации связывания возьмем второй лист базы данных формата MS Excel, созданный ранее. Итак, для связывания в контекстном меню окна базы данных выбираем пункт ***«Связь с таблицей»*** и необходимый документ. После этого запускается визард, первый шаг которого связан с избранием необходимой таблицы.



Далее следует выбрать заголовки полей новосозданной таблицы и указать имя связанной таблицы:

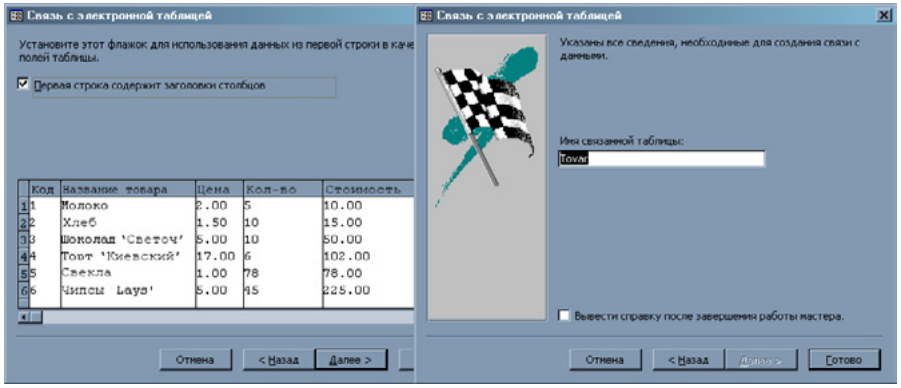


Рис. 49

Перейдем в окно нашей базы данных. Как видно из изображения, связанная таблица отображается с новой специальной пиктограммой со стрелкой, которая разме-

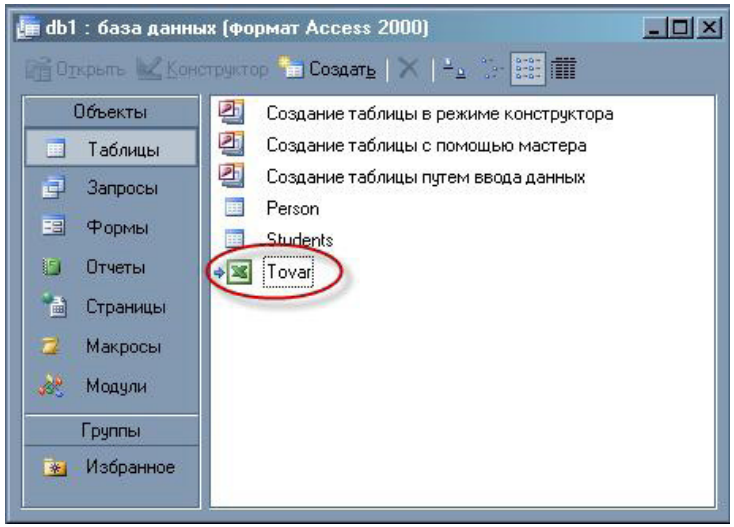


Рис. 50

щается слева от названия объекта базы данных. Именно эта стрелка на значке позволяет отличить связанную таблицу в базе данных MS Access от обычной статической таблицы. Кроме того, пиктограмма также указывает на тип связываемых данных (в нашем случае это иконка MS Excel) (Рис. 50).

Но это еще не все. Существует ряд особенностей работы со связанными данными. Например, структура внешней таблицы, определенной в другой программе, остается неизменной (невозможно изменить порядок, удалить или добавить новые поля). Однако, доступны для настройки свойства таблицы, которые определяют способ отображения данных на экране, поскольку на входную таблицу такие операции никак не влияют.

Подытоживая отметим, ***когда же именно отдать предпочтение связыванию***:

- Когда существует распределеная база данных, которая размещается на сервере и необходимо написать к ней интерфейс через MS Access. В таком случае Ваша база данных будет разделена на две части: серверную, где хранятся все таблицы, и клиентскую, где размещаются формы, отчеты и тому подобное. Копия клиентской БД предоставляется каждому пользователю и сохраняется в его локальной системе, а доступ обеспечивается благодаря связыванию.
- В случае, когда размер базы данных, хранящейся на сервере очень большой.

Экспорт базы данных MS Access

Кроме импорта данных из баз данных, хранимых в других форматах, MS Access поддерживает возможность экспорта данных в другую базу данных типа MS Access или в другие форматы. Но, следует помнить, что экспортировать можно только отдельные объекты, например, таблицы. Экспорт целой базы данных в MS Access не предусмотрен.

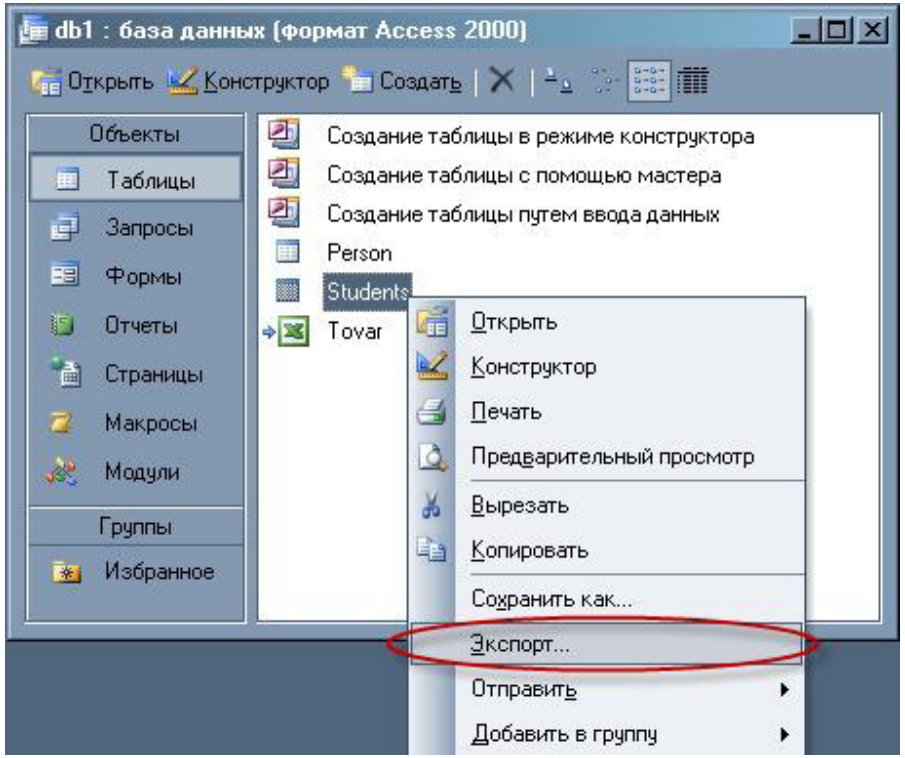


Рис. 50

Для того, чтобы осуществить экспорт данных, например, в формат электронных таблиц MS Excel достаточно выбрать в контекстном меню необходимой таблицы пункт "Экспорт" и указать требуемый формат для экспорта.

Домашнее задание

Разработать базу данных магазина, которая имеет следующую структуру:

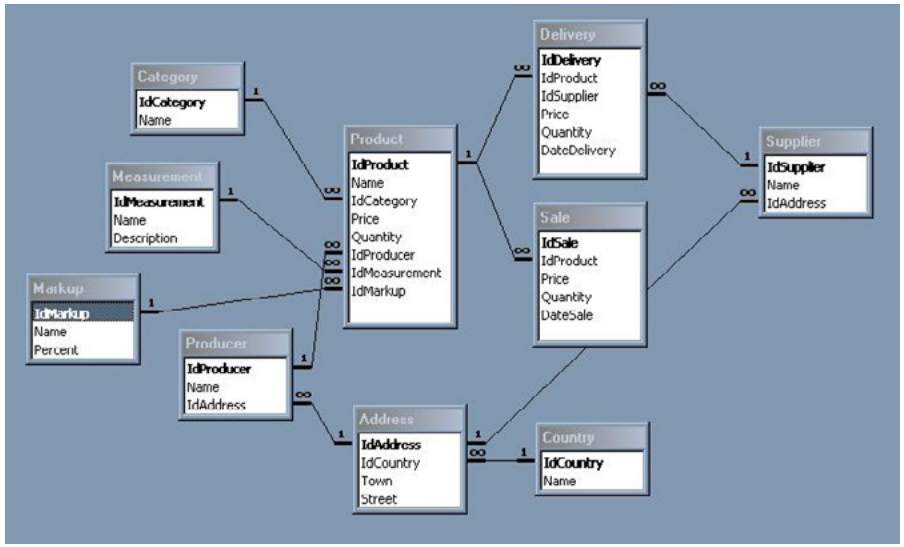


Рис. 51

Краткая характеристика таблиц:

- ***Product*** — содержит информацию о продукте;
- ***Deivery*** — поставка товаров;
- ***Supplier*** — информация о поставщике;
- ***Sale*** — продажа товаров;
- ***Address*** — полный адрес;
- ***Country*** — список стран;
- ***Producer*** — информация о производителях товаров;

- ***Markup*** — информация о скидках на продукцию: их название и процент. Например, сезонная скидка, акция и тому подобное;
- ***Measurement*** — единицы измерения: сокращенное название и описание;
- ***Category*** — категория товара.