ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
Пермский филиал

Кафедра информационных технологий в бизнесе

Разработка информационной системы предприятия

Пермь 2011

Разработка информационной системы предприятия.

Составители:
Викентьева Ольга Леонидовна
Лебедев Валерий Викторович

Учебно-методическое пособие составлено в соответствии с Государственным образовательным стандартом, учебной программой и концепцией дисциплины «Информационные технологии в экономике». Пособие предназначено для студентов и преподавателей ПФ ГУ ВШЭ и содержит серию практических занятий, раскрывающих возможности современных информационных технологий по созданию систем хранении, поиска и представления данных.

Рецензент: доцент кафедры информатики Пермского регионального института педагогических информационных технологий, кандидат педагогических наук, член-корреспондент Академии информатизации образования Кушев В.О.

Занятие 1. Проектирование реляционной базы данных

В обычном смысле БД представляет собой файл или множество файлов, имеющих определенную организацию. Однако при работе с обычными системами файловой обработки возникает ряд проблем, связанных, в частности, с избыточностью и зависимостью хранящихся в них данных. При проектировании БД эти проблемы решаются.

Пользователь должен принимать участие в процессе проектирования БД, так как только он может определить, какие данные необходимы для работы, указать связи, существующие между этими данными и обратить внимание на тонкости их обработки.

Информационные потребности отдельного пользователя обычно затрагивают лишь часть данных, хранящихся в информационной системе, и описание этих потребностей может не совпадать с описаниями потребностей других пользователей. Представление о том, какая именно информация необходима для работы, будет разным для разных групп пользователей, специалистов в различных областях – оно зависит от выполняемых ими обязанностей (специалист отдела кадров и сотрудники бухгалтерии, руководитель подразделения и т.д. нуждаются для выполнения своих функций в различных данных). Эти потребности и описываются на *внешнем уровне* представления данных (представления A, B, и C на рис.1).

Внешних описаний данных, хранящихся в БД, следовательно, может быть множество. Их необходимо свести в единое *концептуальное представление*, описывающее данные на уровне всей информационной системы в целом. Представление этих данных на внутреннем уровне определяется способом их хранения во внешней памяти.

Рассмотрим пример БД информационной системы фирмы, занимающейся поставками товаров в магазины города, причем будем учитывать только информационные потребности двух сотрудников фирмы (в упрощенной форме – иначе пример был бы слишком громоздким).

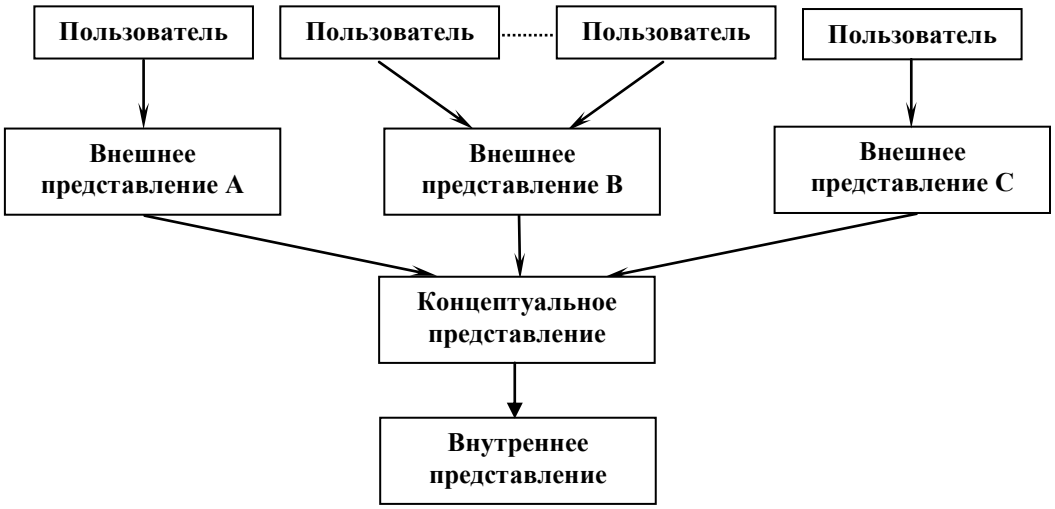


Рисунок 1. Формирование представления данных

Сотрудник, занимающийся связями с клиентами, для выполнения своих обязанностей нуждаются в информации, представленной на рис.2.

Для сотрудника, который работает с платежными формами, необходима другая информация о клиентах (рис.3).

Данные, используемые отдельными специалистами, находятся в единой информационной системе предприятия, в общей для них базе данных. Поэтому внешние представления отдельных пользователей должны быть интегрированы в концептуальном представлении, цель описания данных на концептуальном уровне – создание такого формального представления о данных, чтобы любое внешнее представление являлось его подмножеством. В процессе интеграции внешних представлений устраняются двусмысленности и

противоречия в информационных потребностях отдельных пользователей. Концептуальное описание, представляющее всю БД, должно быть единственным.

Название магазина	Владелец									Телефон	Адрес		
	Фамилия	Имя	Отчество	Дата рождения	Адрес			Паспорт			Улица	Дом	Офис
					Улица	Дом	Квартира	Серия	Номер				

Рисунок 2. Данные для первого сотрудника

Название магазина	ИНН	Телефон	Адрес			Банк	Номер счета	Код по ОКОНХ	Код по ОКПО
			Улица	Дом	Офис				

Рисунок 3. Данные для второго сотрудника

В рассматриваемом примере концептуальное представление должно включать всю информацию, необходимую всем сотрудникам. Противоречия могут возникнуть вследствие того, что сотрудники, которые используют общую информацию, могут представлять ее себе по-разному (например: номер телефона может быть записан в разных форматах). Все эти противоречия должны быть ликвидированы, данные и форма их представления должны быть согласованы.

Тогда концептуальное описание определяется следующей информацией

Название магазина	ИНН	Телефон	Адрес			Владелец								Банк	Номер счета	Код по ОКОНХ	Код по ОКПО
			Улица	Дом	Офис	Фамилия	Имя	Отчество	Дата рождения	Адрес			Паспорт				
										Улица	Дом	Квартира	Серия	Номер			

Данные, описанные концептуальной схемой, должны быть записаны во внешней памяти, на ВЗУ, предназначенных для хранения информации, находящейся в БД. Внутреннее описание данных характеризует способ хранения данных во внешней памяти.

Правила описания данных определяются выбранной *моделью данных* (в данном случае рассматривается только реляционная модель – самая распространенная на настоящее время).

Если взять описание данных о клиентах фирмы, занимающейся поставками товаров в магазины города, из приведенного выше примера, представленное на рис 4, то данные, описанные этой таблицей не могут в таком виде быть представлены в реляционной БД, так как не все значения являются атомарными (компоненты строк «Владелец» и «Адрес» состоят из нескольких значений, т.е. значения этих атрибутов заменяются другими отношениями, расшифровываются ими; в отношении, описывающем владельца, поля «Адрес» и «Паспорт» также не являются атомарными, следовательно строится иерархия отношений).

При проектировании БД могут быть приняты различные решения, но существуют базовые требования, которые должны учитываться в процессе работы: множество отношений должно обеспечивать минимальную избыточность представления информации; манипулирование данными, корректировка отношений не должна приводить к нарушению целостности данных, двусмысленности и потере информации; перестройка набора отношений при добавлении в БД новых атрибутов должна быть минимальной.

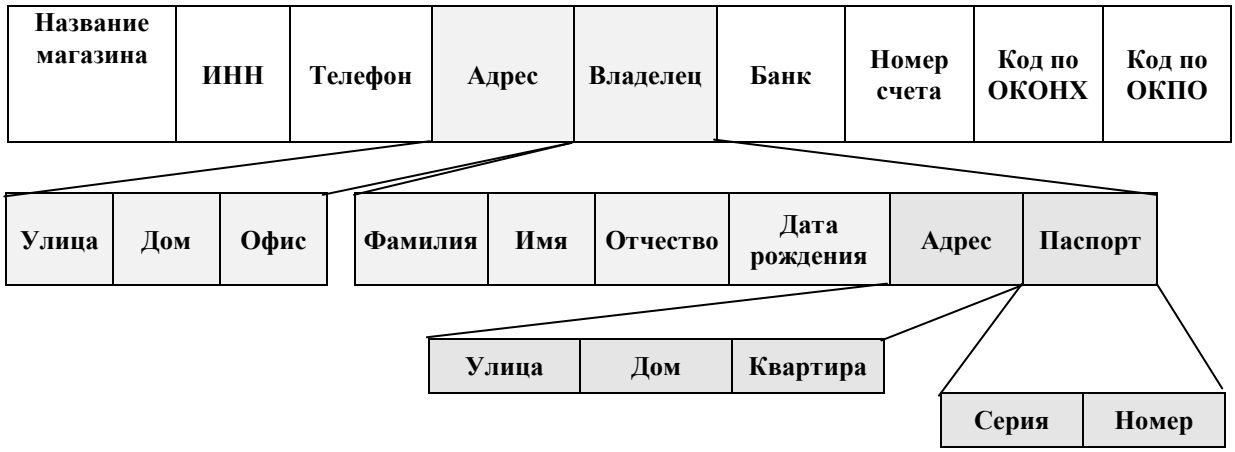


Рисунок 4. Общее представление данных

Описание реальных объектов и взаимосвязей между ними во многом носит субъективный характер, но есть определенные общие правила, в частности, *правила нормализации*. В ходе нормализации обеспечивается защита целостности данных путем устранения дублирования данных. В результате представление данных об одном объекте может быть разбито на несколько более мелких связанных таблиц (декомпозиция без потерь). Ограничения, которые должны соблюдаться при проектировании реляционной БД, достаточно многочисленны. Соблюдение ограничений при определении конкретных отношений в БД связано с реализацией *нормальных форм*. Нормальные формы нумеруются последовательно, начиная от первой. Чем больше номер нормальной формы, которой удовлетворяет БД, тем больше ограничений на хранимые в БД данные должно соблюдаться. Можно к типичным для реляционных СУБД ограничениям ввести дополнительный набор ограничений, что приведет к увеличению числа нормальных форм.

В плохо спроектированной БД вся информация может храниться в одной таблице. Для описанного выше примера такая таблица могла бы содержать следующие столбцы:

Название магазина	ИНН	Телефон	Улица магазина	Дом магазина	Офис	Фамилия	Имя	Отчество	Год рождения	Улица владельца	Дом владельца	Квартира	Серия	Номер	Банк	Номер счета	Код по ОКОНХ	Код по ОКПО

Компоненты адреса «Улица» и «Дом» переименованы для соблюдения требования того, что имена столбцов должны быть уникальны (правила именования зависят от конкретной СУБД).

Какие недостатки имеет такое представление?

Первая нормальная форма требует, чтобы на любом пересечении строки и столбца находилось единственное значение, которое должно быть неделимо (требование атомарности). Кроме того, в реляционной таблице не должно быть повторяющихся строк и групп данных.

Требование атомарности выполнено – составные столбцы «Адрес» и «Владелец» (а для владельца «Адрес» и «Паспорт») разбиты на компоненты, которые включены в общую таблицу. Но у одного магазина может быть несколько владельцев, а один человек может владеть несколькими магазинами. Это приводит к тому, что в таблицу нужно будет включать все строки, представляющие «комбинации» магазинов и их владельцев, т.е. в различных строках будут повторяться группы данных (несколько раз будут повторяться данные о магазине – для каждого его владельца, а данные владельца будут повторяться для каждого его магазина). Такое представление данных ведет к огромной избыточности, к тому, что неэффективно будет расходоваться память на ВЗУ. Кроме того, дублирование

информации может привести к проблемам при ее обработке: чтобы внести изменения в информацию о магазине (например, если у него изменится счет в банке) нужно изменить эти данные в нескольких записях, соответствующих разным владельцам.

При определении того, какие таблицы должны входить в БД, и того, какая информация в них должна храниться, следует учитывать следующее правило: *каждая таблица описывает объект*, существующий самостоятельно, обладающий собственными свойствами. Построение БД следует начать с создания представления каждого объекта в виде строк, содержащих его атрибуты, в соответствующей таблице; определения моделей взаимосвязи объектов. В рассматриваемом примере в БД фактически должна храниться информация об объектах двух типов: о магазинах и об их владельцах. Эту информацию следует поместить в две различные таблицы («Магазины» и «Владельцы»), имеющие следующие столбцы:

«Магазины»

Название магазина	**ИНН**	**Телефон**	**Улица**	**Дом**	**Офис**	**Банк**	**Номер счета**	**Код по ОКОНХ**	**Код по ОКПО**

«Владельцы»

Фамилия	**Имя**	**Отчество**	**Дата рождения**	**Улица**	**Дом**	**Квартира**	**Серия**	**Номер**

Каждая строка таблицы «Магазины» будет описывать экземпляр соответствующего объекта (один магазин). А в каждой строке таблицы «Владельцы» будет находиться информация об одном владельце магазина.

При работе с информацией, хранящейся в БД, СУБД должна уметь отличать строки друг от друга. Атрибут или набор атрибутов, однозначно определяющий строку, – это ее первичный ключ.

Что можно выбрать в качестве первичного ключа для описанных выше таблиц?

Ключом отношения является такое множество атрибутов, что каждое сочетание их значений встречается только в одной строке отношения и никакое подмножество этих атрибутов этим свойством не обладает. Таким образом, ключ однозначно определяет строку, позволяет выбрать ее из всего множества строк отношения.

Определим ключ для таблицы «Магазины».

Если выбрать в качестве ключа атрибут «Название магазина», будет ли он удовлетворять указанному требованию? Нет, если в одном городе может быть несколько магазинов с одинаковыми названиями, расположенных в разных частях города. Чтобы гарантировать однозначность следует дополнить название магазина его адресом (по названию магазина и его адресу можно однозначно выбрать нужную строку в таблице), тогда ключ отношения будет составным.

Простым ключом, идентифицирующим нужный магазин, может быть номер расчетного счета в банке (если у каждого магазина единственный номер расчетного счета и каждый расчетный счет принадлежит одному магазину). Ключом может быть также ИНН (идентификационный номер) магазина.

Выберем в качестве первичного ключа атрибут «ИНН». Далее этот атрибут будет использоваться для организации связей между таблицами «Магазины» и «Владельцы» (эти связи должны отражать реальные взаимосвязи между магазинами и их владельцами).

Определимся с ключами и для таблицы «Владельцы».

Если бы можно быль предположить, что среди владельцев магазинов нет однофамильцев, то в качестве ключа можно было бы выбрать атрибут «Фамилия» рассматриваемого отношения. Но, к сожалению, владельцы магазинов могут быть не только однофамильцами, но и полными тесками (маловероятно, но вполне возможно). Поэтому в качестве ключа можно выбрать паспортные данные владельца, т.е. использовать для его иден-

тификации составной ключ, включающий атрибуты «Серия» и «Номер». Этот ключ будем считать первичным. С его помощью установим связь между владельцем и его магазинами.

Первичные ключи обеспечат не только однозначность при поиске информации (они являются уникальными), но и позволят связать данные, находящиеся в двух таблицах.

Определим тип связи между таблицами «Магазины» и «Владельцы».

Если предположить, что один человек может владеть несколькими магазинами, но у каждого магазина есть единственный владелец, то следовало бы установить между этими таблицами связь «один-ко-многим». Для организации такой связи в БД можно было бы в строку таблицы «Магазины», содержащую информацию о магазине, включить *внешний ключ*, идентифицирующий владельца магазина, т.е. данные его паспорта – атрибуты «Серия» и «Номер». Организовать связь, включив ключ «ИНН», идентифицирующий магазины, в качестве внешнего ключа в таблицу «Владельцы», в данном случае нельзя, так как в этом случае информацию о владельце пришлось бы дублировать для каждого магазина.

Если сделать предположение о том, что один человек может быть владельцем только одного магазина, но у каждого магазина может быть несколько владельцев, получится связь «один-ко-многим», но в данном случае *внешний ключ* (ИНН магазина) пришлось бы включать в таблицу, содержащую сведения о владельцах.

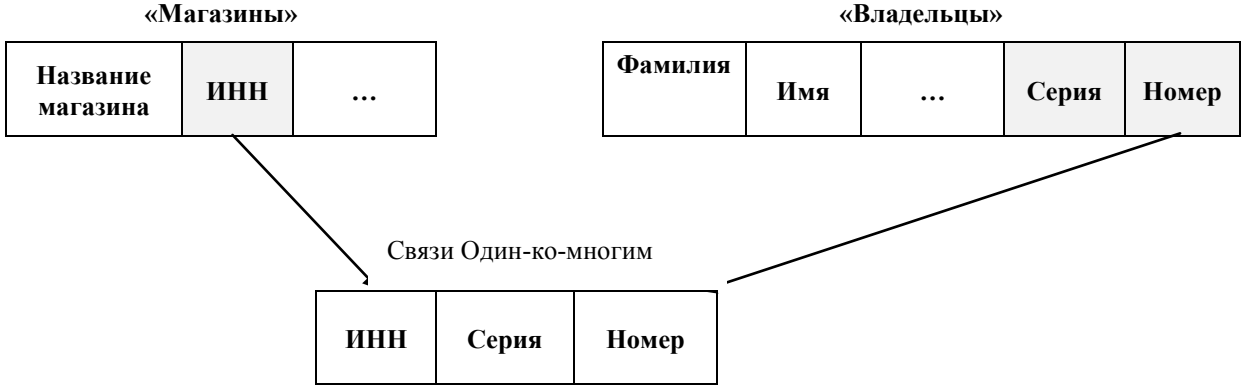
В действительности каждый человек может оказаться владельцем нескольких магазинов и у каждого магазина может быть несколько владельцев, поэтому между таблицами «Магазины» и «Владельцы» должна быть установлена связь «многие-ко-многим», для организации которой создается специальная таблица, описывающая связи между магазинами и владельцами:

«Магазины-Владельцы»

ИНН	Серия	Номер

Эта таблица позволит по атрибуту «ИНН» магазина найти всех его владельцев (через данные их паспортов), а по составному атрибуту, включающему атрибуты «Серия» и «Номер» паспорта владельца найти в БД все магазины, которыми он владеет.

Для этого следует, создав таблицу «Магазины-Владельцы», установить связи «один-ко-многим» между таблицей «Магазины» и таблицей «Магазины-Владельцы», а также между таблицами «Владельцы» и «Магазины-Владельцы»:



«Магазины-Владельцы»

Установленные связи помогают СУБД поддерживать целостность, согласованность информации. Например, можно задать правила обновления информации в связанных таблицах при обновлении информации в основной таблице (при ликвидации магазина, например, должна быть удалена и перенесена в архив информация о нем из БД, причем не только строка из таблицы «Магазины», но и вся информация в связанных с ней таблицах, относящаяся к этому магазину).

Для удобства пользователей, ускорения поиска СУБД поддерживают возможность поиска не только по уникальным ключам. Например, найти в таблице можно все магазины с одинаковыми названиями или все магазины, принадлежащие одному владельцу.

Нормализация данных привела к разделению таблиц, выделению отношений «первичный ключ–внешний ключ» в меньшие таблицы. Результатом нормализации является уменьшение избыточности данных – уже не нужно дублировать данные о каждом владельце для каждого магазина.

Вторая нормальная форма требует, чтобы любой неключевой столбец зависел от своего первичного ключа (причем от всего ключа, а не от отдельных его компонентов). Отношение имеет вторую нормальную форму, если оно соответствует первой нормальной форме и не содержит неполных функциональных зависимостей. Неполная функциональная зависимость определяется двумя условиями: ключ отношения функционально определяет некоторый неключевой атрибут и часть ключа функционально определяет тот же неключевой атрибут.

Отношение, не соответствующее второй нормальной форме, характеризуется избыточностью хранимых данных.

В рассматриваемом примере набор атрибутов отношений и выбор ключей сделан таким образом, что таблицы соответствуют второй нормальной форме. Если бы этого соответствия не было, для приведения таблиц ко второй нормальной форме было бы необходимо выделить повторяющуюся информацию (часть ключа и определяемые ею неключевые атрибуты) в отдельную таблицу.

Например: в БД необходимо хранить информацию о товарах, которые поставлены в магазины. Эта информация включает атрибуты «Наименование», «Код» и «Цена» товара, а также «Количество» поставленного товара. Если включить эту информацию в таблицу «Поставки» в следующем представлении:

«Поставки»

ИНН	**Код**	**Наименование**	**Цена**	**Количество**

(здесь «ИНН» идентифицирует магазин, в который выполнена поставка (это внешний ключ, используемый для создания связи «один-ко-многим» таблицы «Магазины» с данной таблицей), «Наименование» – название товара, «Код» – его уникальный код (товары с разными характеристиками и, следовательно, разными ценами могут иметь одно наименование, но коды будут разными), «Цена» – отпускная цена товара, «Количество» – количество поставленного товара), то может возникнуть избыточность.

Для определения строки, представляющей поставку товара в конкретный магазин, можно задать составной ключ, включающий атрибуты «ИНН» и «Код». Эта информация дает возможность определить цену товара и его количество, поставленное в данный магазин, а также вычислить общую стоимость товара. Если предположить, что товар поставляется во все магазины по одной и той же цене, и цена не изменяется со временем, то неключевой атрибут «Цена» определяется не только составным ключом «ИНН» + «Код», но и его частью – атрибутом «Код». Таким образом, одна и та же цена повторяется во всех строках таблицы, где содержится информация о поставке одного и того же товара. Это ведет к избыточности. Наименование товара также определяется его кодом. Поэтому информацию, относящуюся только к товару и не зависящую от магазина, можно вынести в отдельную таблицу:

«Товар»

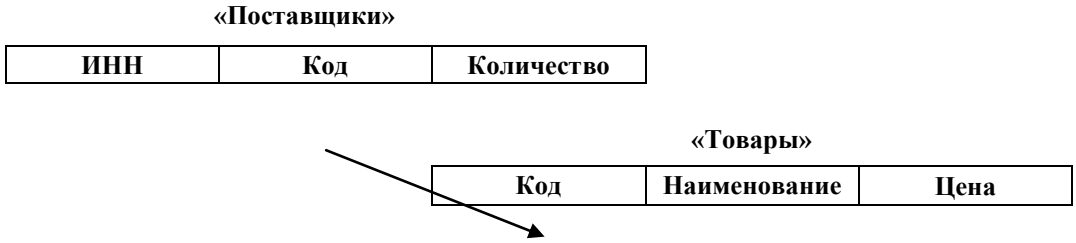
Код	**Наименование**	**Цена**

а в таблице «Поставки» оставить только описанную ниже информацию:

«Поставки»

ИНН	**Код**	**Количество**

Здесь ключевое поле «Код» позволит связать данные, находящиеся в таблице «Поставки», с данными из таблицы «Товары»



Таким образом, приведение ко второй нормальной форме ликвидировало избыточность путем выделения новых таблиц: таблица «Поставки» разбита на две таблицы «Поставки» и «Товары», между которыми установлена связь.

Третья нормальная форма еще больше повышает требования: отношение соответствует второй нормальной форме и среди его атрибутов отсутствуют транзитивные функциональные зависимости (ни один неключевой столбец не должен зависеть от другого неключевого столбца, он может зависеть только от первичного ключа).

В рассматриваемом примере несоответствие третьей нормальной форме проявилось бы при выполнении такого условия: все товары с одинаковым наименованием имеют одну цену (наименование определялось бы кодом, а цена – наименованием товара). В этом случае появилась бы избыточность, так как цена для данного наименования товара повторялась бы столько раз, сколько различных кодов этого товара используется.

Избавиться от избыточности можно было бы, разбив таблицу «Товары» на две таблицы (одна включала бы атрибуты «Код» и «Наименование», а вторая «Наименование» и «Цена»).

Однако в рассматриваемом примере ситуация другая: товары имеют разные коды, если их характеристики различны, следовательно, должны отличаться и цены.

Четвертая нормальная форма запрещает независимые отношения типа «один ко многим» между ключевыми и неключевыми столбцами. Проще говоря, в одну таблицу нельзя помещать разнородную информацию, т.е. данные, между которыми нет непосредственной связи.

Это правило можно рассмотреть на следующем примере. Сотрудник, занимающийся связями с клиентами, для выполнения своих обязанностей собирается использовать информацию о членах семей владельцев магазинов. Эту информацию не следует включать в таблицу «Владельцы», так как трудно определить, сколько места нужно резервировать в строках таблицы, соответствующих конкретным людям, для хранения данных о их семейном положении, – один может быть одиноким, а другой – многодетным отцом. Информацию о членах семьи нужно вынести в отдельную таблицу, каждая строка которой будет содержать информацию об одном члене семьи, включив в нее внешний ключ, идентифицирующий владельца магазина, для организации связи с таблицей «Владельцы».

Пятая нормальная форма обычно завершает процесс нормализации. На этом этапе все таблицы разбиваются на минимальные части для устранения в них избыточности. Каждый фрагмент неключевых данных в таблицах должен встречаться только один раз. Это снимает проблемы с обновлением информации в БД: все изменения неключевой информации должны вноситься только один раз, что обеспечивает возможность управления целостностью данных.

Процесс проектирования БД является очень важным этапом в разработке информационных систем. Именно качество проектирования во многом определяет эффективность использования БД.

В настоящее время широко используются специальные средства, облегчающие процесс разработки информационных систем (CASE-средства – Computer-Aided Software/System Engineering).

Вопросы для самоконтроля:

1. Что представляет собой база данных?
2. Что такое внешнее представление данных?
3. В чем сущность концептуального представления данных?
4. Что такое модель данных?
5. Что такое нормализация?
6. Что такое ключ отношения?
7. Какой ключ называется внешним?
8. Какие связи могут быть организованы в базе данных?
9. В чем сущность каждой из пяти нормальных форм?

Задание для самостоятельной работы:

Спроектировать базы данных некоторой фирмы, занимающейся обслуживанием клиентов. База данных нужна трем сотрудникам фирмы. Первый из них занимается учетом услуг, оказываемых фирмой, и нуждается в следующей информации:

Услуги			**ФИО исполнителя**	**Оплачено**
Код услуги	**Наименование**	**Стоимость**		
1	Верстка	2		
2	Набор	5		
3	Печать	1		
4	Сканирование	1		
5	Копирование	2		

Второй сотрудник собирает сведения об исполнителях и его интересует:

Номер исполнителя	**ФИО**	**Услуги**			
		Наименование	**Стоимость**	**Дата приема**	**Дата исполнения**
1	К.К.Кукушкин				
2	И.О.Гришин				
3	П.П.Петров				
4	С.С.Сидоров				
5	А.А.Андреев				
6	А.А.Александров				
7	М.К.Сергеев				
8	А.П.Аркадьев				
9	А.А.Антонов				

Третий сотрудник работает с клиентами и ему важно знать:

Номер клиента	**ФИО**	**Адрес**	**Заказы**				
			Наименование	**Дата приема**	**Дата исполнения**	**Стоимость**	**Оплачено**
1	И.И.Иванов	Пермь, Ленина, 12-40					
2	С.С.Сидоров	Пермь, Сибирская, 11-36					
3	И.И.Ильин	Пермь, Пушкина, 22-65					
4	О.О.Овсов	Пермь, Ким,16					
5	К.П.Шишкин	Пермь,Хасана, 4-59					

Занятие 2. Создание структуры базы данных с помощью СУБД

Созданию структуры базы данных в СУБД предшествует этап проектирования базы данных, который может быть выполнен вручную, без применения компьютера. Однако для оформления результатов проектирования целесообразно использовать универсальный текстовый процессор типа Word, в частности его средства работы с таблицами. Результатом проектирования является система нормализованных таблиц, связанных между собой различными отношениями. Рассмотрим процесс создания базы данных, представленной на рис. 5.

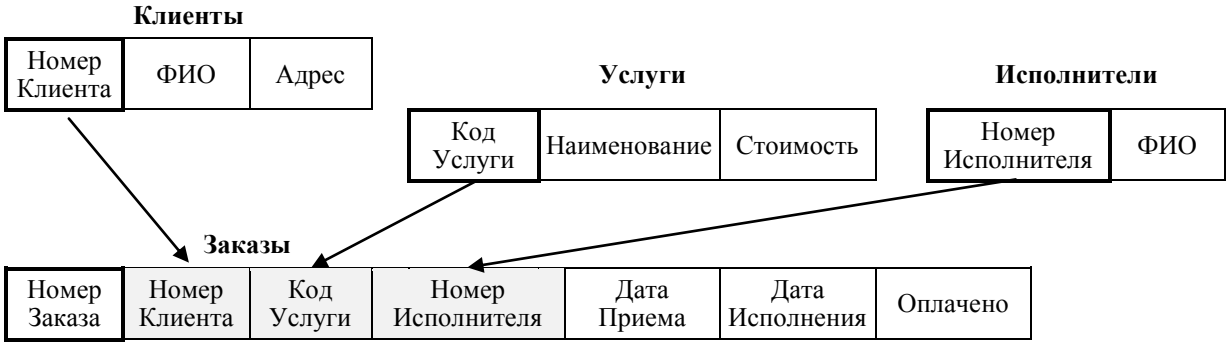


Рисунок 5. Структура базы данных Фирма

Таблица **Услуги** содержит перечень услуг, оказываемых клиентам, и информацию о стоимости каждой услуги. Таблица **Исполнители** содержит список исполнителей, которым передаются заказы клиентов. Таблица **Клиенты** содержит информацию о клиентах, каждый из которых может оформить заказы на оказание различных услуг. Таблица **Заказы** содержит данные обо всех заказах, сделанных клиентами. Стрелками показаны связи, существующие между таблицами. В данном примере тип всех связей – ***один-ко-многим***, так как каждый клиент может оформить несколько заказов на различные услуги, одна и та же услуги может быть заказана несколькими клиентами и один исполнитель может выполнять несколько заказов. Первичные ключи для каждой таблицы выделены черной рамкой. В таблице **Заказы** серую заливку имеют поля, содержащие внешние ключи, включенные в эту таблицу для создания связей.

СУБД Access 2007содержит следующие основные объекты для создания БД и работы с ней: ***Таблицы, Формы, Запросы*** *и* ***Отчеты***. ***Таблицы*** содержат данные. ***Запросы*** предназначены для получения нужных данных из БД. ***Формы*** и ***отчеты*** облегчают редактирование, графическое представление и печать данных. Для создания этих объектов СУБД предоставляет в распоряжение пользователя набор специальных средств: ***Конструктор***, ***Мастер*** и т.д.

2.1. Создание новой базы данных

Создание базы данных начинается с запуска СУБД. После запуска Access 2007 на экране появляется окно программы Приступая к работе с Microsoft Office Access, содержащее набор шаблонов (рис. 6). Для создания новой базы можно воспользоваться командой Новая база данных. Создаваемая база данных должна быть обязательно сохранена в папке пользователя под соответствующим именем.

Задание 1. Создайте новую базу данных **Фирма**, выполнив следующие операции:

1. Запустите СУБД Access любым способом и выберите команду **Новая база данных.**
2. Откроется правая боковая панель Новая база данных. С помощью этой панели выберите папку, в которой будет храниться новая БД (любая папка в вашей папке Мои доку-

менты), затем в поле **Имя файла** запишите нужное имя (**Фирма**) новой базы, сохранив в имени расширение **accdb**. Нажмите кнопку **Создать**.

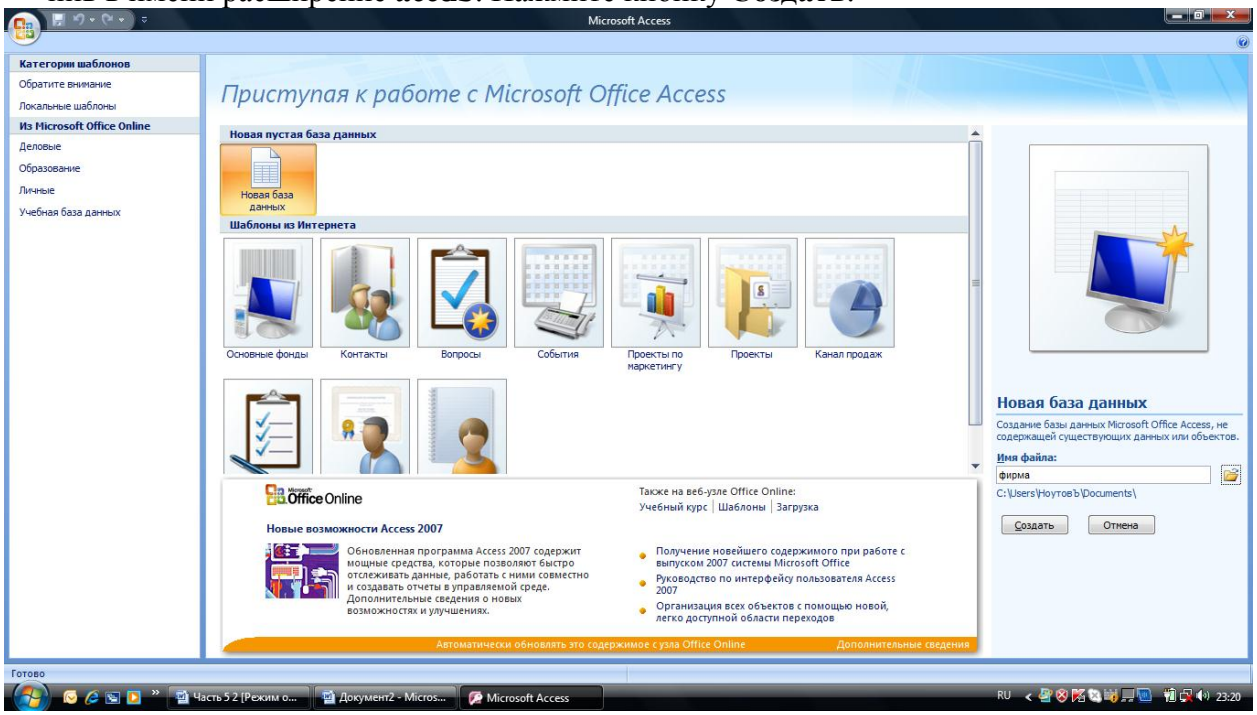


Рисунок 6. Окно программы Microsoft Access с диалоговым окном для создания новой БД

3. Откроется новой базы **Фирма**, предоставляя в распоряжение пользователя набор средств для создания первой таблицы. При этом будет раскрыта контекстная инструментальная лента Рабата с таблицами, Режим таблицы (рис.7). Созданная база данных пока является пустой и не содержит никаких объектов.

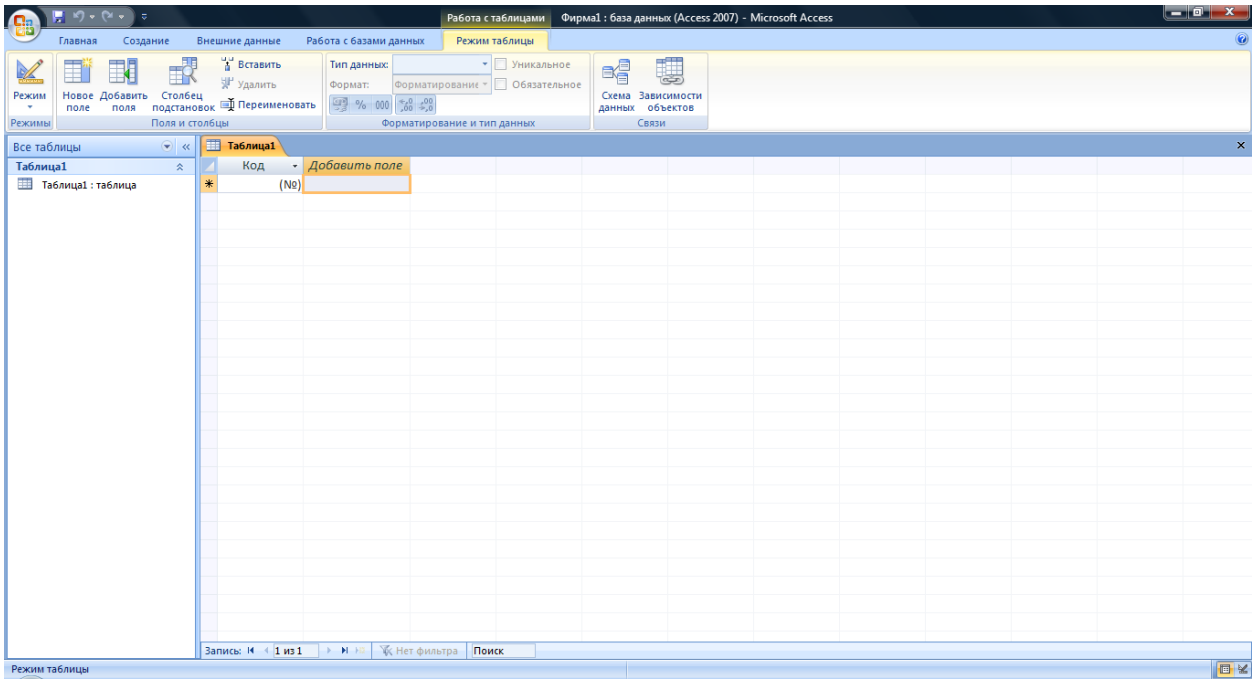


Рисунок 7. Окно новой базы данных Фирма

Используя предлагаемую методику, можно создать любое количество новых баз данных. Эффективным способом создания базы данных является использование шаблонов. Если использовать шаблоны, предоставляемые программой, то СУБД позволяет сформировать структуру новых объектов автоматически. Шаблон **Новая база данных** позволяет

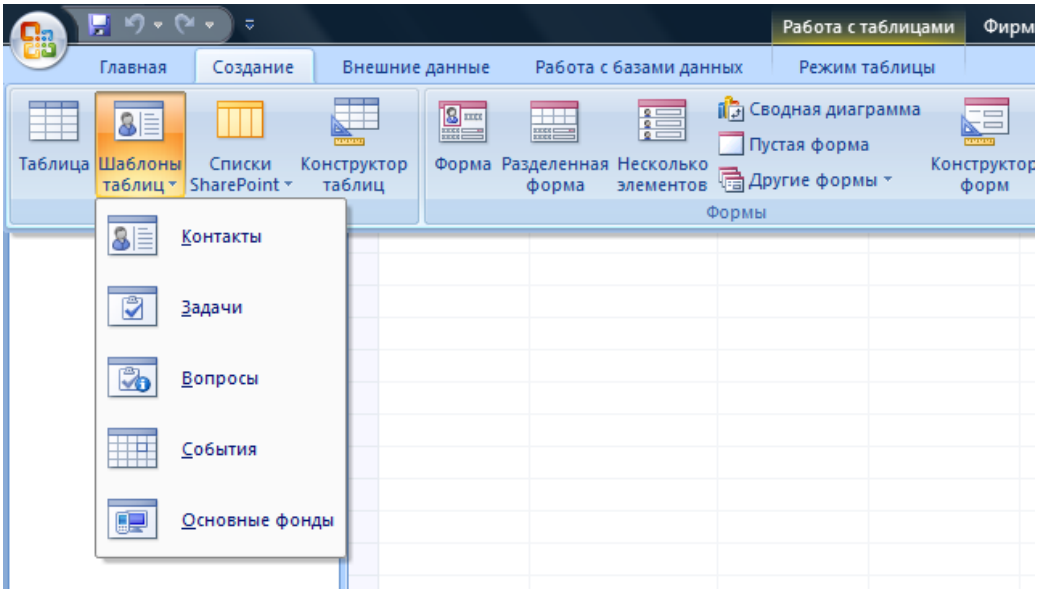
только открыть базу, а формирование структуры новых объектов пользователь выполняет сам.

Вопросы для самоконтроля:

1. Основные объекты СУБД Access и их назначение.
2. Как создать новую БД на основе шаблона БД? Какие шаблоны баз данных имеются на вашем компьютере?
3. Как создать новую пустую базу данных?

2.2. Создание новых таблиц в базе данных

Для создания новой таблицы можно воспользоваться контекстной вкладкой **Режим таблицы**. Другие средства для создания таблиц размещены на вкладке **Создание**. Новую таблицу можно построить на основе шаблонов таблиц. Если вы создаете новый список на узле SharePoint, то вместе с ним создается и связанная таблица в текущей базе данных. Удобным средством для создания таблиц является **Конструктор таблиц**.



Для создания новой таблицы в режиме конструктора необходимо на вкладке Создание выбрать команду Конструктор таблиц. Откроется окно конструктора **Таблица 1. (**рис. 8). Конструктор позволяет создать структуру новой таблицы («шапку» таблицы), то есть определить, сколько столбцов будет иметь таблица, как называется каждый столбец, какие данные планируется записывать в каждый столбец и т.д. Окно конструктора содержит три столбца: **Имя поля**, **Тип данных** и **Описание**. В каждой строке вводится информация, описывающая одно поле (столбец) создаваемой таблицы.

Исследуйте содержимое этого окна, последовательно вводя в таблицу данные, представленные на рис. 8. **Имя поля** может содержать не больше 64 символов и состоять из букв, цифр, пробелов и знаков пунктуации. Для ввода имени поля таблицы нужно установить курсор в соответствующую строку столбца **Имя поля** конструктора. Каждый атрибут представляется в строке таблицы БД значением определенного *типа*. По умолчанию в Access полю присваивается тип **Текстовый**. Пользователь имеет возможность задать свой тип для определяемого поля: при перемещении курсора в столбец **Тип данных** в соответствующей ячейке появляется кнопка раскрывающегося списка, которая позволяет раскрыть список всех используемых в Access типов данных.

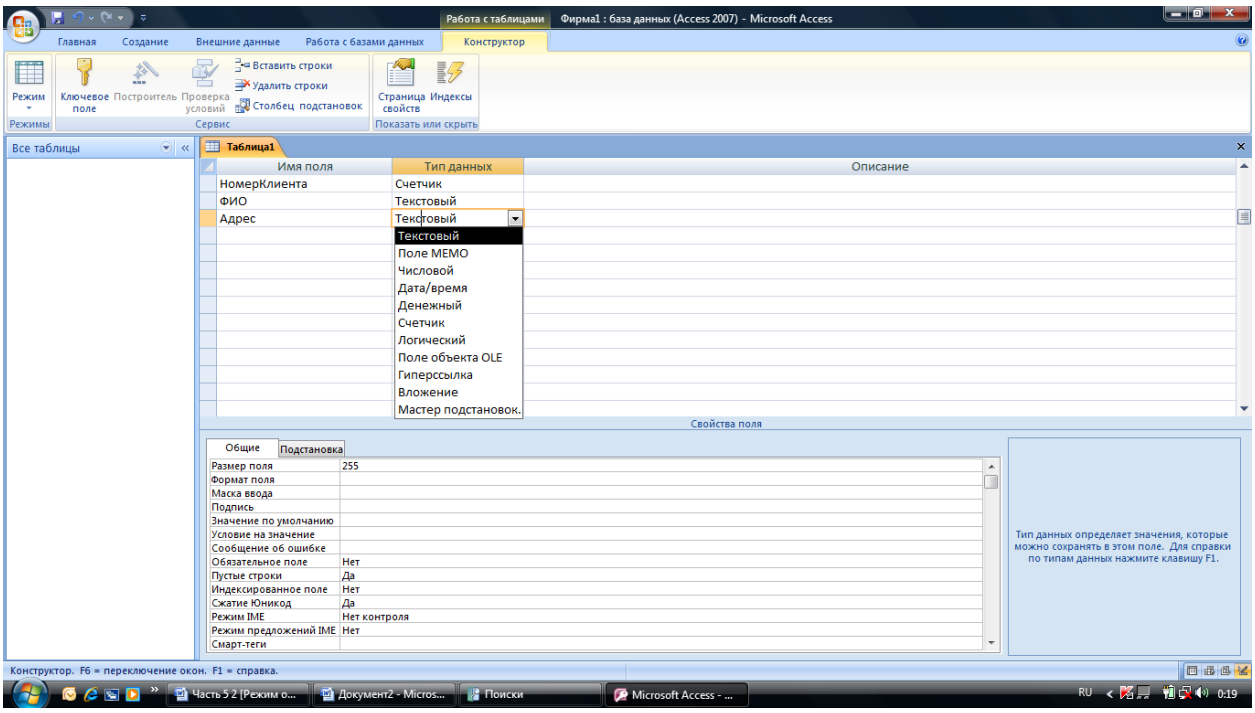


Рисунок 8. Диалоговое окно конструктора таблиц

В этом списке можно выбрать нужный тип, соответствующий назначению атрибута:

- **Текстовый тип** используется для хранения любой последовательности символов. Текстовые поля могут содержать до 255 символов (по умолчанию длина равна 50 символам).
- Тип **Числовой** используется для представления числовых данных (кроме денежных сумм).
- Тип **Дата/время** предназначен для хранения даты и/или времени.
- Тип **Счетчик** используется для автоматической нумерации добавляемой записи. В первой записи таблицы этому полю автоматически присваивается значение 1. Значение этого атрибута в каждой следующей записи увеличивается на 1. Можно также задать случайный выбор значений.
- **Денежный тип** используется для хранения числовых значений денежных сумм. Использование этого типа позволяет избежать ошибок округления.
- **Логический тип** применяется для хранения логических величин, принимающих только два значения типа **Да** (некоторое условие выполнено) и **Нет** (соответствующее условие не выполнено).
- **Поле MEMO** используется для хранения текста большого объема (свыше 65 тыс. символов).

Кроме того, используются **Поля объекта OLE**, содержащие объект OLE (такой объект может содержать документ другого приложения Windows: текстовый документ или таблицу, аудио- или видеозапись, рисунок и т.п.), поля типа "**Гиперссылка**", содержащие буквенно-цифровой идентификатор – *адрес гиперссылки*, указывающий путь к другому объекту, документу или Web-вкладке. **Мастер подстановок** используется для задания набора значений, которые может принимать это поле (например, если значения поля представляют коды, заданные в некотором словаре (например, районов города), то можно указать, что поле должно содержать только коды, перечисленные в этой таблице-словаре).

В нижней части окна после определения типа поля на специальных вкладках (раздел **Свойства Поля**) можно задать параметры, устанавливаемые для значения каждого типа данных. Например, для текстовых данных определяется их размер, формат и маска ввода, значение, устанавливаемое по умолчанию, обязательно ли вводить значение в это поле, допустима ли в качестве значения пустая строка и т.д. Свойство поля может быть введено с клавиатуры в соответствующем поле ввода как символьная строка, выбрано из списка

(тогда при выборе этого свойства справа от поля ввода появляется кнопка раскрытия списка) или сформировано в специальном диалоговом окне, раскрываемом кнопкой, содержащей обозначение ..., появляющейся после выбора этого поля справа от строки ввода.

Для многих типов (например, **Значение по умолчанию** или **Условие на значение**) можно указать специальные условия, которым должно удовлетворять данное поле. Эти условие определяется в окне **Построителя выражений**, открываемом щелчком по кнопке ..., расположенной справа от поля ввода этого параметра (рис. 9). Сообщение об ошибке, может быть выведено при неправильном вводе, не удовлетворяющем заданным условиям. Таблицу можно проиндексировать по значениям, расположенным в некоторых столбцах (обычно – по ключевым значениям), что ускоряет поиск и сортировку в ней данных.

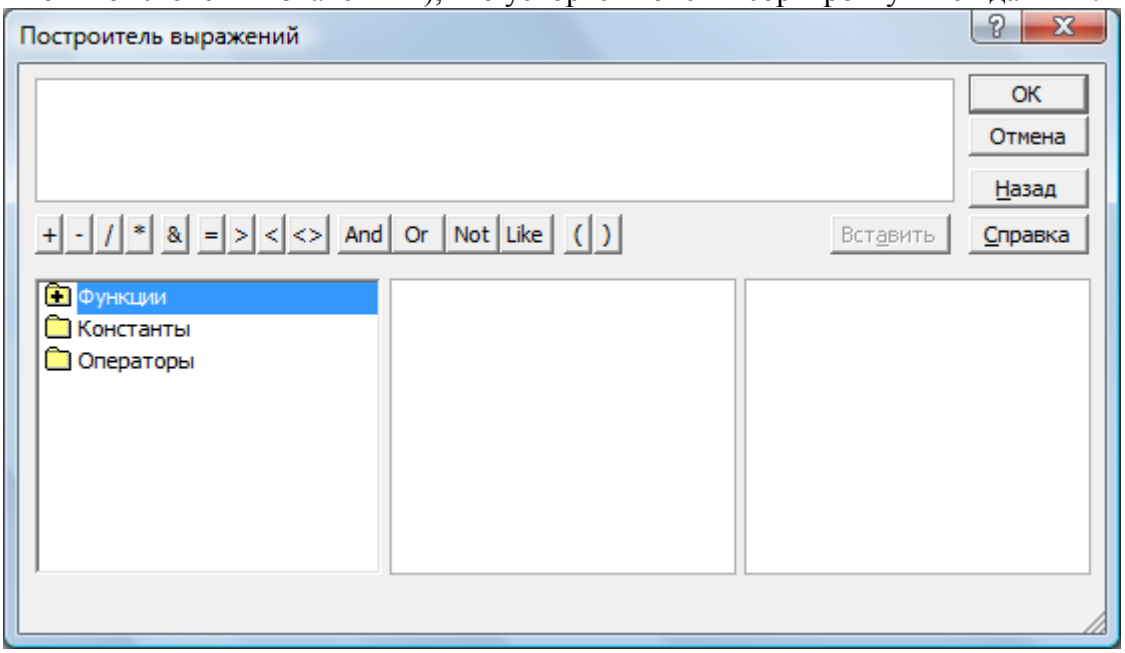


Рисунок 9. Диалоговое окно Построитель выражений

Формат поля и **Условие на значение** – это наиболее мощные средства. **Формат поля** определяет вид данных в поле таблицы. При описании формата используются специальные символы (как и в Excel).

Если поле ввода содержит символьные константы (т.е. в одних и тех же позициях вводимой строки должны постоянно появляться одни и те же символы) и позиции для заполнения, то можно задать свойство **Маска ввода.** Это свойство обеспечивает соответствие данных определенному формату, а также заданному типу значений, вводимых в каждую позицию. При вводе данных символы шаблона, выбранные в маске и размещенные в нужных позициях, заменяются вводимыми символами.

Если для поля определены как формат отображения, так и маска ввода, то при добавлении и редактировании данных используется маска ввода, а параметр **Формат поля** определяет отображение данных при сохранении записи. Если используются оба свойства, результаты их действий не должны противоречить друг другу.

Выражения, определяющие условия и записываемые с помощью **Построителя выражений** (рис. 9), могут включать символы математических операций, операций сравнения, вызовы функций, скобки, а в качестве операндов могут использоваться константы и имена полей, заключенные в квадратные скобки. Для ввода выражения можно использовать имеющиеся в окне кнопки, элементы списков (значки списков похожи на значки папок, их можно раскрыть двойным щелчком по значку). Более подробную информацию по этому вопросу можно найти в справочной системе или специальной литературе [1].

Заполнение раздела **Описание** (комментария) окна конструктора является необязательным.

Для ***определения и изменения структуры*** разрабатываемой таблицы достаточно щелкнуть внутри соответствующего поля и внести необходимые изменения.

Для вставки нового поля следует поместить указатель в то место, куда должно быть вставлено новое поле и выполнить команду вставки строки (Вкладка Конструктор, группа Сервис, команда Вставить строки). Для удаления поля его нужно выделить щелчком мыши на кнопке слева от имени поля и выполнить команду **Удалить строки.**

Для ***определения ключевых полей*** следует выделить их (если ключ является составным, выбирается сначала первое поле щелчком на кнопке, расположенной слева от названия поля, а затем при нажатой клавише ***Ctrl*** выделяются остальные поля) и выбрать команду **Ключевое поле**. Рядом с выбранными полями появится пиктограмма ключа. Выбор поля в качестве ключевого можно отменить теми же средствами.

Если ключевые поля не заданы, то при сохранении таблицы Access предложит их создать автоматически. Ключевые поля размещаются в таблице первыми.

Для ***сохранения созданной таблицы*** можно закрыть окно с помощью кнопки на его заголовке. Появится запрос о том, нужно ли сохранять внесенные в таблицу изменения. В открывшемся диалоговом окне нужно ввести имя таблицы. Если при сохранении таблицы Access обнаружит ошибки, на экране появится соответствующее сообщение и операция не будет выполнена. После исправления обнаруженных ошибок операцию можно повторить.

Значок созданной таблицы появляется в окне создаваемой БД.

Задание 2. Создайте первую таблицу **Клиенты**, выполнив следующие операции:

1. Установите курсор в столбец **Имя поля** первой строки **окна Конструктора таблиц**. Введите имя столбца создаваемой таблицы **НомерКлиента.** Обратите внимание на то, что имена полей записываются без пробелов как одно слово.
2. Переведите курсор в поле **Тип данных** той же строки. Из списка возможных типов выберите тип **Счетчик**.
3. Щелкните по кнопке **Ключевое поле** на вкладке **Конструктор** в группе **Сервис**, указывая, что данное поле будет первичным ключом таблицы (слева от имени поля появится значок ключа). По ключевому полю автоматически будет выполняться индексация. Все значения счетчика будут автоматически определяться Access и будут уникальными для таблицы. В строке **Индексированное поле** должно быть установлено **Да(Совпадения не допускаются).**
4. Повторите шаги 1 и 2 для второй строки, указав имя поля **ФИО** и выбрав для него текстовый тип. Укажите, что максимальная длина поля должна быть равна 30, установив курсор в строчку **Размер поля** на вкладке **Общие** в разделе **Свойства поля** окна конструктора. Для данного поля можно указать **маску ввода**, которая позволит контролировать правильность ввода данных. Введенная с клавиатуры маска **>L.>L.>L<??????????????????????????** позволит вводить пользователю только буквы (точки после инициалов будут расставляться автоматически), причем будет автоматически выполняться перевод инициалов и первой буквы фамилии на верхний регистр (буквы будут заглавными). Введенные в соответствии с этой маской данные будут отображаться в таблице в следующем виде: после ввода в поле **ФИО** БД строки **пппетров** запомнится значение **П.П.Петров**. Укажите, что ввод данного поля обязателен (в строке **Обязательное поле** нужно установить значение **Да**), пустая строка в нем не допускается (в строке **Пустые строки** задается значение **Нет**).
5. Повторите описанные выше шаги для следующей строки, в которой должен быть записан адрес (маску задавать не нужно, индексировать по этому полю также не нужно).
6. Закройте окно **Конструктора** с помощью кнопки **Закрыть**, сохранив внесенные изменения, при закрытии задайте имя таблицы - **Клиенты**.

Задание 3. С помощью конструктора таблиц самостоятельно разработайте структуру таблицы **Услуги.** Названия полей этой таблицы приведены на рис.5. Используйте следующие типы полей: **КодУслуги** – **Счетчик** (ключевое поле); **Наименование** – текстовое поле (длина до 20, обязательное поле, пустые строки не допускаются); **Стоимость** – числовое поле, целое, положительное, обязательное (**Условие на значение** - >0), сообщение об ошибке – **Допустимы только целые положительные значения**.

Задание 4. Разработайте структуру таблицы **Исполнители,** используя для ее полей следующие параметры: **НомерИсполнителя –** счетчик, ключевое поле, совпадения не допускаются; **ФИО** задается аналогично соответствующему полю в таблице **Клиенты**.

Задание 5. Разработать структуру таблицы **Заказы,** определив для ее полей следующие параметры:

- Поле **НомерЗаказа** имеет тип **Счетчик** и является ключевым, совпадения не допускаются.
- Поля **НомерКлиента**, **КодУслуги** и **НомерИсполнителя** являются внешними ключами. Они соответствуют ключевым полям из связанных таблиц, поэтому типы этих полей должны соответствовать типам полей в основных таблицах (счетчикам) и могут иметь только числовой тип (размер **Длинное целое**). Так как связи между таблицами имеют тип **Один-ко-многим**», необходимо выполнить индексацию по этим полям и установить для индексированного поля значение **Да(Допускаются совпадения).**
- Поля **ДатаПриема** и **ДатаИсполнения** являются обязательными, они имеют тип **Дата/время** и для них определен формат поля - **Краткий формат даты).**
- Обязательное поле **Оплачено** имеет **логический** тип. Значение, принимаемое по умолчанию, для этого поля не задается.

Используя предлагаемую методику, пользователь может создать любое количество таблиц и определить свойства каждого поля каждой таблицы. На этом этапе проектирования базы данных создаются пустые таблицы. Для их заполнения лучше использовать другие средства. Перед заполнением таблиц необходимо установить связи между таблицами.

Вопросы для самоконтроля:

4. Какие режимы создания таблиц используются в СУБД?
5. Назначение **Конструктора** таблиц.
6. Какие типы данных используются в БД? Как выбрать тип данных?
7. Что такое свойства поля? Как их задать?
8. Что такое **Условие на значение**? С помощью какого диалогового окна оно задается?
9. Как вызвать **Построитель выражений**? Как им пользоваться?
10. Что такое **Маска ввода**? Какие возможности она предоставляет пользователю и как ее задать?
11. Что такое ключ? Как определить ключевое поле (составное ключевое поле). Как создать новую пустую базу данных?

Задание 6. Самостоятельно сформулируйте и запишите вывода по лабораторной работе 1.

Занятие 3. . Установка связей между таблицами и ввод данных в таблицы

3.1. Установка связей между таблицами

Одно из наиболее мощных средств Access – возможность ***устанавливать связи между таблицами***. Создание связей между таблицами облегчает выполнение операций над данными, позволяет контролировать их целостность. Для реализации этой операции используется команда **Схема данных** на вкладке **Работа с базами данных** в группе **Показать или скрыть**. При первом выполнении этой команды появляется диалоговое окно **Добавление таблицы** (рис. 10).

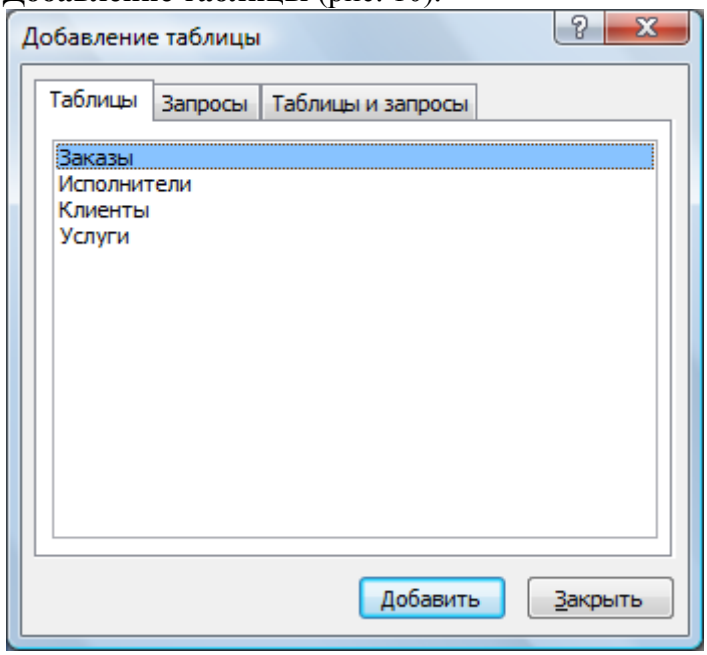


Рисунок 10. Диалоговое окно Добавление таблицы

Это окно позволяет выбрать таблицы, между которыми устанавливается связь.

Задание 6. Установить связи между созданными таблицами (**Заказы, Исполнители, Клиенты, Услуги**) в соответствии со схемой представленной на рис. 5 лабораторного занятия 2.

Для выполнения задания нужно выполнить следующие операции:

1. Открыть базу данных **Фирма**. Подайте команду **Схема данных** на вкладке **Работа с базами данных** в группе **Показать или скрыть.**
2. В диалоговом окне **Добавление таблицы** выбрать таблицу **Заказы** и нажать кнопку **Добавить**. В диалоговом окне **Схема документа** появится выбранная таблица. Добавить в схему документа остальные таблицы и закрыть диалоговое окно **Добавление таблицы.**
3. Разместить таблицы в окне так, как показано на рис. 11. Окно содержит изображение каждой таблицы базы данных с перечнем всех включенных в них полей. Ключевые поля обозначены «золотым ключиком».
4. Для ***добавления связи*** между таблицами следует воспользоваться следующим приемом: с помощью мыши переместить поле (или поля), которое необходимо связать, из исходной (главной) таблицы в соответствующее поле (или поля) второй (подчиненной) таблицы. Например: установите курсор мыши на поле **НомерКлиента** в таблице **Клиенты**, нажмите левую кнопку мыши и, удерживая ее нажатой, переместите курсор до его совмещения со строкой с тем же именем **НомерКлиента** в таблице **Заказы**. В

большинстве случаев ключевое слово первой таблицы связывается с аналогичным полем второй.

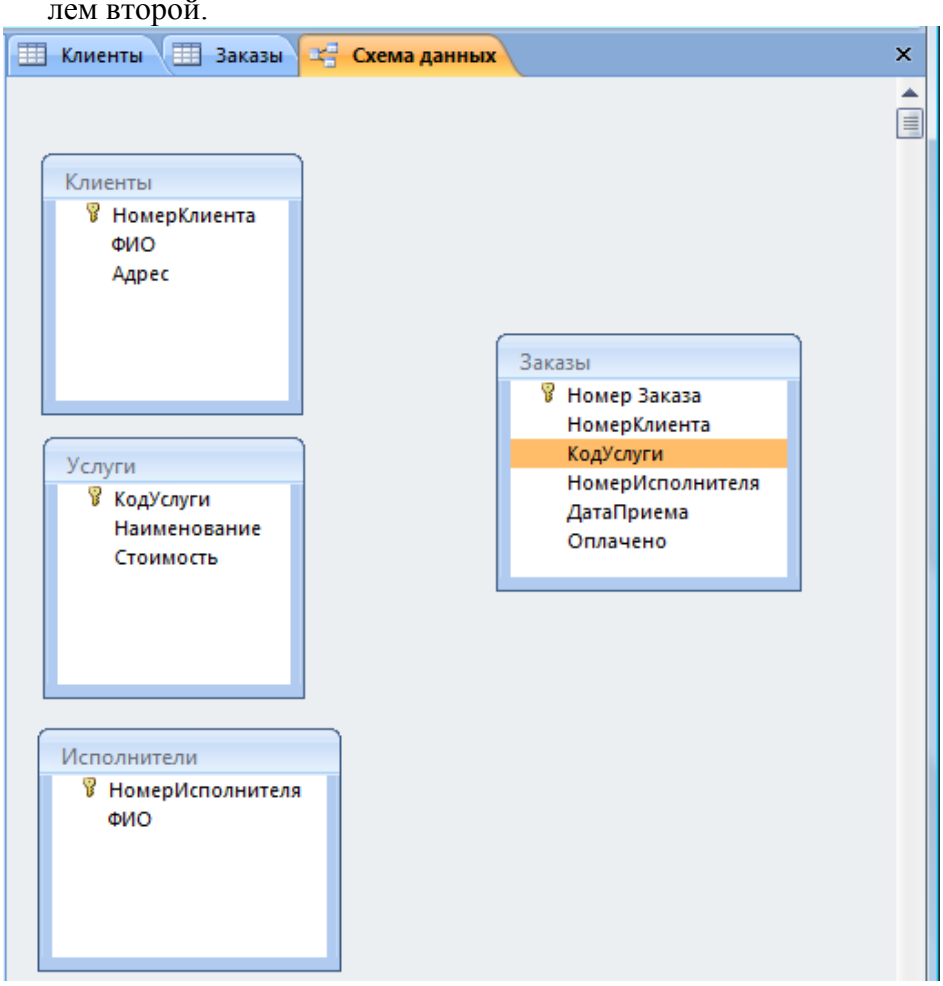


Рисунок 11. Диалоговое окно Схема связи с выбранными таблицами

5. После перемещения поля появляется диалоговое окно **Изменение Связей**, в котором, если это необходимо, можно изменить имена полей. В этом же окне (рис. 12) можно установить флажок **Обеспечение целостности данных**, что дает возможность указать вариант обновления информации в связанных таблицах при внесении изменений в исходную таблицу: связанные поля могут быть каскадно удалены или обновлены, что сохранит согласованность данных в различных таблицах БД. Установите все три флажка: **Обеспечение целостности данных, Каскадное обновление связанных полей, Каскадное удаление связанных полей**. Нажмите кнопку **Создать**. Между полями **НомерКлиента** будет установлена связь **Один-ко-многим.**

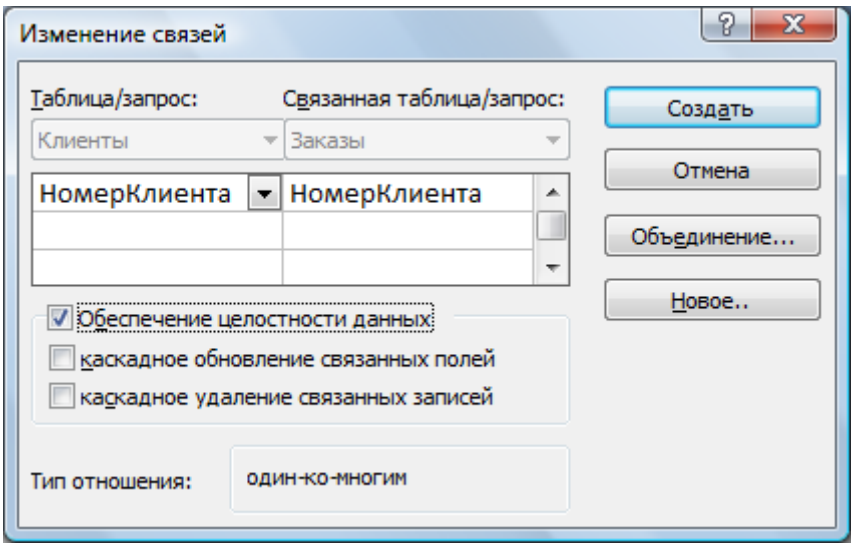


Рисунок 12. Диалоговое окно Изменение связей

6. Установите остальные связи между таблицами так, как показано на рис. 13.

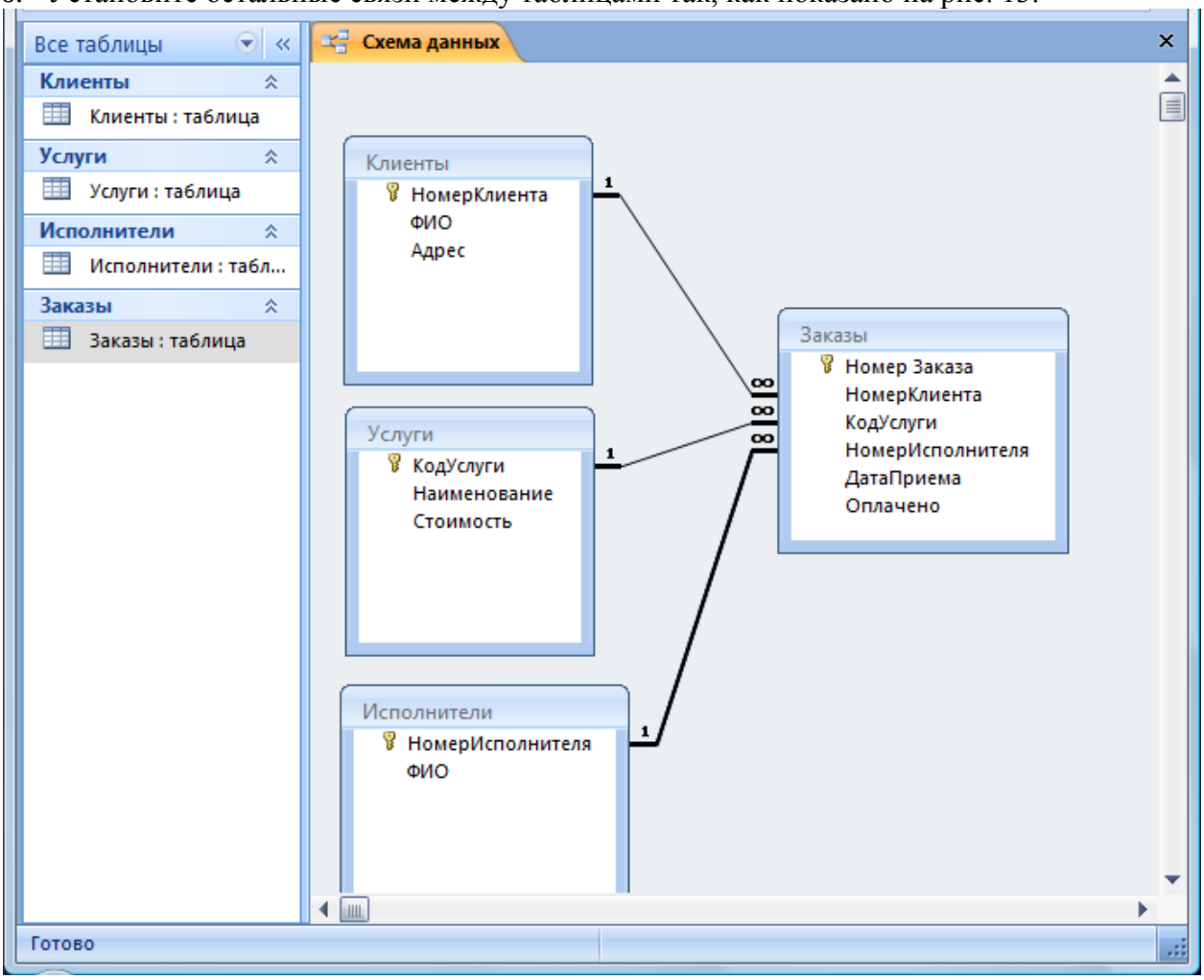


Рисунок 13. Диалоговое окно с установленными связями

7. Закройте схему связей, подтвердив сохранение изменений.

Примечание. Ошибки при установке связей чаще всего возникают из-за того, что неверно были установлены тип данных или свойства полей при создании структур таблиц. Для исправления ошибок следует вернуться к соответствующей таблице в режиме **Конструктор**.

Вопросы для самоконтроля:

1. Какие связи могут существовать между таблицами в СУБД?

2. Что обеспечивает создание связей?
3. Как разместить в схему данных все необходимые связываемые таблицы?
4. Как установить связь между таблицами?
5. Зачем выставляется флажок **Обеспечение целостности данных**?
6. Что нужно делать, если целостность данных при создании связей не обеспечивается?

3.2. Ввод данных в таблицы БД

Ввод данных в реляционную базу данных целесообразно осуществлять последовательно, начиная с данных которые известны заранее. Так руководитель фирмы, прежде чем принимать заказы, должен определиться с перечнем услуг, которые будет оказывать фирма и с исполнителями этих услуг. Эти данные рассматривались на практическом занятии 1. Ввод данных можно осуществить напрямую, открыв таблицу. При этом данные в таблице отображаются в форме, сходной с представлением данных в электронных таблицах (рис.14). В каждой строке размещается информация об одном объекте.

Услуги

КодУслуги	Наименова	Стоимость
1	верстка	3
2	набор	7
3	печать	2
4	сканирование	1
5	копирование	2
(№)		

Рисунок 14. Таблица Услуги

Услуги | Исполнители

НомерИспс	ФИО
1	К.К.Кукушкин
2	И.О.Гришин
3	П.П.Петров
4	С.С.Сидоров
5	А.А.Андреев
6	А.А.Александ
7	М.К.Сергеев
8	А.П.Аркадьев
9	А.А.Антонов
(№)	

Рисунок 15. Таблица Исполнители

Задание 7. Введите в таблицу **Услуги** данные, представленные на рис. 14, выполнив следующие операции:

1. Откройте таблицу **Услуги** в режиме просмотра, щелкнув по ее имени на левой боковой панели.
2. Введите данные в поля **Наименование** и **Стоимость**. Данные в поле **КодУслуги** записываются автоматически при создании новой записи.
3. Закройте таблицу, сохранив изменения.

Задание 8. Введите в таблицу **Исполнители** данные, представленные на рис. 15, выполнив следующие операции:

1. Откройте таблицу **Исполнители** в режиме просмотра.
2. Введите данные (инициалы и фамилию) в поле **ФИО.** С учетом маски ввода эти данные можно вводить только строчными буквами и без знаков препинания. Например: **кккукушкин** – будет отображено как **К.К.Кукушкин**. Таким образом, маска ввода позволяет не только записать однотипные данные в одном формате, но и упростить ввод. Поле **НомерИсполнителя** будет заполняться счетчиком автоматически.
3. Закройте таблицу, сохранив изменения.

Фирма готова к работе: определен перечень оказываемых услуг и их исполнители – сотрудники фирмы!

Такой способ ввода данных оказывается неудобным для ввода данных в подчиненные таблицы (например, в таблицу **Заказы**), так как для ввода данных в нее нужно знать значения ключевых полей соответствующих записей из главных таблиц. Например: заказ от-

носится к конкретному клиенту, следовательно, чтобы связать его с клиентом необходимо знать номер клиента, заказ делается на конкретную услугу, следовательно, нужно знать ее код, заказ поручается конкретному исполнителю, следовательно, нужно знать его номер. Реальные таблицы базы данных обычно достаточно большие и содержат десятки столбцов и сотни строк. Естественно, что такая таблица целиком не помещается на экране. Кроме того, в таких таблицах, как правило, заполняются или редактируются не все столбцы. Выполнять такие операции в большой таблице неудобно. Для удобства работы с данными в БД создаются формы.

3.3. Создание форм

Форма в СУБД - это диалогового окна для просмотра и редактирования данных, содержащихся в отдельной записи БД. Для создания форм используется группа **Формы** на вкладке **Создание** инструментальной ленты (рис. 16): **Форма, Разделенная форма, Несколько элементов, Другие формы, Сводная диаграмма, Пустая форма, Конструктор форм.**

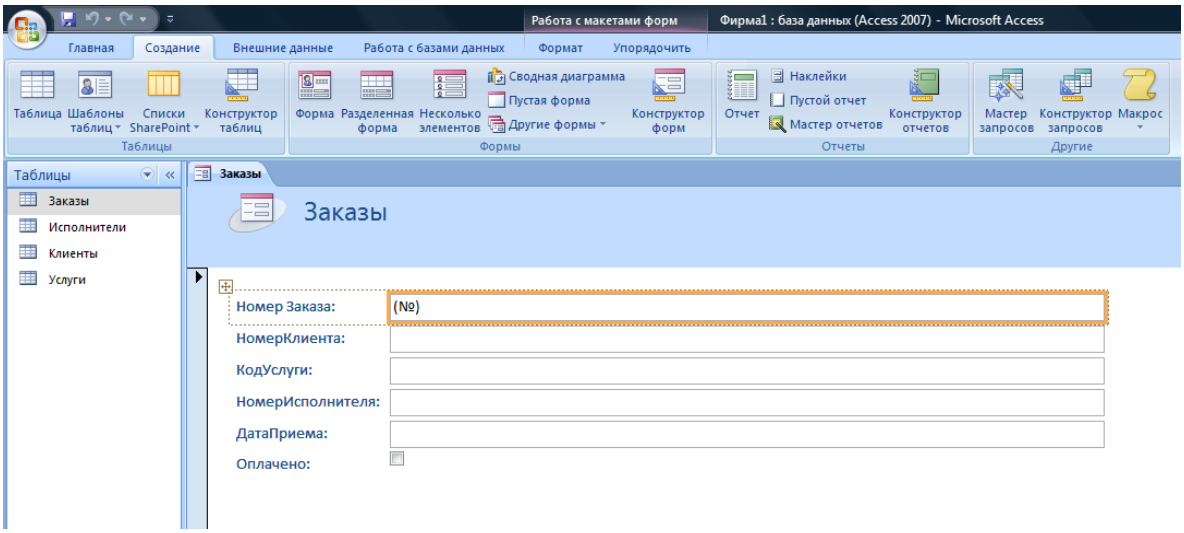


Рисунок 16. Средства для создания форм

Наиболее простой способ создания формы – использование команды **Форма**. Диалоговое окно формы создается автоматически в зависимости от выделенной таблицы.

В форму по желанию пользователя могут быть включены элементы управления для ввода данных в таблицу и кнопки, позволяющие перемещаться между записями таблицы. Созданная форма отображается на экране, при ее закрытии Access запрашивает подтверждение ее сохранения и имя (обычно имя совпадает с именем соответствующей таблицы).

Задание 9. Создайте форму для заполнения таблицы **Клиенты**, выполнив следующие операции:

1. В окне базы данных на левой боковой панели выберите таблицу **Клиенты**.
2. Нажмите кнопку **Форма** в группе **Формы** на вкладке **Создание**. Появится окно с формой **Клиенты**, в нижней части которого указана связанная с таблицей **Клиенты** таблица **Заказы**.
3. Закрыть появившуюся форму с помощью кнопки ☒.
4. Подтвердить сохранение формы щелчком по кнопке **Да**.
5. Ввести имя формы в окне **Сохранение** и щелкнуть кнопку **ОК**.

Задание 10. Самостоятельно создайте форму для заполнения таблицы **Услуги**.

3.3.1. Создание форм с помощью мастера

Мастер форм строит форму в диалоге с пользователем. Он позволяет выбрать поля таблицы, которые должны отображаться в форме (рис. 17), выбор полей осуществляется с

помощью кнопок [>] и [>>] (кнопка [>] позволяет отбирать поля записи для вывода в форме по одному, а кнопка [>>] отбирает все поля сразу). Кнопки [<] и [<<] позволяют отказаться от выбора полей. Щелчок по кнопке **Далее** после выбора полей позволяет перейти к следующему этапу создания формы. В следующих диалоговых окнах можно выбрать внешний вид формы (например, в один столбец) и требуемый стиль (например, обычный). Кнопка **Назад** позволяет вернуться в предыдущему шагу. В последнем диалоговом окне **Мастера создания форм** можно ввести имя созданной формы и щелкнуть кнопку **Готово**. Если в этом окне установлен переключатель **Открытие формы для просмотра или ввода данных**, то сразу после щелчка по кнопке **Готово** форма открывается в рабочем режиме.

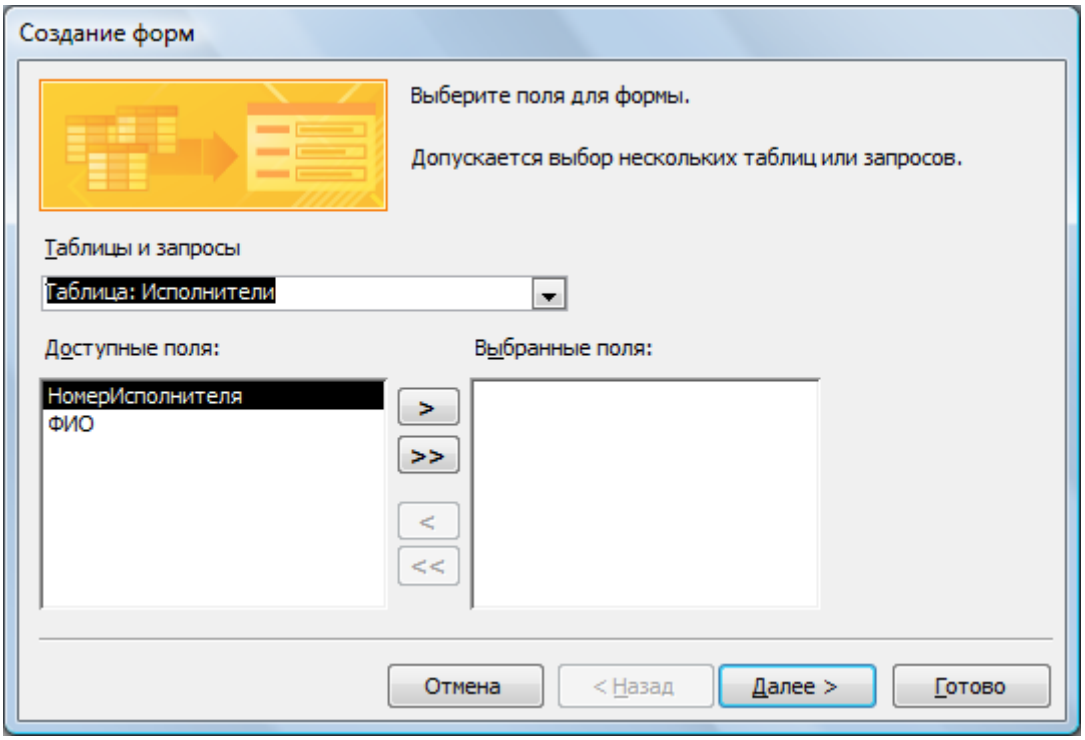


Рисунок 17. Окно мастера создания форм

Задание 11. Самостоятельно создайте форму для таблицы **Исполнители** (включите в нее все поля) с помощью **Мастера создания форм.**

3.3.2. Создание форм вручную

Если требуется создать форму, которую невозможно получить с помощью мастера, можно в режиме конструктора создать пустую форму и разместить на ней элементы управления по своему усмотрению. Это способ является наилучшим в том случае, когда отсутствуют готовые варианты, удовлетворяющие вашим потребностям или когда необходимо разместить группу переключателей, подчиненные формы, колонтитулы с текстовыми полями и управляющие кнопки.

Для создания формы вручную необходимо:

1. Открыть окно конструктора командой **Создание, Конструктор форм**. Откроется пустая форма **Form1**. Для заполнения этой формы используется группа **Элементы управления** вкладки **Конструктор** (рис.18): Поле, Подпись, Кнопка, Поле со списком и т.д.
2. Закройте окно конструктора, присвоив форме соответствующее имя.

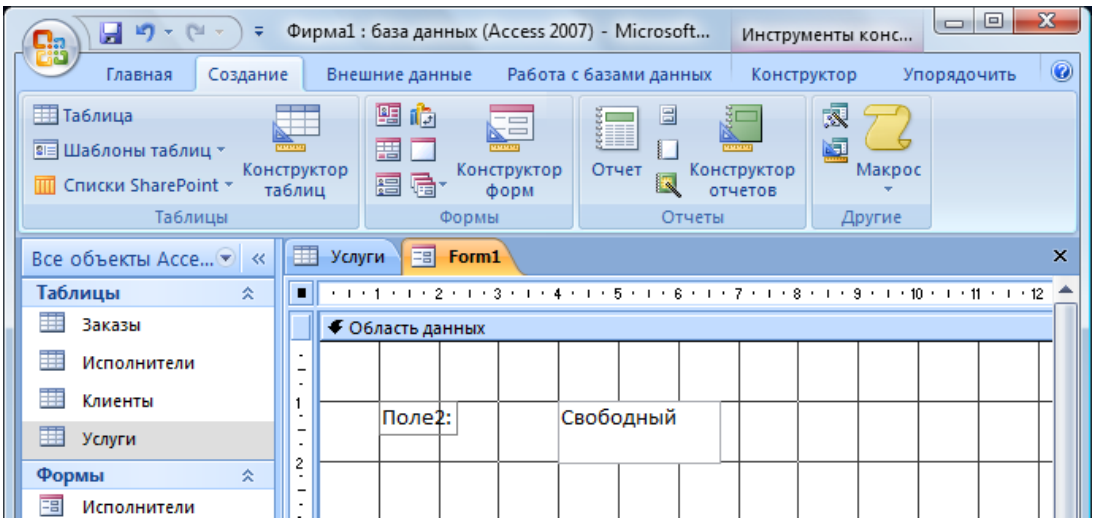


Рисунок 18. Новая пустая форма в режиме конструктора

Вопросы для самоконтроля:

1. Что такое форма? Зачем она нужна?
2. Какие способы создания форм существуют в СУБД?
3. Когда целесообразно использовать **Автоформу**?
4. Как работает **Мастер форм**?

Задание 7. Самостоятельно сформулируйте и запишите вывода по лабораторной работе 2.

Занятие 4. Создание сложных форм для работы с базой данных

4.1. Создание форм, содержащих элементы управления

Для удобства работы с большими таблицами, а также для ввода и редактирования данных в подчиненных таблицах пользователь может создать соответствующие формы, содержащие различные элементы управления. Для расширения возможностей форм в них можно включить командные кнопки. Такие формы создаются с помощью конструктора форм и соответствующей вкладки инструментальной ленты (рис. 19).

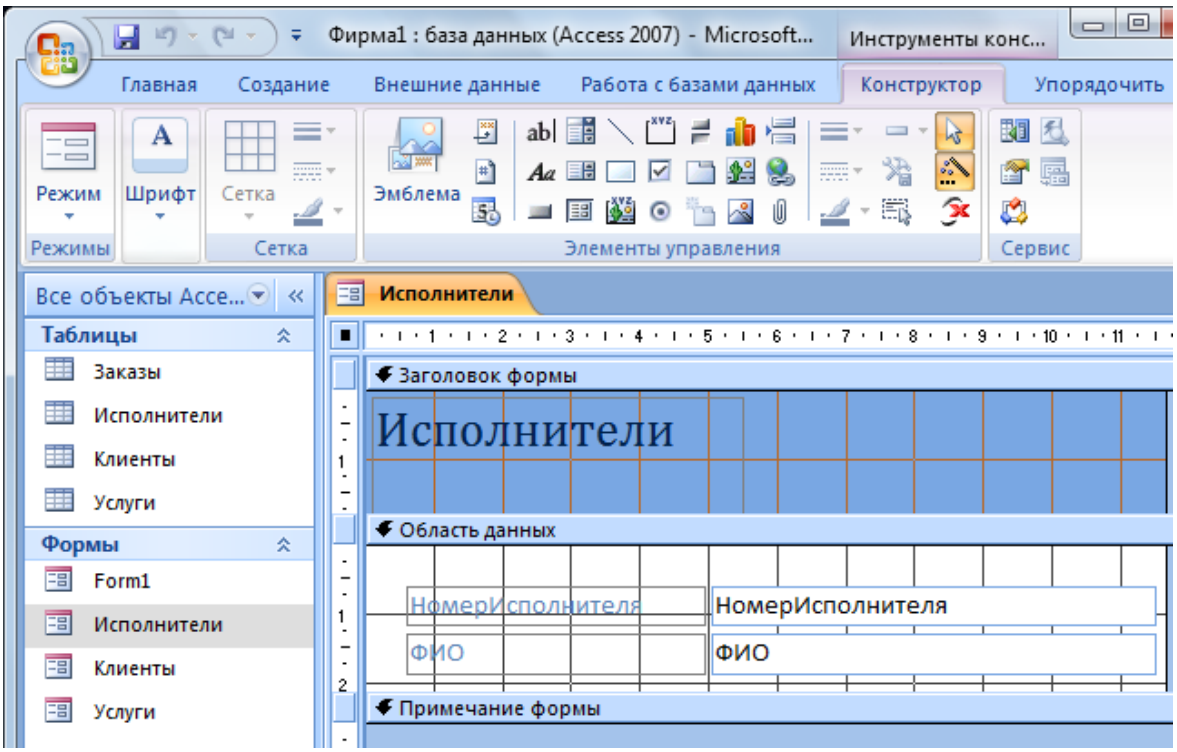


Рисунок 19. Окно конструктора форм с элементами управления

Задание 12. Добавьте в форму **Исполнители** командные кнопку для выполнения операций удаления отображаемой в форме записи.

Для этого нужно выполнить следующие действия:

1. Откройте форму **Исполнители** в режиме конструктора.
2. В окне конструктора с помощью мыши раздвинуть область размещения формы на экране, переместив поле **Примечание формы** вниз так, чтобы освободить место для размещения кнопок (рис. 19).
3. Найдите на вкладке **Конструктор** в группе **Элементы управления** элемент **Кнопка**. Нарисуйте кнопку на форме с помощью мыши (рис.20).
4. В открывшемся диалоговом окне **Создание кнопок** выбрать категорию команды **Обработка записей**, предназначенную кнопке, и выбрать конкретное действие **Удалить запись**, которое будет выполняться при нажатии на кнопку. Нажать кнопку **Далее.**
5. В следующем диалоговом окне «оформите кнопку»: назначьте ей рисунок или введите текст (**Удалить**), который будет отображаться на кнопке, и щелкните кнопку **Готово**.
6. Проверьте работоспособность созданной кнопки. Для этого сначала занесите с помощью формы **Исполнители** новую запись в таблицу. Затем удалите новую запись с помощью созданной кнопки **Удаление.**

Задание 13. Создайте кнопку в форме **Исполнители** для поиска нужной записи, выполнив следующие операции:

1. Откройте форму **Исполнители** в режиме конструктора.
2. Нарисуйте еще одну кнопку
3. В окне **Создание кнопок** для категории **Переходы по записям** выберите действие **Найти запись**. Нажмите кнопку **Далее.**
4. В следующем окне мастера создания кнопок установите переключатель **Текст**. В поле слева от переключателя должна быть размещена та надпись, которую вы хотите видеть на кнопке управления (**Поиск**). Нажмите кнопки **Далее** и **Готово**. В форме **Исполнители** появиться появится новая кнопка **Поиск.**

Проверьте работоспособность созданной кнопки, организовав поиск любой записи в

таблице **Исполнители**. Для этого следует:

1. Открыть форму **Исполнители** и перейти к первой записи.
2. Установить курсор в поле ввода **ФИО**.
3. Нажать кнопку **Поиск**.
4. В диалоговом окне **Поиск и замена** на вкладке **Поиск** записать образец для поиска (например: А.А.А). В раскрывающемся списке **Просмотр** выбрать позицию **Все**; в раскрывающемся список **Совпадение** установить - **С начала поля.** Нажмите кнопку **Найти далее** Для поиска следующей записи, удовлетворяющей заданным условиям, нужно щелкнуть кнопку **Найти далее**.
5. После завершения поиска закройте диалоговое окно поиска.
6. Повторите процедуру поиска, установив свой критерий поиска.

5.

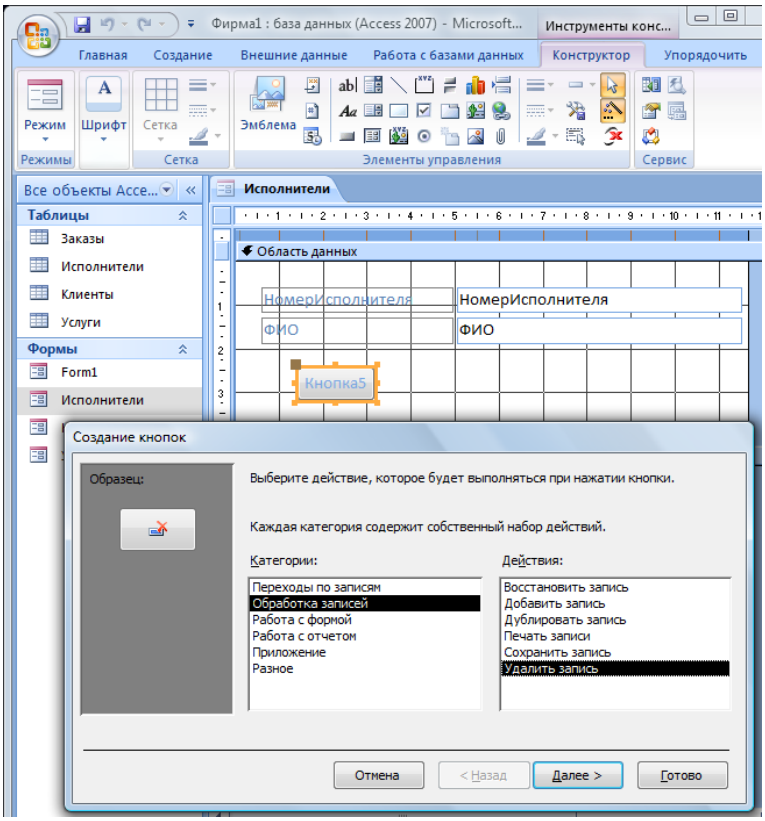


Рисунок 20. Окно конструктора формы и окно мастера создания кнопок в процессе создания кнопки

Задание 14. Добавьте кнопки удаления и поиска в формы **Услуги** и **Клиенты**.

4.2. Работа с данными с помощью формы

Форма позволяет просматривать таблицу, вносить новые строки, редактировать существующие записи, удалять ненужные, сортировать записи в таблицах и т.д.

Задание 15. Отсортируйте записи в списке исполнителей, упорядочив по полю **ФИО** в алфавитном порядке

Для этого нужно установить курсор в поле **ФИО** и выполнить команду **Сортировка по возрастанию** в группе **Сортировка** и фильтр на вкладке **Главная**.

Формы можно ***копировать***. Для этого следует открыть окно базы данных, выделить нужную форму на левой боковой панели, щелкнув по форме правой кнопкой мыши и выполнить команду **Копировать** в контекстном меню. Затем щелкнуть правой кнопкой мыши по свободному месту левой боковой панели и выполнить команду **Вставить** (при выполнении команды вводится имя, под которым запишется в БД копия формы).

Задание 166. Скопируйте форму **Клиенты**, записав копию в БД под именем **Данные о клиентах**.

Вопросы для самоконтроля:

1. Как с помощью формы внести новую запись в таблицу?
2. Порядок поиска записи с помощью формы.

4.3. Создание сложных форм

Сложные формы создаются для работы сразу с несколькими таблицами и помимо кнопок управления могут содержать поля различного назначения, раскрывающиеся списки, флажки, формулы т.д. Такие формы позволяют создать удобную информационную среду для работы со сложными базами данных, организовав диалог только с теми полями таблиц, которые нужны пользователю для выполнения конкретной работы. С их помощью можно разграничить доступ пользователей к соответствующим разделам базы данных.

Сложные формы являются удобным механизмом для работы с подчиненными таблицами. Для создаваемой базы данных **Фирма** такой подчиненной таблицей является таблица **Заказы**. Непосредственное заполнение этой таблицы требует, чтобы пользователь помнил номера всех клиентов, коды всех услуг и номера всех исполнителей. Это очень неудобно, так как человек привык работать с фамилиями (а не номерами) и наименованием услуг (а не их кодами). Создание сложных форм позволяет решить эту проблему.

Обычно сложные формы сначала создаются с помощью **Мастера создания форм,** а затем дорабатываются с помощью **Конструктора**.

Задание 177. Создайте с помощью **Мастера** форму для таблицы **Заказы**, выполнив следующие операции:

1. На вкладке **Создание** в группе **Формы** раскройте список **Другие формы** и выберите пункт **Мастер форм**.
2. В окне мастера **Создание форм** с помощью списка **Таблицы и запросы** выберите таблицы **Заказы**. Из списка **Допустимые поля** переместите в список **Выбранные поля** только следующие поля: **НомерЗаказа, ДатаПриема, ДатаИсполнения.** В последнем диалоговом окне установите переключатель «**Изменить макет формы**» (это приведет к переключению в режим **Конструктора**) и щелкните кнопку **Готово**.
3. В окне конструктора расширьте форму так, чтобы в ней можно было разместить новые элементы (переместите поле **Примечание формы** вниз с помощью мыши).
4. Измените надписи перед полями «**НомерЗаказа**», «**ДатаПриема**» и «**ДатаИсполнения**», установив курсор в надпись двойным щелчком и введя с клавиатуры строки «**Номер заказа**», «**Дата приема**» и «**Дата исполнения**» соответственно. Такое оформление полей больше соответствует стандартному. Обратите внимание на то, что изменять следует текст только в левых прямоугольниках (надписях к полю). Значение самих полей изменять не надо!
5. Дата приема заказа обычно совпадает с текущей датой. Поэтому можно установить для этого поля значение по умолчанию – текущую дату и запретить ее «ручной» ввод с клавиатуры. Для этого:
   - щелкните по полю **ДатаПриема** правой кнопкой мыши и в открывшемся контекстном меню выберите строку **Свойства**;
   - в открывшемся диалоговом окне на вкладке **Данные** выберите строку **Значение по умолчанию** и щелчком по кнопке … вызовите **Построитель выражений**;
   - в окне программы-построителя раскройте щелчком по значку **+** папку **Функции** и двойным щелчком вложенную в нее папку **Встроенные функции**;
   - в списке категорий встроенных функций выберите категорию **Дата/время** и в перечне функций этой категории – функцию **Date**, задающую текущую системную дату, установленную в компьютере;

- вставьте эту функцию щелчком по кнопке **Вставить** в выражение, вычисляющее значение по умолчанию;
- закройте построитель выражений, нажав кнопку **ОК**;
- в диалоговом окне свойств поля установите, что к полю нет доступа, и есть блокировка;

6. При приеме заказов пользователю удобнее иметь дела с наименованием услуги, а не ее кодом. Пользователь записывает наименование услуги, а ее код компьютер формирует сам. Для включения в форму данные об услуге можно воспользоваться раскрывающимся списком предоставляемых услуг. Вставьте раскрывающийся список в форму, выполнив следующие операции:
   - Выберите на панели элементов управления элемент **Поле со списком**. Поместите поле на форме с помощью мыши.
   - В открывшемся диалоговом окне установите переключатель **Объект «поле со списком» будет использовать значения из таблицы или запроса** и щелкните кнопку **Далее.**
   - В следующем диалоговом окне установите в группе **Показать** переключатель **Таблицы** и выберите в открытом списке таблиц таблицу **Услуги**. Щелкните кнопку **Далее.**
   - Выберите в очередном окне поля **КодУслуги** и **Наименование** таблицы **Услуги** для включения в форму и щелкните кнопку **Далее.**
   - В следующем окне установите флажок **Скрыть ключевой столбец** и щелкните кнопку **Далее** для перехода к очередному окну **Мастера.**
   - Установите переключатель **Сохранить в поле** и выберите из списка поле **КодУслуги** таблицы **Заказы**. Щелчком по кнопке **Далее** перейдите в следующее окно.
   - Задайте имя **Услуга** и щелкните кнопку **Готово.**
7. Пользователю будет удобнее работать, если наряду с раскрывающимся списком услуг в форму вставить поле **Стоимость**. Для включения поля **Стоимость** в форму заказа выполните следующие операции:
   - Среди элементов управления на вкладке **Конструктор** выберите элемент **Поле** и введите поле в форму.
   - Щелкните по полю правой кнопкой мыши и в открывшемся контекстном меню выберите строку **Свойства;**
   - В диалоговом окне команды на вкладке **Данные** выберите строку **Данные** и щелчком по кнопке … вызовите **Построитель выражений** (рис. 21).
   - Запишите формулу для определения стоимости (выбора из таблицы **Услуги** стоимости по коду услуги). Для этого нужно ввести выражение, содержащее, например, следующую функцию:

     =DLookUp("[Стоимость]";"Услуги";"[Услуги]![КодУслуги] = " &
     [Forms]![Данные о клиентах]![Заказы]![КодУслуги])

     (стоимость выбирается из поля **Стоимость** таблицы **Услуги**, причем выбирается стоимость той услуги, код которой будет зафиксирован при оформлении заказа клиентом). Для записи этой формулы использованы конструкции языка программирования Visual Basic for Applications (VBA). С этим языком можно познакомиться на специальных факультативных занятиях.
   - Установите блокировку и отсутствие доступа в свойствах поля (стоимость нельзя менять при оформлении заказа).
   - С левой стороны от поля в надпись вставьте слово **Стоимость**
8. Разместите на форме поле со списком для выбора **исполнителя** заказа, выполнив операции:
   - Выберите на панели элементов управления элемент **Поле со списком**. Поместите поле на форме с помощью мыши.
   - В открывшемся диалоговом окне установите переключатель **Объект «поле со списком» будет использовать значения из таблицы или запроса** и щелкните кнопку **Далее.**

- В следующем диалоговом окне установите в группе **Показать** переключатель **Таблицы** и выберите в открытом списке таблиц таблицу **Исполнители**. Щелкните кнопку **Далее**.
- Выберите в очередном окне поля **НомерИсполнителя** и **ФИО** таблицы **Исполнители** для включения в форму и щелкните кнопку **Далее**.
- В следующем окне установите флажок **Скрыть ключевой столбец** и щелкните кнопку **Далее** для перехода к очередному окну **Мастера**.
- Установите переключатель **Сохранить в поле** и выберите из списка поле **НомерИсполнителя**. Щелчком по кнопке **Далее** перейдите в следующее окно.
- Задайте имя **Исполнитель** и щелкните кнопку **Готово**.

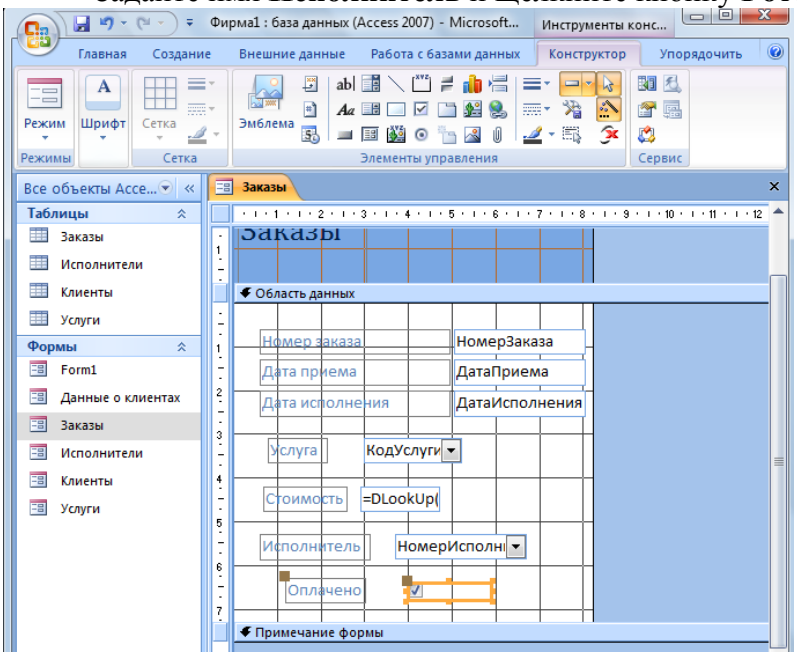


Рисунок 21. Форма Заказы в режиме конструктора

9. Для отображения информации о том, оплачен ли заказ, на форме следует разместить **Флажок**. Для размещения флажка выполните следующие операции:
   - Выбрать элемент управления **Флажок** среди элементов управления и с помощью мыши разместить его на форме.
   - Щелчком мыши установить текстовый курсор в поле надписи флажка и ввести надпись **Оплачено**.
   - Щелчком правой кнопки мыши по флажку вызвать контекстное меню и выбрать в нем команду **Свойства**. На вкладке **Данные** диалогового окна свойств флажка выбрать строку **Данные** и в списке выбрать поле **Оплачено** (таким образом значение флажка связывается в полем таблицы **Заказы**). После ввода этой информации диалоговое окно нужно закрыть.
   - Закройте **Конструктор** и подтвердите сохранение внесенных в форму изменений.
   - Откройте форму **Заказы** для просмотра (рис. 22).

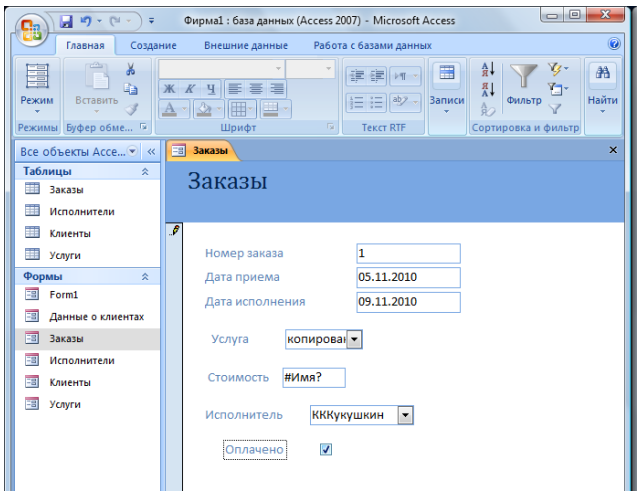


Рисунок 22. Форма Заказы в режиме просмотра

Примечание: Если вывести разрабатываемую форму **Заказы** на экран в режиме просмотра, то в поле **Стоимость** будет показана ошибка до тех пор, пока эта форма не будет вставлена в более сложную форму, обеспечивающую связь между таблицами базы данных.

Для отображения полной информации о заказчике и всех его заказах нужно создать сложную реляционную форму, связывающую информацию из нескольких таблиц.

Задание 188. Создать реляционную форму путем внедрения формы **Заказы** в форму **Данные о клиентах**. Для этого:

1. Выделите форму **Данные о клиентах** и откройте ее в режиме **Конструктор**;
2. Перетяните значок формы **Заказы** из левой боковой панели в нижнюю часть формы **Данные о клиентах** (копия формы **Клиенты**) на свободное место;
3. Отформатируйте элементы формы так, чтобы все они размещались на ней, и закройте конструктор.

Полученная форма должна иметь вид, показанный на рис. 23.

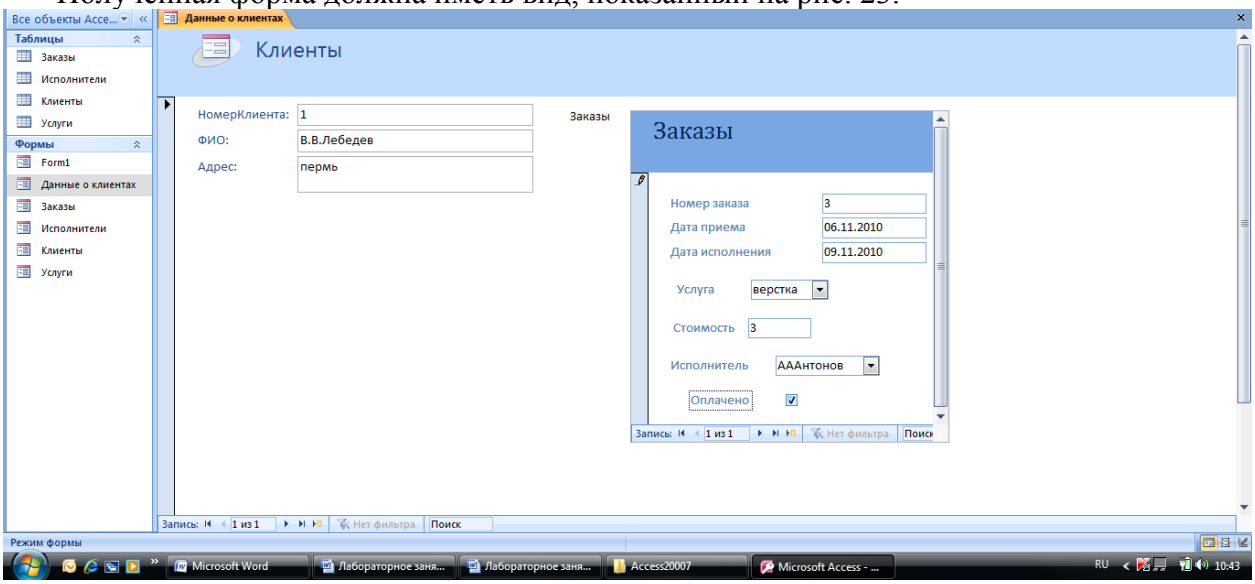


Рисунок 23. Форма Данные о клиентах. используемая для приема заказов

Задание 199. Введите с помощью формы **Данные о клиентах** информацию о заказчиках и «оформите» заказы для них, используя произвольные данные.

Таким образом, формы позволяют последовательно просматривать записи БД, искать нужные записи, удалять записи при необходимости. Для отбора информации по более сложным критериям, анализа и подведения итогов в БД создаются запросы и отчеты.

Вопросы для самоконтроля:

1. Как в форму вставить текущую дату или формулу?
2. Как вставить в форму раскрывающийся список и флажок?
3. Как создать реляционную форму из нескольких форм?

Занятие 5. Работа с запросами и отчетами

5.1. Создание и фильтрация запросов

Для отбора информации из БД создаются специальные запросы, которые позволяют пользователю обратиться к базе данных с любым вопросом и получить на него ответ в виде виртуальной таблицы. В рамках полученного запроса можно производить обработку данных в том числе и фильтрацию. Отбор данных из БД с помощью запроса, осуществляется в соответствии с критерием, который формирует пользователь в зависимости от характера возникающего вопроса.

Задание 1. Создать запрос **Оплата заказов**, позволяющий вывести необходимые данные из нескольких таблиц БД. Для выполнения задания необходимо:

1. В окне БД выбрать объект **Запросы**. На вкладке **Создание** в группе **Другие** нажать кнопку **Мастер запросов**.
2. В диалоговом окне **Новый запрос** выбрать тип запроса (выберем наиболее простой вариант – **Простой запрос**) и щелкнуть кнопку **ОК**.
3. В списке **Таблицы и Запросы** выберите строку **Таблица: Заказы**. С помощью кнопки [>] выберите из списка доступных в таблице полей поля **НомерЗаказа**, **ДатаПриема**, **ДатаИсполнения** и **Оплачено** для включения их в запрос.
4. В списке «**Таблицы и Запросы**» выберите строку **Таблица: Клиенты**. С помощью кнопки [>] выберите из списка доступных в таблице полей поле **ФИО** для включения его в запрос.
5. В списке **Таблицы и Запросы** выберите строку **Таблица: Услуги**. С помощью кнопки [>] выберите из списка доступных в таблице полей поля **Наименование** и **Стоимость** для включения их в запрос.
6. Щелкните кнопку **Далее**.
7. В следующем диалоговом окне установите переключатель «**Подробный**…» и щелкните кнопку **Далее**.
8. Задайте имя запроса **Оплата заказов**, установите переключатель **Открыть запрос для просмотра данных** и щелкните кнопку **Готово**.

НомерЗака	ДатаПрием	Дата Испол	Оплачено	Фио	КодУслуги	Наименова	Стоимость
11	03.03.2006	15.03.2006	☑	И.И.Иванов	2	верстка	3
12	03.03.2006	23.03.2006	☐	И.И.Иванов	6	сканирование	1
13	03.03.2006	14.03.2006	☑	С.С.Сидоров	7	копирование	2
14	03.03.2006	23.03.2006	☐	И.И.Ильин	2	верстка	3
15	03.03.2006	10.03.2006	☑	И.И.Ильин	3	набор	7
16	03.03.2006	25.03.2006	☐	И.И.Ильин	7	копирование	2
17	03.03.2006	04.04.2006	☑	О.О.Овсов	3	набор	7
18	03.03.2006	14.03.2006	☑	О.О.Овсов	4	печать	2
19	03.03.2006	23.04.2006	☑	К.П.Шишкин	6	сканирование	1
20	03.03.2006	12.04.2006	☐	К.П.Шишкин	7	копирование	2
*	(№)		☐		(№)		

Рисунок 24. Запрос на выборку Заказы

Результаты созданного запроса будут представлены на экране в виде таблицы. Этот запрос объединяет нужные сведения из трех таблиц. Поля в запросе путем перетаскивания можно менять местами. К этой таблице можно применить фильтр для выделения части информации, нужной пользователю.

Задание 2. С помощью фильтра выведите информацию только об оплаченных заказах. Для этого необходимо:

1. Открыть запрос **Оплата заказов** в режиме просмотра.
2. На вкладке **Главная** в группе **Сортировка и фильтр** раскрыть список **Параметры расширенного фильтра** и выбрать пункт **Изменить фильтр**.
3. Установить с помощью мыши флажок в поле **Оплачено**.

4. В группе **Сортировка и фильтр** раскрыть список **Параметры расширенного фильтра** и выбрать пункт **Применить фильтр**.

В результате отбора информации с помощью фильтра в таблице остались только записи об оплаченных заказах.

СУБД позволяет сформировать различные виды запросов:

1. *Запрос-выборка* – позволяет выбрать группу записей из одной или нескольких таблиц в соответствии с заданными условиями отбора.
2. *Запрос-обновление* – вносит изменения в группу записей, которые удовлетворяют заданному условию.
3. *Запрос-удаление* – позволяет удалить из БД записи, удовлетворяющие заданному условию.
4. *Запрос-добавление*.
5. *Запрос на создание таблиц*.
6. *Перекрестный запрос*.

Условия отбора формируются с помощью операций отношения и логических операций. В СУБД эти операции обозначаются следующим образом.

Операции отношения	
=	равно
< >	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Логические операции	
И	And
ИЛИ	Or
НЕ	Not

Для формирования таких запросов необходимо:

1. В окне базы данных выбрать объект **Запросы** и на вкладке **Создание** в группе **Другие** нажать кнопку **Конструктор запросов**.
2. Появятся два диалоговых окна: **Запрос** и **Добавление таблицы**. В окне **Добавление таблицы** нужно выбрать необходимую таблицу и нажать кнопку **Добавить**. Повторить эту операцию для следующей таблицы. Когда все таблицы будут добавлены следует нажать на кнопку **Закрыть**. В окне **Запрос** появятся все выбранные таблицы с установленными связями (рис. 25).

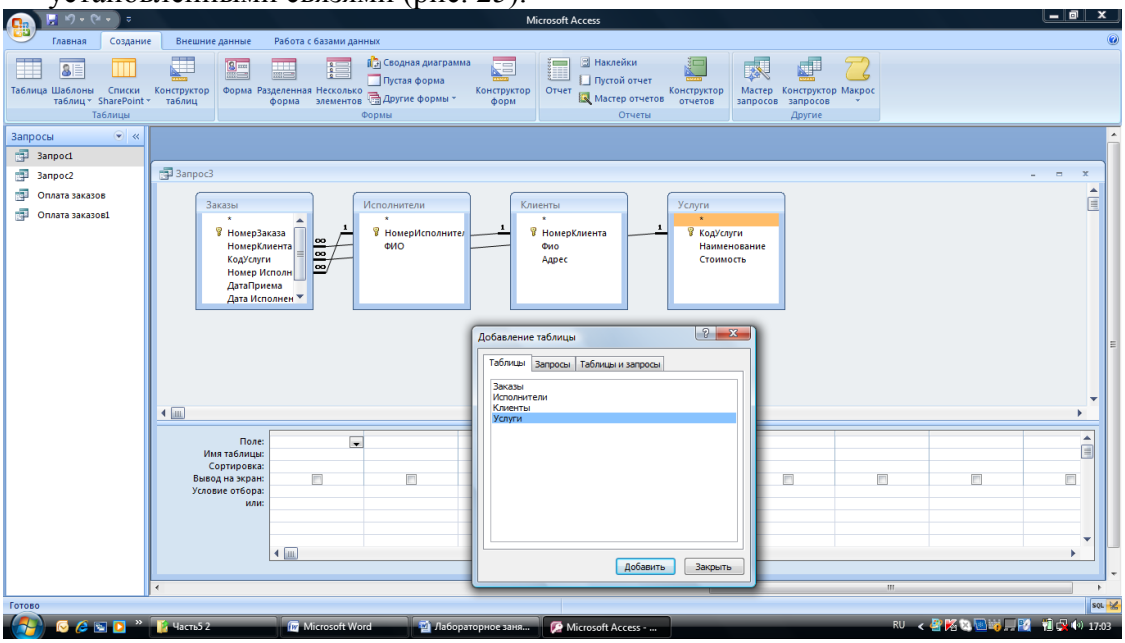


Рисунок 25. Диалоговые окна для формирования запросов

3. На вкладке **Конструктор** в группе **Тип запроса** выбрать нужный тип запроса.
4. Записать условия отбора в соответствующих полях окна запроса.

Задание 3. Создайте запрос на выборку, позволяющий вывести все заказы на сканирование или набор, дата исполнения которых до 15.03.2006 (дата может быть изменена).

Для выполнения задания необходимо в режиме конструктора сформировать запрос представленный на рис. 26.

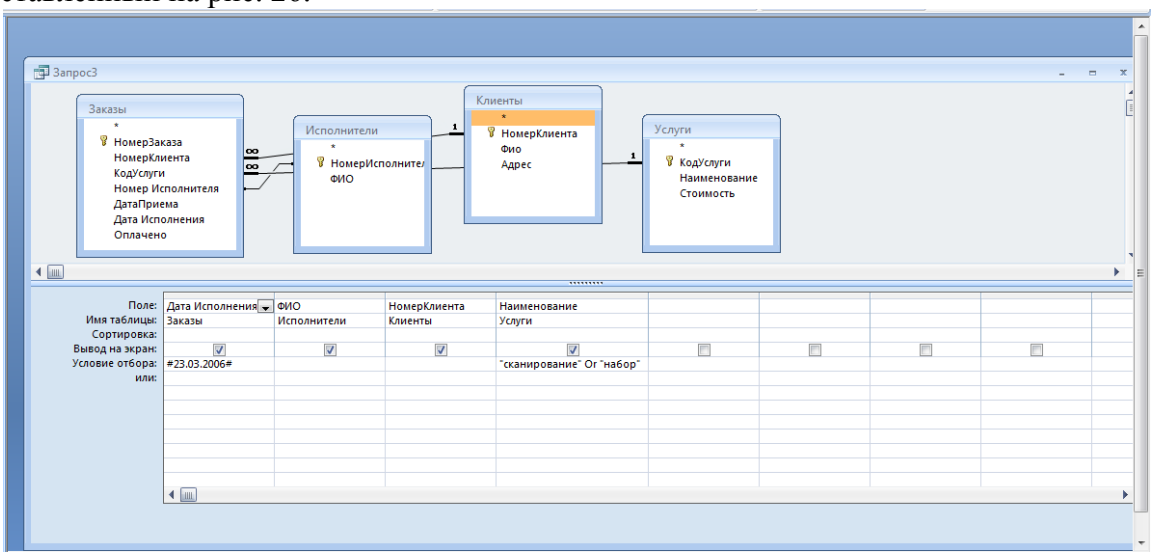


Рисунок 26. Пример запроса на выборку

Задание 4. Самостоятельно сформировать следующие запросы:

- Запрос на выборку: выбрать все заказы, выполненные Сергеевым в период с 01.03.2006 до 30.03.2006, стоимость которых больше 2 единиц (даты и стоимость могут быть изменены).
- Любой запрос на обновление.
- Любой запрос на удаление.

Вопросы для самоконтроля:

1. Как создать запрос к БД?
2. Какие типы запросов можно сформировать в СУБД?

5.2. Создание и просмотр отчетов

Для вывода результатов выполнения запроса можно использовать отчеты.

Задание 5. Создайте отчет на основе запроса **Оплата труда**, выполнив следующие операции:

1. На левой боковой панели выберите запрос **Оплата заказов**.
2. На вкладке **Создание** в группе **Отчеты** нажмите кнопку **Отчет**.
3. Щелкните кнопку **ОК** и сохраните созданный отчет под именем **Оплата заказов**.

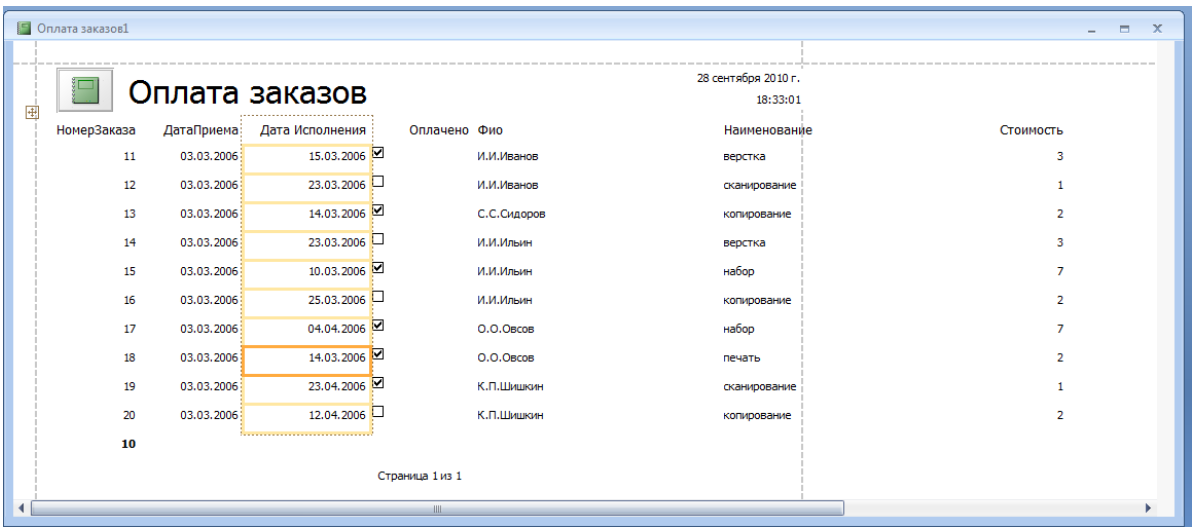


Рисунок 27. Отчет Оплата заказов

Обычно отчет создается с помощью **Мастера отчетов** или с помощью команды **Отчет (**Автоотчет**).** Если отчет, предоставленный компьютером, пользователя не устраивает, то такой отчет можно доработать с помощью **Конструктора**. В отчете можно сгруппировать ряд записей, например, объединить в группу все заказы одного заказчика, посчитать стоимость этих заказов и общую стоимость всех заказов.

Задание 6. Внесите в отчет итоговые данные по оплаченным и неоплаченным заказам. Для модификации отчета выполните следующие шаги:

1. Щелкните по значку отчета **Оплата заказов** правой кнопкой и в контекстном меню выберите команду Конструктор. Откроется диалоговое окно, содержащее следующие разделы:
   - **Заголовок отчета**.
   - **Верхний колонтитул** для размещения имен столбцов (полей) отчета.
   - **Область данных** для размещения полей, содержащих данные.
   - **Нижний колонтитул** может содержать дату создания отчета, номера страниц и т.д.
   - **Примечание отчета** для размещения итоговых данных по всему отчету и т.д.

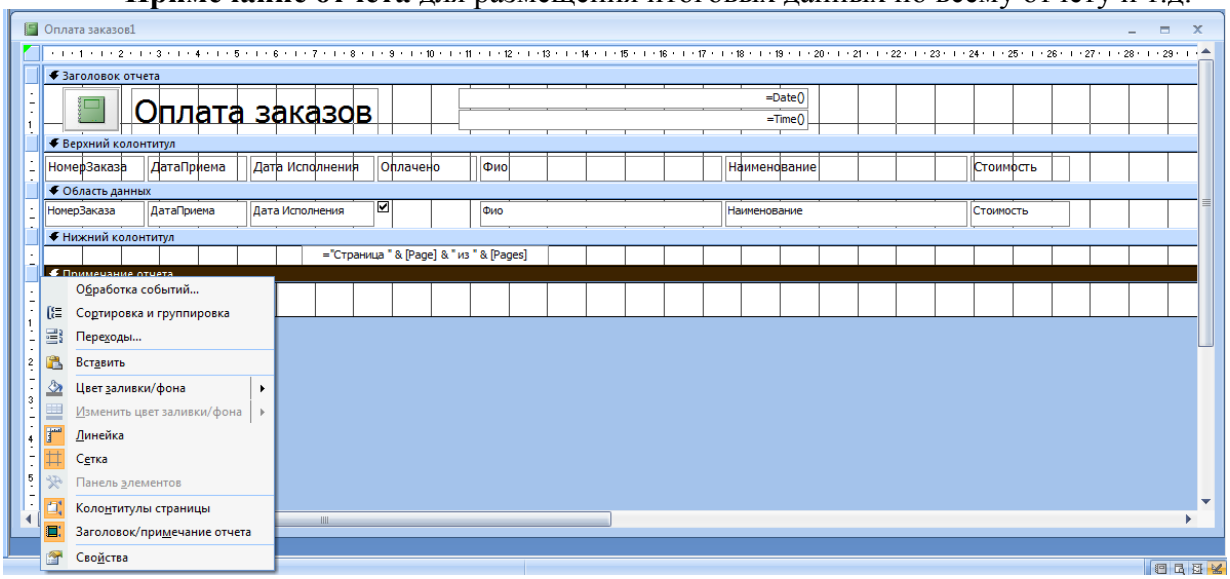


Рисунок 28. Диалоговое окно конструктора отчетов

2. Отсортируйте, а затем сгруппируйте данные в отчете по именам клиентов, выполнив следующие операции:

- В окне конструктора щелкните правой кнопкой мыши по вертикальной линейке (слева) и в контекстном меню выберите команду **Сортировка и группировка**.
- В области **Группировка, сортировка и итоги** нажмите кнопку **Добавить сортировку** и в открывшемся списке выберите поле **ФИО** и установите группировку по возрастанию.

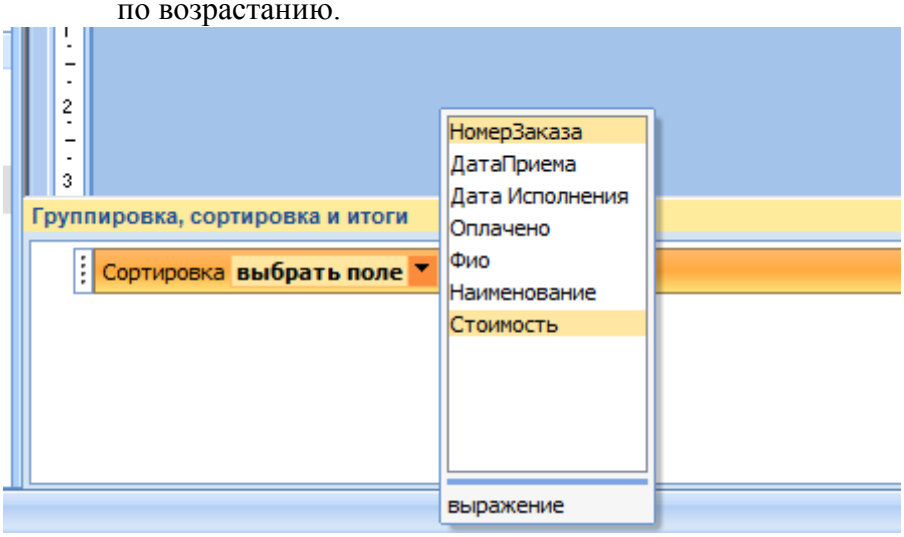


Рисунок 29. Выбор поля для сортировки

- В области **Группировка, сортировка и итоги** нажмите кнопку **Добавить группировку** и в открывшемся списке выберите поле **ФИО**. После группировки в конструкторе появятся новые разделы, такие как **Заголовок группы ФИО**. Используя элементы управления вкладки Конструктор, добавьте в этот раздел надпись **Заказчики**.

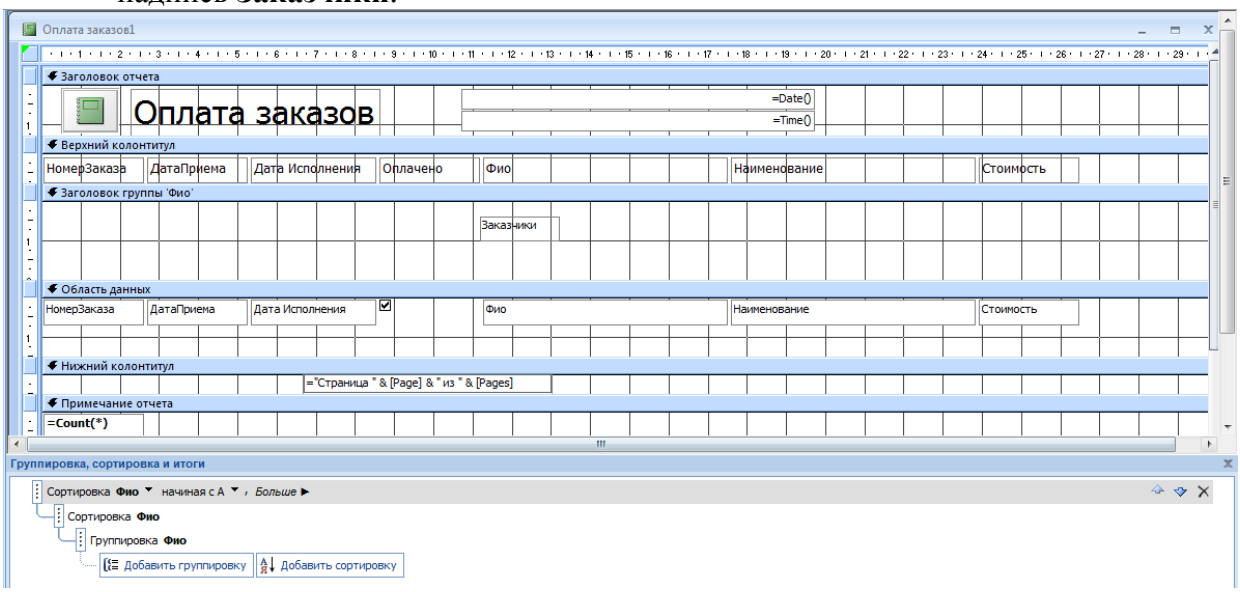


Рисунок 30. Отчет в режиме конструктора

- После сохранения всех изменений этот отчет будет выглядеть так

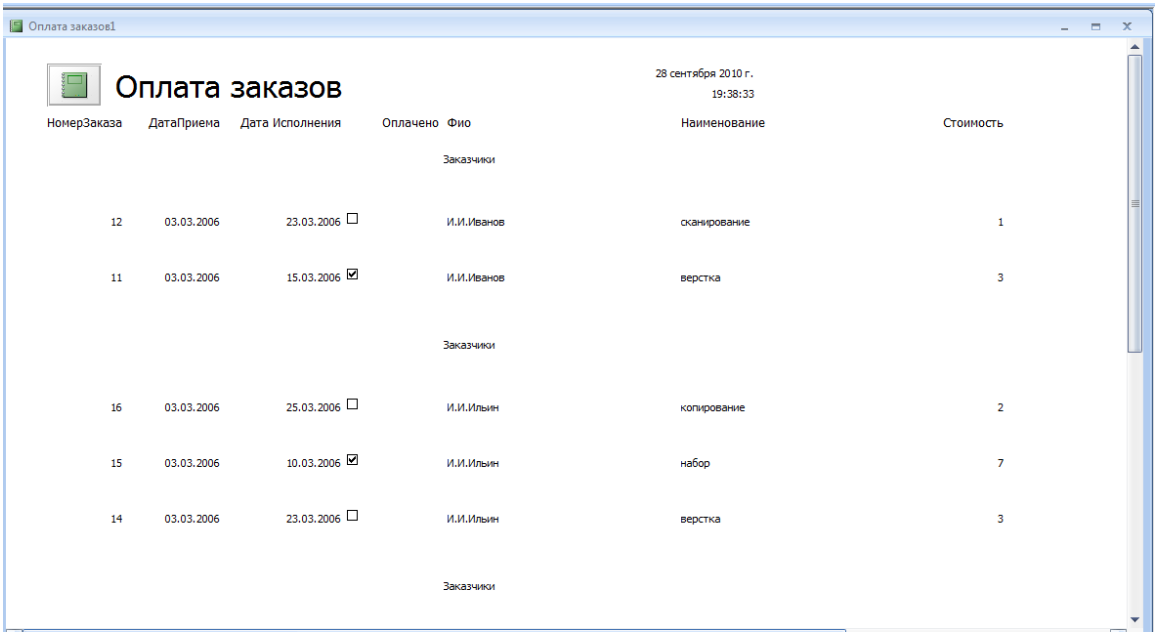


Рисунок 31. Отчет после группировки и сортировки

3. Добавьте в отчет два рассчитываемых поля для получения итоговых значений:
   - Откройте отчет в режиме конструктора.
   - Используя элементы управления вкладки Конструктор разместите в области **Заголовок группы** новое поле, озаглавив его **Всего заказов.**
   - Правой кнопкой мыши щелкните по самому полю и выберите в контекстном меню строку **Свойства.**
   - В диалоговом окне свойств поля выберите вкладку **Данные** и в строке **Данные** щелкните кнопку **…** для вызова построителя выражений.
   - Двойным щелчком по значку + раскройте папку **Функции** и вложенную в нее папку **Встроенные функции**.
   - Выберите в расположенном справа окне категорию функций **Статистические** и в данной категории выберите функцию **Count**, вставьте ее в выражение двойным щелчком или щелчком по кнопке **Вставить**. После выбора в окне построителя выражений должно появиться выражение **= Count («expr»).**
   - Выражение в скобках нужно заменить на название поля, количество которых будет подсчитываться. Для этого дважды щелкните мышью по строке **expr** (она будет выделена) и раскройте в расположенном ниже окне папку **Оплата заказов**. Для отчета с таким именем в окне справа будут выведены все включенные в него элементы. Найдите элемент **Стоимость** и дважды щелкните по нему. Построитель выражений сформирует строку **=Count([Стоимость]).**
   - Закройте окно построителя выражений.
   - Закройте диалоговое окно свойств поля.
   - Аналогичным образом разместите рядом вычисляемое поле **Общая стоимость,** задав для вычисления данных выражение **=Sum([Стоимость])** (его можно ввести с клавиатуры или построить с помощью построителя выражений). Полученные результаты можно увидеть, если на панели инструментов окна базы данных нажать кнопку **Предварительный просмотр** или **Вид**
   - Закройте окно конструктора отчетов.

4. Внесите в отчет итоговые данные по всем заказам. Для этого снова откройте конструктор отчетов и выполните следующие шаги:
   - Выделите в области **Заголовок группы ФИО** размещенные в этой области вычисляемые поля и скопируйте их с помощью команды **Копировать.**

- Выделите область **Примечание отчета** и выполните вставку полей из буфера.
- Переименуйте первое поле, введя в надпись строку **Итого заказов** вместо **Всего заказов**.
- Разместите новое поле, присвоив ему имя **Оплачено** и задав для вычисления данных выражение

=DSum("[Стоимость]";"[Оплата заказов]";"[Оплата заказов]![Оплачено]")

(суммируется стоимость оплаченных заказов, данные для суммирования выбираются из поля **Стоимость** отчета **Оплата заказов** по полю **Оплачено**).

- Скопируйте это поле и вставьте его ниже, изменив надпись на **Не оплачено** и выражение для вычисления значения соответственно на

=DSum("[Стоимость]";"[Оплата заказов]";" not [Оплата заказов]![Оплачено]")

Готовый отчет представлен на рис. 32, содержащем начало и конец отчета

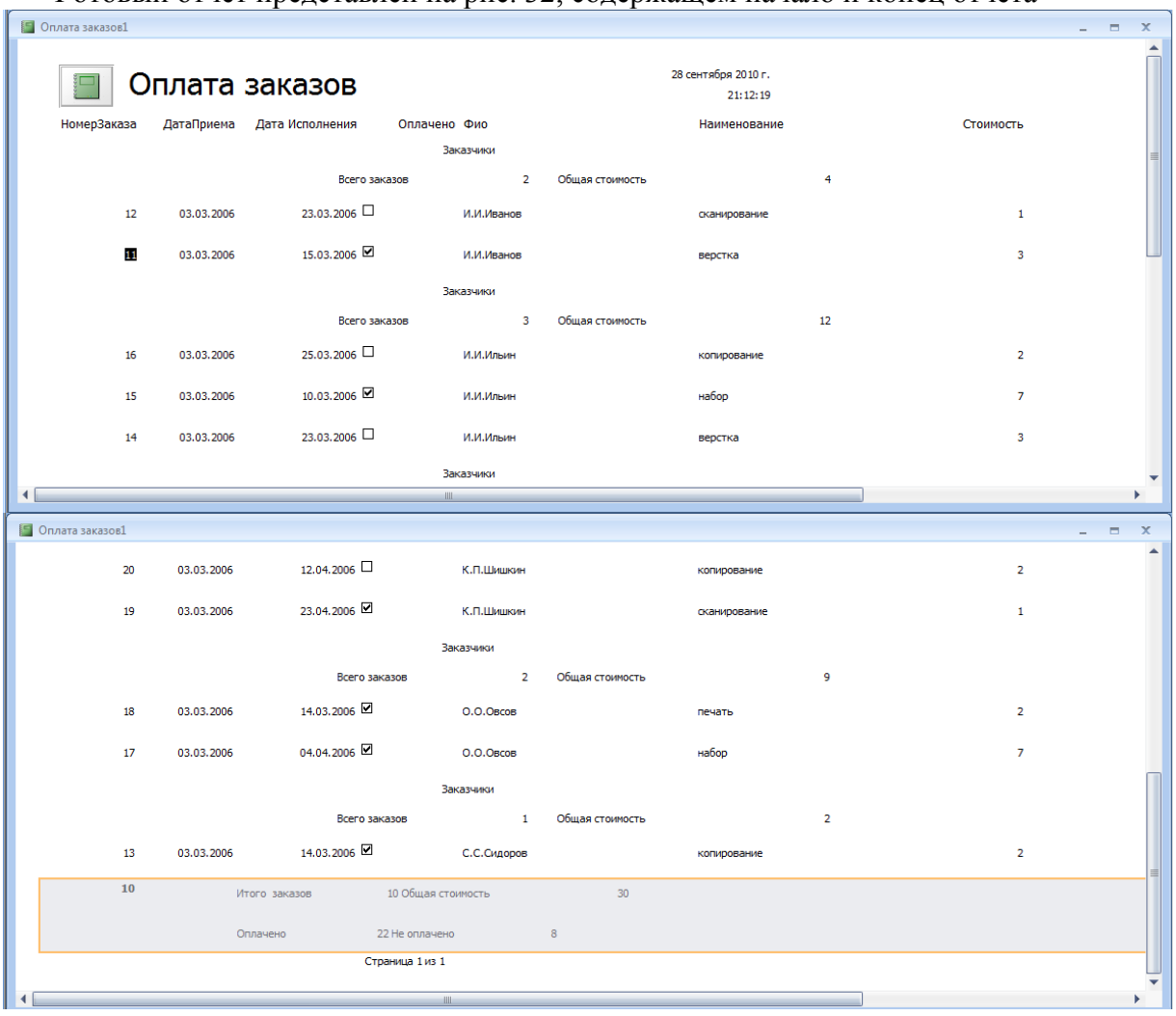


Рисунок 32. Начало и конец отчета

Вопросы для самоконтроля:

1. Как создать и оформить отчет с помощью **Мастера отчетов**?
2. Зачем используется команда **Сортировка и группировка**?
3. Как и зачем используются в отчетах встроенные функции?
4. Составьте запрос и отчет по заказам, выполненным конкретным исполнителем.

Оглавление