

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**  
**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**  
**«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**  
**КАФЕДРА ИНФОРМАТИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ**

**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ**  
**Система управления базами данных**  
**Microsoft Access 2010**

Брест 2014

УДК 681.3

Учебно-методическое пособие содержит основные теоретические сведения по проектированию баз данных, описанию работы в среде Microsoft Access 2010 и сквозной пример создания базы данных.

Предназначено для студентов экономических специальностей очной и заочной форм обучения, магистрантов и аспирантов.

Составители: Быков В.Л., доцент, к.т.н.;  
Гучко И.М., старший преподаватель;  
Кулешова А.М., старший преподаватель

Рецензенты: Котов И.В., доцент кафедры Информатики и компьютерных систем  
УО «БГУ им. А.С. Пушкина», доцент, к.ф.м.н., доцент.  
Савчук В.Ф., доцент кафедры прикладной математики и технологии  
программирования УО «БГУ им. А.С. Пушкина», к.ф.м.н., доцент.

## Содержание

Система управления базами данных Access.....	5
1. Общие сведения.....	5
1.1. Основные понятия и определения.....	5
1.2. Модели представления данных.....	5
1.3. Операции реляционной алгебры.....	9
Примеры операций над отношениями.....	9
Объединение.....	9
Пересечение.....	10
Разность.....	10
Декартово произведение.....	10
Деление.....	10
Проекция.....	10
Соединение.....	11
Выбор.....	11
1.4. Понятия о языке структурированных запросов SQL.....	12
Синтаксис SQL-запроса.....	12
Оператор SELECT.....	13
Создание вычисляемых полей.....	13
Директива FROM.....	14
Использование предикатов.....	14
Директивы JOIN и WHERE.....	14
Директива JOIN.....	14
Директива WHERE.....	15
Критерии фильтрации.....	15
Предикат IN.....	16
Предикат BETWEEN.....	16
Сортировка.....	16
2. Проектирование базы данных.....	17
3. Система управления базами данных ACCESS.....	19
3.1. Назначение и возможности СУБД.....	19
Объекты Access.....	20
3.2. Создание базы данных.....	20
Совместимость с предыдущими версиями баз данных.....	23
3.3. Создание Таблицы.....	23
Создание таблицы в Режиме конструктора таблиц.....	23
Создание первичного ключа.....	26
Свойства полей базы данных.....	27
Создание Схемы данных.....	32
Установление связи между таблицами.....	34
Обеспечение целостности данных.....	34
Каскадное обновление и удаление связанных полей.....	34
Ввод и редактирование данных.....	35

Просмотр базы данных .....	36
Вставка и удаление полей, изменение ключей .....	36
Удаление и переименование таблиц .....	36
Изменение макета таблицы .....	36
Печать таблицы и форм в ACCESS .....	36
Использование баз данных .....	37
Поиск данных .....	37
Сортировка записей в таблицах и формах .....	37
Фильтрация данных .....	38
Создание запросов, форм и отчетов .....	41
Типы запросов .....	41
Создание запросов .....	42
Однотабличный запрос .....	42
Правила включения полей в запрос .....	43
Многотабличный запрос .....	43
Примеры запросов на выборку с условиями .....	44
Параметрические запросы .....	45
Запросы с вычисляемыми полями .....	46
Функции работы с датами .....	49
Логические функции .....	51
Создание запросов на основе других запросов .....	52
Перекрестный запрос .....	58
Запрос на создание таблицы .....	60
Формы .....	61
Быстрое создание форм .....	61
Создание форм с помощью мастера форм .....	63
Создание формы в режиме Конструктора .....	63
Создание отчетов .....	66
Автоматизация вычислений в Access .....	67
Создание макроса .....	67
Использование макроса .....	67
Использование команд управления в макросе .....	68
Создание кнопочной формы вручную .....	68
Установка кнопок .....	69
Создание кнопочной формы с помощью Диспетчера кнопочных форм .....	70
Добавление Диспетчера кнопочных форм на Ленту .....	70
Создание главной кнопочной формы .....	70
Формирование элементов взаимосвязи кнопочных форм .....	71
Создание команд управления для подчинённых форм .....	72
Редактирование кнопочной формы в режиме Конструктора .....	73
Добавление рисунков на кнопочную форму .....	74
Определение параметров запуска приложением .....	75
Список рекомендуемой литературы .....	75

## СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ACCESS

### 1. Общие сведения

#### 1.1. Основные понятия и определения

**Базами данных называют совокупность данных, совместно используемых различными задачами в рамках единой автоматизированной информационной системы.**

Основными понятиями теории баз данных являются:

- **предметная область** – часть реального мира, подлежащая изучению с целью организации управления в этой сфере и последующей автоматизации процесса управления;
- **объект** – элемент информационной системы, сведения о котором хранятся в базе данных. Иногда объект называют *сущностью*;
- **атрибут** – это информационное отображение свойств объекта;
- **ключевой атрибут** – это такой атрибут (или группа атрибутов), который позволяет определить значения других атрибутов;
- **запись данных** – совокупность значений связанных элементов данных;
- **первичный ключ** – это атрибут (или группа атрибутов), который уникальным образом идентифицирует каждый экземпляр объекта (запись).

Процедуры хранения данных в базе должны подчиняться некоторым общим принципам, среди которых в первую очередь можно выделить следующие:

- **целостность и непротиворечивость данных**, под которым понимается как физическая сохранность данных, так и предотвращение неверного использования данных, поддержка допустимых сочетаний их значений, защита от структурных искажений и несанкционированного доступа;
- **минимальная избыточность данных** обозначает, что любой элемент данных должен храниться в базе данных в единственном виде, что позволяет избежать дублирования операций, проводимых над ними.

Программное обеспечение, осуществляющее операции над базами данных, получило название «система управления базами данных» (СУБД).

- Основными задачами СУБД являются:
- обеспечение хранения и использования данных;
- обеспечение пользователей языковыми средствами (прикладные программы) описания и манипулирования данными;
- обеспечение поддержки логических моделей данных. Модель данных определяет логическое представление физических данных;
- обеспечение операций создания и манипулирования логическими данными (выбор, вставка, обновление, удаление и т.п.) и одновременное отображение (выполнение) этих операций над физическими данными;
- обеспечение защиты и целостности (согласованности) данных при их коллективном использовании.

#### 1.2. Модели представления данных

При разработке баз данных разрабатывается информационная и физическая модель данных.

**Информационная модель** должна отображать предметную область в терминах понятных и привычных для пользователя. Обычно это информация об интересующих его фактах, явлениях, предметах, событиях, об их свойствах и связях между ними. На основе этой информационной модели разрабатывается внутренняя схема базы данных с использованием языка программирования соответствующей СУБД.

**Физическая модель данных** (в дальнейшем – модель данных) определяется составом объектов, входящих в базу данных, и связями между ними.

Модель данных, которую поддерживает СУБД на логическом уровне, определяется тремя компонентами:

- допустимой структурой данных, разнообразием и количеством типов объектов, которые можно описать с помощью языковых средств СУБД;
- множеством допустимых операций над данными;
- ограничениями для контроля целостности: это логические ограничения, накладываемые на данные, для сохранения непротиворечивости данных и обеспечения адекватности отображения предметной области в базе данных.

Возможны три модели баз данных: сетевые, иерархические и реляционные. Сетевые и иерархические СУБД получили наибольшее распространение на больших- и мини-ЭВМ. На персональных компьютерах (ПК) используется преимущественно реляционная модель данных.

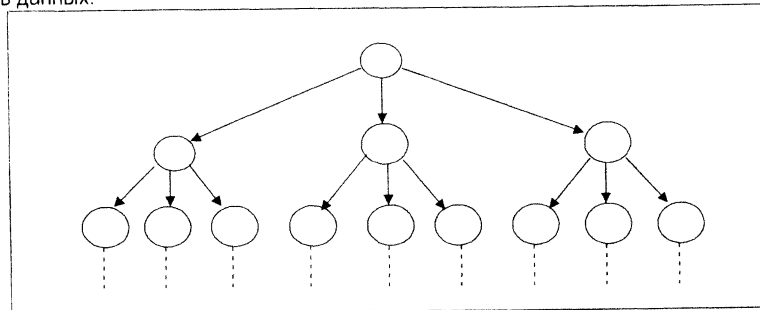


Рисунок 1.1 – Схема иерархической модели данных

**Иерархических СУБД** данных представляют собой древовидную структуру (рис. 1.1). В этой модели все записи, агрегаты и атрибуты образуют иерархически организованный набор, то есть такую структуру, в которой все элементы связаны отношением подчиненности, и при этом любой элемент может подчиняться только одному какому-нибудь элементу. Иерархические модели данных базируются на понятии “ветвь”. В качестве примера иерархической модели данных можно привести различные классификаторы растительного и животного мира: царства, типы, класс, семейство, вид, подвид и т.д.; родословная (генеалогическое древо). Классификаторы в науке, технике, экономике, управлении, например, классификация автомобилей, документов, рынка сбыта, Государственных стандартов и т.д.

**Сетевые СУБД** используют модель представления данных в виде произвольного графа (рис. 1.2). Сетевой подход является развитием иерархической модели. В сетевой модели запись-потомок может зависеть не от одного прародителя, а от многих, напри-

мер, сетевая модель построения сетей ЭВМ, в которой имеется Центральный вычислительный комплекс, вычислительные центры нижнего уровня и вычислительные центры нижнего уровня обмениваются данными друг с другом.

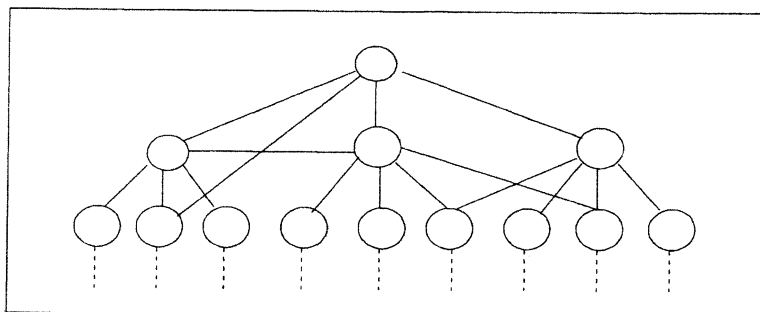


Рисунок 1.2 – Схема сетевой модели данных

**Реляционная модель** ориентирована на представление данных в виде таблицы. Таблица реляционной БД (рис. 1.3) представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – это один элемент данных, повторяющиеся группы отсутствуют;
- все столбцы (колонки) в таблице однородные. Это означает, что все элементы одного столбца имеют одинаковую природу. Например: марка автомобиля или размер заработной платы;
- столбцам присвоены уникальные имена;
- в таблице нет одинаковых строк;
- в операциях с таблицей ее строки и столбцы могут просматриваться в любом порядке и в любой последовательности безотносительно к их информационному содержанию и смыслу.

Системы управления базами данных (СУБД) – один из классов программных средств. Они предназначены для создания, ведения и использования баз данных, справочных, информационно-поисковых систем. Основными компонентами информационной системы являются: база данных (БД), система управления базой данных, прикладная программа и интерфейс (рис. 1.3). База данных содержит интересующую пользователя информацию, а также описание структуры хранимых данных. СУБД выполняет типовые процедуры управления данными, осуществляет взаимодействие с прикладной программой. Прикладная программа реализует требуемый алгоритм ведения диалога пользователя с информационной системой, ввода и контроля запросов, организации информационного поиска, выборки и представления данных в виде справок и отчетов. Взаимодействие между прикладной программой и СУБД осуществляется с помощью специальных операторов или команд языка управления базой данных. Интерфейс, то есть система взаимодействия пользователя с программой, обеспечивается средствами операционной системы персонального компьютера (ПК).

Логическую и физическую структуру БД поясняет рис. 1.3.

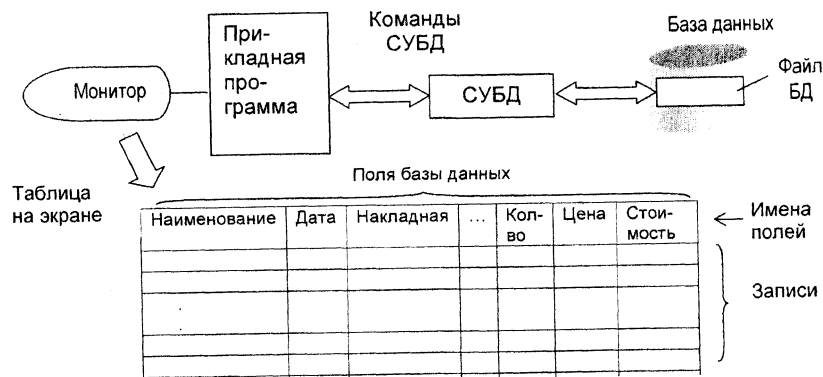


Рисунок 1.3 – Логическая и физическая структура СУБД

Таблица базы данных на экране монитора соответствует файлу данных на диске, строка таблицы – запись файла, колонка таблицы – полю записи. На экране монитора может отображаться не вся БД, а только ее часть. Размещение в одной строке таблицы определенных элементов данных означает установление между ними связи или отношения. Например, если база данных содержит сведения о запасных частях к автомобилям, то в одной строке могут быть помещены сведения о запасных частях к автомобилю конкретной марки. То есть, данные в одной строке связаны между собой тем, что принадлежат одной марке автомобиля. Строка таблицы с этими данными представляет собой один конкретный экземпляр отношения данного типа или его *кортеж*, а всю таблицу в целом называют *отношением*. Таким образом, при описании реляционной модели данных отношением называют всю таблицу в целом как совокупность конкретных экземпляров отношения. Слова "отношения" и "реляционный" (от латинского relation – отношение) представляют собой синонимы.

Совокупность значений элементов данных, размещенных в одном столбце таблицы и определяющих некоторую характеристику или свойство объектов, описываемых строками таблицы, называют атрибутом отношения. Количество элементов данных в кортеже (количество столбцов в таблице) определяет *степень* отношения. Если таблица включает  $n$  столбцов, то она представляет собой отношение степени  $n$ . Количество кортежей в отношении (число строк в таблице) определяют его *мощность* –  $m$ . Тогда общее количество элементов данных в отношении степени  $n$  будет равно  $n \times m$ .

Атрибут, значение которого идентифицирует кортеж, то есть позволяет однозначно выделить его из других кортежей данного отношения, называется *ключевым атрибутом* или просто *ключом*. Ключ может включать несколько атрибутов – *составной ключ* или представлять собой только часть значения атрибута – *частичный ключ*. В приведенном выше примере в качестве ключа может быть номер автомобиля, что позволяет однозначно выделить кортеж из всего отношения. Часто степень отношения БД может быть достаточно большой. Пользоваться такой базой данных неудобно, поэтому она может быть разбита на несколько самостоятельных частей – несколько отношений. Деление одной большой базы данных на несколько необходимо также с целью исключения



повторяющихся записей. Эта операция называется *нормализацией*. В таком случае возникает потребность установления логической связи между различными кортежами, обеспечивающей непротиворечивость и полноту базы данных. Логическая связь между кортежами разных отношений в целях их совместной обработки реализуется различными способами:

- а) использованием в отношениях одинаковых ключей;
- б) включение в отношение *внешнего ключа*;
- в) использование *связующего отношения*.

В нормализованной базе данных между таблицами могут устанавливаться несколько видов отношений:

- “Один к одному” – (1 : 1). Данный вид отношения предполагает, что каждой записи одного отношения соответствует одна запись другого отношения;
- “Один ко многим” – (1 : M) или (1 : ∞). В этом случае каждой записи первого отношения соответствует множество записей другого отношения.

Связь двух таблиц, находящихся в отношении “один ко многим”, устанавливается по **уникальному ключу**, входящему в **главную таблицу**, которая представляет в отношении сторону “один”.

Вторая таблица, представляющая в отношении сторону “многие”, называется **подчинённой** таблицей. В подчинённой таблице используются **составные ключи**, которые включают первичный ключ и один или несколько ключей связи. Составной ключ в подчинённой таблице обеспечивает, с одной стороны, его **уникальность**, а с другой стороны – **связь** с подчинёнными таблицами. Часть составного ключа, которая не участвует в связи с другими таблицами, называется **первичным ключом**, другая часть ключа, обеспечивающая связь с другими таблицами, называется **ключом связи**. Ключ связи в этой таблице может быть частью уникального ключа, либо не входит в состав уникального ключа. Ключ связи в подчинённой таблице называют также **внешним ключом**.

Основными операциями над отношениями реляционной базы данных являются **объединение** отношений, их **пересечение**, **декартово произведение**, **проекция**, **деление**, **соединение** и **выбор**.

Для выполнения операций над отношениями реляционной базы данных используются языки **реляционной алгебры** и языки **реляционного исчисления**.

### 1.3. Операции реляционной алгебры

Языки реляционной алгебры основаны на реляционной алгебре (алгебра Кодда). Записывая последовательно операции над отношениями в соответствующем порядке, можно получить желаемый результат.

#### Примеры операций над отношениями

##### Объединение

Операция выполняется над двумя **совместимыми** отношениями (отношения совместимы, если они имеют одинаковую степень и одинаковые типы соответствующих атрибутов). Результат объединения включает все кортежи первого отношения и недостающие кортежи из второго отношения.

Пусть имеется две таблицы: Клиент 1 и Клиент 2. Тогда итоговая таблица, включающая всех клиентов, есть результат операции объединения.

Клиент 1		Клиент 2		Клиент (результатирующее отношение)	
Фамилия	Возраст	Фамилия	Возраст	Фамилия	Возраст
Иванов	20	Иванов	20	Иванов	20
Петров	23	Тихонов	35	Петров	23
Тихонов	35	Достоевский	30	Тихонов	35
Тихонов	49			Тихонов	49
				Достоевский	30

### Пересечение

Результат пересечения содержит только те кортежи первого отношения, которые есть во втором отношении.

Клиент (результатирующее отношение. Пересечение)	
Фамилия	Возраст
Иванов	20
Тихонов	35

### Разность

Результат включает только те кортежи первого отношения, которых нет в первом отношении

### Декартово произведение

Здесь отношения-операнды могут иметь разные схемы. Степень результирующего отношения равна сумме степеней отношений операндов, а мощность – произведению их мощностей.

Клиент (результатирующее отношение. Разность)	
Фамилия	Возраст
Петров	23
Тихонов	49

Таблица 1	Таблица 2		Декартово произведение (результатирующее отношение)		
Фамилия	Предмет	Дата	Фамилия	Предмет	Дата
Иванов	Информ.	7.06.05	Иванов	Информ.	7.06.05
Петров	Математ.	10.06.05	Петров	Информ.	7.06.05
Тихонов			Тихонов	Информ.	7.06.05
			Иванов	Математ.	10.06.05
			Петров	Математ.	10.06.05
			Тихонов	Математ.	10.06.05

### Деление

Отношение-делитель должно содержать подмножество атрибутов отношения делимого. Результирующее отношение содержит только те атрибуты делимого, которых нет в делителе. В него включаются только те кортежи, декартовы произведения которых с делителем содержатся в делимом.

Экзаменационная ведомость			Делитель		Результат
Фамилия	Предмет	Оценка	Предмет	Оценка	Фамилия
Иванов	Информ.	4	Информ.	4	Иванов
Иванов	Математ.	5	Математ.	5	Петров
Петров	Информ.	5			Тихонов
Петров	Математ.	5			
Тихонов	Информ.	4			
Тихонов	Математ.	3			

### Проекция

Эта операция выполняется над одним отношением на некоторые атрибуты. Результирующее отношение включает часть атрибутов исходного, на которые выполняется проекция. Например, "Номер отдела" и "Должность". Кортежи дубликаты отсутствуют.

Сотрудники		
Фамилия	Номер отдела	Должность
Иванов	1	Инженер
Петров	1	Инженер
Поздняков	2	Инженер
Хачатурян	2	Слесарь
Орбенин	3	Механик
Гроцилин	3	Слесарь
Кожубей	3	Слесарь

Должности	
Номер отдела	Должность
1	Инженер
2	Инженер
2	Слесарь
3	Механик
3	Слесарь

### Соединение

Соединение выполняется над двумя отношениями. В каждом отношении выделяется атрибут, по которому будет выполняться соединение.

Группы	
Специальность	Код студента
Механик	1
Программист	4
Технолог	6

Студенты		
Код студента	Фамилия	Курс
1	Давыдов	1
2	Цыбуля	2
3	Цаплин	2
4	Воробьев	1
6	Зайцев	3

В качестве атрибута для соединения выберем Код студента. Результирующее отношение содержит все атрибуты первого и второго отношения.

Группы + Студенты			
Специальность	Код студента	Фамилия	Курс
Механик	1	Давыдов	1
Программист	4	Воробьев	1
Технолог	6	Зайцев	3

### Выбор

Операция выполняется над одним отношением. Результирующее отношение содержит подмножество кортежей, выбранных по некоторому условию, например "возраст" >= 18 лет.

Студент	
Фамилия	Возраст
Петров	16
Волков	18
Перепелица	17
Дубинин	21
Крылов	18

Студент-выбор	
Фамилия	Возраст
Волков	18
Дубинин	21
Крылов	18

Для выполнения запросов на выборку информации необходимо иметь возможность извлекать подмножество столбцов таблицы, а также объединять столбцы разных таблиц, создавая новые отношения большей или меньшей степени, чем исходные, с включением в результирующие (выходные) отношения кортежей, удовлетворяющих определенным условиям. Система управления реляционной базой данных должна выполнять подобные операции "разрезания" и "склеивания" таблиц и отбора информации, удовлетворяющим условиям поиска. Современные базы данных позволяют выполнять данные

операции и, кроме того, обеспечивают вывод на экран или печать результатов вычислений в графической форме. Эффективность конкретной системы управления реляционной базой данных определяется наличием и удобством использования средств реализации указанных выше операций.

Языки реляционного исчисления основаны на классическом исчислении предикатов. Они предоставляют пользователю набор правил для записи запросов к базе данных. В таком запросе содержится лишь информация о желаемом результате. На основании запросов СУБД автоматически, путем формирования новых отношений, выдаст результат. В реляционных базах данных большое распространение получил язык структурированных запросов SQL.

#### 1.4. Понятия о языке структурированных запросов SQL

С 1986 года язык структурированных запросов де-факто был признан в качестве стандарта языка для работы с реляционными базами данных.

Язык структурированных запросов SQL представляет собой набор программных команд, позволяющих пользователю получать информацию из баз данных, редактировать, добавлять или удалять данные, выполнять некоторые вычислительные операции с извлеченным набором данных и выполнять некоторые другие функции.

В составе SQL могут быть выделены следующие группы инструкций:

- язык описания данных – DDL;
- язык манипулирования данными – DML;
- язык управления транзакциями.

Инструкции DDL предназначены для создания, изменения и удаления объектов базы данных (CREATE – создание объектов, DROP – удаление объектов, ALTER – изменение объектов).

Инструкции DML позволяют выбирать данные их таблицы, а также добавлять, удалять и изменять их (SELECT – выбрать, INSERT – добавить, UPDATE – изменить значение записей и полей, DELETE – удаление записей).

Инструкции транзакции обеспечивают правильность выполнения операций обработки данных. Все действия, составляющие транзакцию, должны либо выполняться полностью, либо вообще не выполняться (COMMIT – фиксация в базе данных всех изменений, сделанных текущей транзакцией, SAVEPOINT – установка точки сохранения (начала транзакции), ROLLBACK – откат изменений, сделанных с момента транзакции).

SQL-оператор состоит из трех составных частей:

**Объявления параметров** – необязательные параметры, которые передаются в SQL-операторы программой.

**Управляющий оператор** – сообщает ядру обработки запросов тип операции.

**Опциональные объявления** – передают ядру обработки запросов информацию о сортировке, фильтрации и другие параметры, которые необходимо применить к обрабатываемым данным.

#### Синтаксис SQL-запроса

[Объявления параметров] Управляющий оператор [Опциональные объявления]

Наибольший интерес для пользователя представляют управляющие операторы и операционные объявления.

### Оператор SELECT

Основным оператором языка запроса является оператор SELECT. Синтаксис этого оператора:

```
SELECT <поля> FROM <таблица> [WHERE <параметр>]
```

В структуре оператора три ключевых слова:

**SELECT** – указывает, какие поля должны быть запрошены после запроса;

**FROM** – определяет, из каких таблиц будут извлекаться записи, указанные в блоке SELECT;

**WHERE** – необязательный блок, указывает условия выборки, сортировки, фильтрации, группировки записей.

Параметр *поля* блока SELECT используется для указания полей, которые должны быть включены в результирующий набор записей. Можно включать поля из нескольких таблиц и даже поля, вычисляемые из других полей таблиц. В списке полей имена полей разделяются запятыми. Если имя поля содержит пробелы, его следует заключать в квадратные скобки. Для указания имен полей может использоваться маска – символ "\*", который определяет все поля базы данных.

Простейший вид оператора запроса:

```
SELECT * FROM Водители
```

Данным запросом выбираются все поля из таблицы Водители.

```
SELECT Фамилия, Имя, Отчество, [Год рождения] FROM Водители
```

При выборе полей из нескольких таблиц, перед именем поля указывается имя таблицы и разделитель – точка. В блоке FROM при этом указываются имена таблиц, разделенные запятой, а в блоке WHERE или условии JOIN указываются отношения между таблицами. Отношения между таблицами устанавливаются по ключевым полям.

```
SELECT " Водители.Фамилия, Работа.[Время выполнения] FROM Водители, Работа WHERE Водители.[Табельный номер] = Работа.[Табельный номер]"
```

В приведенном примере связываются две таблицы: Водители и Работа, связь между таблицами осуществляется по полям *Табельный номер*.

### Создание вычисляемых полей

В SQL-запросе имеется возможность создать вычисляемое поле. Если вычисляемому полю не присваивать имя, то программа автоматически присвоит ему имя, например Expr1001 (Выражение1) – для расчета первого поля. Для числовых полей можно применять арифметические операторы +, -, \*, / и функции языка VB 6.0. Для текстовых полей можно применять операции конкатенации и функции обработки строковых переменных.

Примеры вычисляемых полей:

1. Вычисление оплаты за работу. Создается новое поле, новому полю присваивается имя Оплата:

```
SELECT Работа.[Код водителя], Работа.[Время выполнения (час)], [Виды работ].[Оплата за час], [Время выполнения (час)]*[Оплата за час] AS Оплата
```

```
FROM [Виды работ] INNER JOIN Работа ON [Виды работ].[Код работы] = Работа.[Код работы].
```

2. Объединение трех полей таблицы: Фамилии, Имени и Отчества в одно поле:

```
SELECT Водители.Фамилия, Водители.Имя, Водители.Отчество, [Фамилия] & [Имя] & [Отчество] AS Выражение1 FROM Водители.
```

Полю можно присвоить имя. Для этого после выражения необходимо указать ключевое слово (директиву) As и указать имя поля:

```
SELECT Водители.Фамилия, Водители.Имя, Водители.Отчество, [Фамилия] & [Имя] & [Отчество] AS ФИО FROM Водители.
```

3. Пример вычисления стоимости топлива за пробег.

```
SELECT Работа.Расстояние, [Транспортное средство].[Пробег в км], ГСМ.[Марка топлива], ГСМ.[Цена за литр]
```

```
FROM (ГСМ INNER JOIN [Транспортное средство] ON ГСМ.[Код топлива] = [Транспортное средство].[Код топлива]) INNER JOIN (Водители INNER JOIN Работа ON Водители.[Табельный номер] = Работа.[Код водителя]) ON [Транспортное средство].Номер = Водители.[Номер транспортного средства].
```

#### **Директива FROM**

Директива From служит для указания источника данных. Синтаксис этой директивы имеет следующий вид:

```
FROM таблица1 {[In База_данных_1] [As псевдоним], таблица2 [In База_данных_2] [As псевдоним]}
```

В простейшем виде директива FROM имеет вид: *FROM таблица*

Директива позволяет указать две базы данных. Имя базы данных, в которой находится таблица, вводится после служебного слова *IN*. Псевдоним заменяет имя базы данных, для сокращения длины текста при обращении к соответствующей таблице. Например:

```
FROM Водители IN Автобаза As BD1, [Год приёма на работу] IN Кадры As BD2.
```

```
SELECT BD1.Фамилия, BD2.[Год приёма на работу] FROM Водители IN Автобаза As BD1, [Год приёма на работу] IN Кадры As BD2
```

#### **Использование предикатов**

Предикаты используются в операторе SELECT для управления выборкой данных. В качестве предикатов используются ключевые поля ALL – все, DISTINCT – уникальный и DISTINCTROW – уникальная строка.

Данные предикаты используются при выборке записей по критериям, например:

```
SELECT All * FROM Водители
```

Предикат ALL используется по умолчанию, поэтому может быть опущен.

```
SELECT DISTINCT Фамилия, Имя, Отчество FROM Водители
```

В данном примере будут отбираться только те записи, в которых поля, указанные в запросе, не совпадают.

```
SELECT DISTINCTROW Фамилия, Имя, Отчество FROM Водители
```

В этом примере на совпадение будут проверяться все поля записи, даже если они не указаны в запросе.

#### **Директивы JOIN и WHERE**

Директивы JOIN и WHERE позволяют устанавливать связи между таблицами и условия выборки.

#### **Директива JOIN**

Директива JOIN позволяет связать две таблицы. Синтаксис директивы:

```
таблица1 {INNER| LEFT| RIGHT} JOIN таблица2 ON таблица1.поле=таблица2.поле
```

Опция ON содержит условие выборки – совпадение значений полей в таблицах 1 и 2. Таблица1 называется левой, а таблица2 – правой. В логическом выражении можно использовать любые операторы сравнения (<, >, <=, >=, <>).

Директива JOIN имеет три варианта выполнения: *INNER*, *LEFT* и *RIGHT*.

При использовании опции *INNER* в результирующую таблицу будут помещены записи из обеих таблиц, удовлетворяющие заданному условию в опции *ON*. При использовании опции *LEFT* в результирующую таблицу будут помещены все записи из таблицы1 (левой) и те записи из таблицы2, которые удовлетворяют логическому условию. При использовании опции *RIGHT* в результирующую таблицу будут помещены все записи из таблицы2 (правой) и те записи из таблицы1, которые удовлетворяют логическому условию.

```
SELECT Водители.Фамилия, Работа.[Код работы]
FROM Водители INNER JOIN Работа ON Водители.[Табельный номер] = Рабо-
та.[Код водителя].
```

#### Директива WHERE

Директива *WHERE* также может использоваться для связи таблиц, функционирует аналогично директиве *INNER JOIN*. Кроме этого, данная директива используется для поиска, фильтрации и сортировки извлекаемых данных.

```
SELECT Работа.Дата, Водители.Фамилия, Работа.Расстояние
FROM Водители INNER JOIN Работа ON Водители.[Табельный номер] = Рабо-
та.[Код водителя]
```

```
WHERE (((Работа.Расстояние) Between 10 And 100))
```

```
ORDER BY Работа.Дата, Водители.Фамилия – (сортировка по полям Дата базы
данных Работа и Фамилия базы данных Водители).
```

#### Критерии фильтрации

Для выборки данных в SQL-запросе может использоваться фильтр. Для этой цели в директиве *WHERE* используются предикаты *Comparison*, *LIKE*, *IN*, *BETWEEN*.

Предикат *Comparison* предназначен для сравнения значения поля с указанной величиной:

```
SELECT * FROM Водители WHERE Фамилия='Петров'
```

```
SELECT * FROM Водители WHERE Физика>=4
```

```
SELECT * FROM Водители WHERE Дата_Рождения>#01/06/85#
```

В качестве операторов в условном выражении могут использоваться следующие операторы: <, >, <=, >=, =, <> – не равно. Текстовые значения в условном выражении заключаются в апострофы, значения типа дата заключаются в символы "#".

Предикат *LIKE* действует так же, как предикат *Comparison*, но обладает большей гибкостью в задании условий сравнения, в частности он позволяет использовать маски. В предикате *LIKE* могут использоваться списки/диапазоны символов. К оформлению списков предъявляется ряд требований:

- список должен заключаться в квадратные скобки;
- первый и последний символы диапазона должны разделяться дефисом (знак "минус");
- диапазон символов должен указываться в возрастающем порядке, например, а – я, но не я – а.

Используемые шаблоны приведены в табл. 1.1.

Таблица 1.1 – Шаблоны, используемые в предикате LIKE

Элемент шаблона	Назначение	Пример шаблона	Пример результата
*	Заменяет все слово или его часть	П*	Петров, Перепелица, Постолюк
?	Заменяет один символ	Петров?	Петров, Петрова
#	Знакоместо для цифры	200#	2000, 2001, 2002 ...
[список]	Одиночный символ из списка	[c-f]	d, e, f
[!список]	Одиночный символ не из списка	[!c-f]	a, b, g, h, ...
комбинация	В зависимости от вида шаблона	a?l*	art, antique, artist

Если в качестве значения используется переменная, то используется следующий синтаксис:

*WHERE Фамилия LIKE ' " & Name & "'*  
*WHERE Фамилия LIKE ' " & Name & "%'*

Символ % означает удаление хвостовых пробелов.

*Парные кавычки и апостроф, указанные в конце выражения, следуют без пробелов*

#### Предикат IN

Предикат IN позволяет проверить совпадение в значении поля одного или нескольких символов.

*WHERE Фамилия IN ('ов', 'ич')*

#### Предикат BETWEEN

Предикат BETWEEN проверяет попадание значения в определенный диапазон. Он может применяться для сравнения строк, чисел и дат. При сравнении граничные значения диапазона также попадают в выборку. В синтаксисе выражения может использоваться оператор NOT, который позволяет выбрать записи, не попадающие в указанный диапазон.

Примеры:

*SELECT \* FROM Водители WHERE Фамилия BETWEEN 'М' AND 'С'*

*SELECT \* FROM Водители WHERE Физика BETWEEN 3 AND 5*

*SELECT \* FROM Водители WHERE Дата\_Рождения BETWEEN Дата\_Рождения #01/06/85# AND #01/12/85#*

*SELECT \* FROM Водители WHERE Фамилия NOT BETWEEN 'М' AND 'С'*

#### Сортировка

В операторе SELECT допускается использовать сортировку возвращаемых записей. Сортировку можно проводить по одному или нескольким полям. Для этой цели используется директива ORDER BY. Для изменения порядка сортировки используется опция DESC после имени поля:

*SELECT \* FROM Водители ORDER BY Фамилия*

*SELECT \* FROM Водители ORDER BY Цех, Фамилия*

*SELECT \* FROM Водители ORDER BY Цех DESC, Фамилия*

В первом примере выполняется сортировка по одному полю, во втором примере сортировка выполняется по двум полям: номеру цеха и фамилии, в третьем примере сортировка также выполняется по двум полям, причем по номеру цеха сортировка выполняется в обратном порядке, а по фамилии в прямом порядке.



## 2. Проектирование базы данных

При определении исходной структуры реляционной базы данных необходимо решить ряд вопросов: сколько и каких отношений должна включать база данных; какова степень и состав каждого отношения; какие атрибуты отношения используются в качестве ключей; как устанавливается связь между разными отношениями; как обеспечить непротиворечивость, согласованность и полноту информации, хранящейся в базе данных. Решение этих вопросов относится к проектированию базы данных информационной системы.

При проектировании базы данных необходимо придерживаться следующей схемы:

1. Выполнить описание предметной области, определить задачи, которые должна выполнять база данных. При описании предметной области необходимо определить, какие объекты входят в предметную область, какова связь между этими объектами, источники информации для удовлетворения предполагаемых запросов, определить потребность в обработке данных.

2. На основе описания предметной области определить состав и структуру данных предметной области. Структура данных предметной области определяется информационно-логической моделью.

3. На основе информационной модели определить структуру реляционной базы данных.

Структура реляционной базы данных задается в виде системы взаимосвязанных таблиц (отношений). Отношения БД содержат как структурную, так и семантическую (смысловую) информацию. Структурная информация задается схемой отношений, а семантическая выражается функциональными связями между атрибутами, известными и учитываемыми в схеме.

Состав атрибутов отношений БД должен удовлетворять двум основным требованиям:

- между атрибутами не должно быть нежелательных функциональных зависимостей;
- группировка атрибутов должна обеспечивать минимальное дублирование данных, обеспечивать их обработку и обновление без трудностей.

Удовлетворение этих требований достигается нормализацией отношений БД (в настоящем пособии не рассматривается).

### Рассмотрим структуру учебной Базы данных "Автобаза"

База данных включает следующие таблицы (отношения):

*Водители:* Табельный номер, Фамилия, Имя, Отчество, Дата рождения, Дата приёма на работу, Трудовой стаж, Число детей, Льготы, Номер транспортного средства, Телефон.

*Транспортные средства:* Номер транспортного средства, Марка транспортного средства, Дата выпуска, Дата ввода в эксплуатацию, Дата прохождения техосмотра, Пробег в км, Моточасы (час), Код топлива, Норма расхода топлива.

*ГСМ:* Код топлива, Марка топлива, Цена за литр.

*Работа:* Код работы, Код водителя, Дата, Расстояние, Время выполнения (час).

*Виды работ:* Код работы, Наименование, Оплата за час.

В таком виде использовать данные неудобно, поэтому представим их в виде таблиц.

<b>Водители</b>				
Содержание поля	Тип данных	Размер поля	Формат и др. свойства	Ключевые атрибуты (поля)
Табельный номер	Текстовый	10		Ключ (первичный)
Фамилия	Текстовый	50		
Имя	Текстовый	50		
Отчество	Текстовый	50		
Дата рождения	Дата		Краткий	
Дата приёма на работу	Дата		Краткий	
Трудовой стаж	Числовой	Байт	число дес.зн.-2	
Число детей	Числовой	Байт	число дес.зн.-0	
Льготы	Текстовый	50		
Номер транспортного средства	Текстовый	10	Поле подстановки из табл. Транспортное средство	Ключ (внешний)
Телефон	Текстовый	15	Маска ввода	
<b>Транспортное средство</b>				
Номер	Текстовый	10		Ключ (первичный)
Марка	Текстовый	10		
Дата выпуска	Дата		Краткий	
Дата ввода в эксплуатацию	Дата		Краткий	
Дата техосмотра	Дата		Краткий	
Пробег в км	Числовой	Одинарное	Основной	
Моточасы, час	Числовой	Одинарное	Основной	
Код топлива	Текстовый	10	Поле подстановки из табл. ГСМ	Ключ (внешний)
Норма расхода топлива	Числовой	Одинарное	Основной	
<b>Работа</b>				
Номер по порядку	Счётчик	Длинное целое	Последовательные	Ключ (первичный)
Код работы	Текстовый	10	Поле подстановки из табл. Виды работ	Ключ (внешний)
Код водителя	Текстовый	10	Поле подстановки из табл. Водители	Ключ (внешний)
Дата	Дата		Краткий	
Расстояние	Числовой	Одинарное	Основной	
Время выполнения (час)	Числовой	Одинарное	Основной	
<b>ГСМ</b>				
Код топлива	Текстовый	10		Ключ (уникальный)
Марка топлива	Текстовый	10		
Цена за литр	Числовой	Одинарное	Денежный	
<b>Виды работ</b>				
Код работы	Текстовый	10		Ключ (уникальный)
Наименование	Наименование	50		
Оплата за час	Числовой	Одинарное	Денежный Число десятичных знаков – 0	

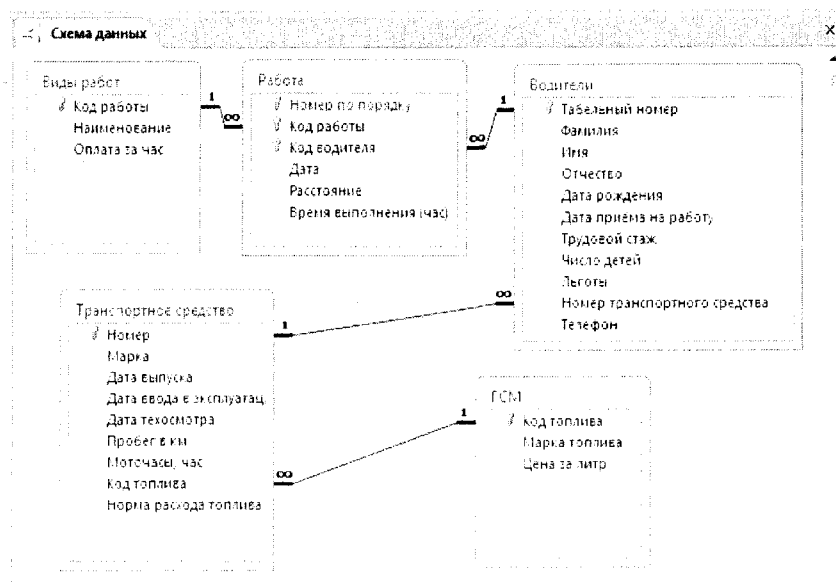


Рисунок 2.1 – Схема данных

После описания структуры атрибутов необходимо также установить ключевые атрибуты. Эти ключевые атрибуты указаны в таблице. Ключевые атрибуты устанавливаются на основе анализа зависимости между отношениями, при описании базы данных.

В отношениях Водители, Транспортное средство, ГСМ, Виды работ используются простые ключи. В отношении Работа используется составной ключ: Номер по порядку, Код работы и Код водителя. В этом отношении потребовалось ввести поле типа счётчик, потому что составной ключ Код работы и Код водителя не обеспечивали уникальность ключа, так как один и тот же водитель мог выполнять одну и ту же работу в разные дни и даже в один день мог выполнить несколько одинаковых работ.

Все таблицы связаны между собой связями **Один ко многим**.

Виды связей между таблицами приведены на рис. 2.1.

### 3. Система управления базами данных ACCESS

#### 3.1. Назначение и возможности СУБД

Система управления базами данных Microsoft Access является реляционной базой данных. Она включает все необходимые инструментальные средства для создания локальной базы данных, общей базы данных в локальной сети с файловым сервером или создания приложений пользователя, работающего с базой данных на SQL-сервере.

Internet-технология позволяет распространять и получать доступ к разнородной информации в глобальных и корпоративных сетях. В Access имеются средства конструирования Web-страниц доступа к данным в базах Access и SQL-серверов. При этом Web-

Browser используется как универсальный интерфейс для доступа и работы с информацией из внешней среды вне зависимости от аппаратно-программной платформы компьютера пользователя и компьютера – источника информации. Для создания, просмотра и работы со страницами доступа к данным Access, а также для работы с этими страницами в Internet и Intranet, нужен современный браузер Microsoft Internet Explorer.

Система доступа к данным в Access 2010 построена на основе ядра базы данных Access Database Engine. Это ядро выполняет загрузку, сохранение и извлечение данных из пользовательских и системных баз данных, обеспечивает высокую производительность и улучшенные сетевые характеристики, поддержку двухбайтового представления символов – Unicode. Ядро базы данных Access 2010 настроено для приложений системы Microsoft Office и обеспечивает интеграцию со службами Microsoft Windows SharePoint Services 3.0 и Microsoft Office Outlook 2010. Указанные особенности ядра базы данных Access 2010 позволяют обмениваться данными через Интернет, вести совместную разработку баз данных удалённым пользователям, используя Веб-страницы. При публикации баз данных службы Access создают сайт SharePoint, содержащий базу данных. Все данные базы данных перемещаются в списки SharePoint на этом сайте. Для работы с этими данными пользователь должен иметь необходимые разрешения в браузере.

#### **Объекты Access**

СУБД Access работает с объектами базы данных, к которым относятся *таблицы, запросы, формы, отчеты*.

**Таблицы** – основной объект базы данных. Они обладают структурой и содержанием и предназначены для хранения данных. Под структурой БД понимают состав полей и их параметры.

**Запросы** – это специальные объекты, предназначенные для выборки данных из взаимосвязанных таблиц и выполнения необходимых вычислений и представления результатов в виде *результатирующих* таблиц. Результатирующие таблицы реально не существуют и после закрытия пропадают. Но, при необходимости, их можно сохранить и создавать на их основе новые таблицы.

**Формы** – объекты, предназначенные для ввода данных в таблицы. Формы можно применять также для выдачи результатов запросов.

**Отчеты** – это объекты, предназначенные для вывода информации на печать. При формировании отчетов можно проводить и вычисления.

Кроме перечисленных объектов, в MS Access могут создаваться макросы и модули.

**Макросы** – программы, состоящие из последовательности макрокоманд, которые выполняются по вызову или при наступлении некоторого события в объекте приложения или его элементе управления. Макросы позволяют автоматизировать некоторые действия в приложении пользователя. Access 2010 содержит большое число встроенных макросов, кроме того, пользователь может создавать пользовательские макросы.

**Модули** – содержат процедуры и функции на языке Visual Basic for Applications, которые разрабатывает пользователь для реализации нестандартных функций в приложении пользователя или процедуры для обработки событий.

### **3.2. Создание базы данных**

Для запуска программы найдите на панели быстрого запуска или на рабочем столе ярлык MS Access и щёлкните по нему мышкой. При отсутствии ярлыка на рабочем столе

найдите имя программы в главном меню: **Пуск, Программы, Microsoft Office, Microsoft Access 2010** – и щёлкните по нему мышкой.

После запуска программы Access на экране появляется рабочее окно программы с активной вкладкой **Файл** (рис. 3.1) – Окно представления (Backstage). В этом окне имеется возможность создать новую базу данных или открыть существующую базу данных. По умолчанию предлагаются имена последних четырёх файлов, с которыми работал пользователь. Однако можно выбрать команду **Последние**, тогда откроется окно диалога "**Последние базы данных**", в котором можно будет выбрать существующую базу данных из списка. В нижней части окна диалога "**Последние базы данных**" можно изменить число последних файлов, предлагаемых в списке при открытии базы данных, либо вообще снять флажок "**Число последних баз данных для быстрого доступа**". По умолчанию предлагается имя базы данных Database1. Расширение имени файла базы данных .accdb. В качестве места хранения файла базы данных предлагается диск C: , однако маршрут сохранения файла базы данных предлагается изменить, щелкнув мышкой по кнопке справа от окна ввода имени файла. В этом случае на экран выводится стандартное окно диалога для сохранения файла новой базы данных (рис. 3.2). Выберите диск, папку, введите имя файла и щелкните по кнопке **ОК** – окно диалога закрывается.

Присвоим нашей базе данных имя "**Автобаза**". Для завершения операции создания новой базы данных щёлкните по кнопке **Создать** в окне диалога **Доступные шаблоны** – создается файл новой базы данных и на экране появляется рабочее окно базы данных (рис. 3.3). База данных будет сохранена с установленным именем.

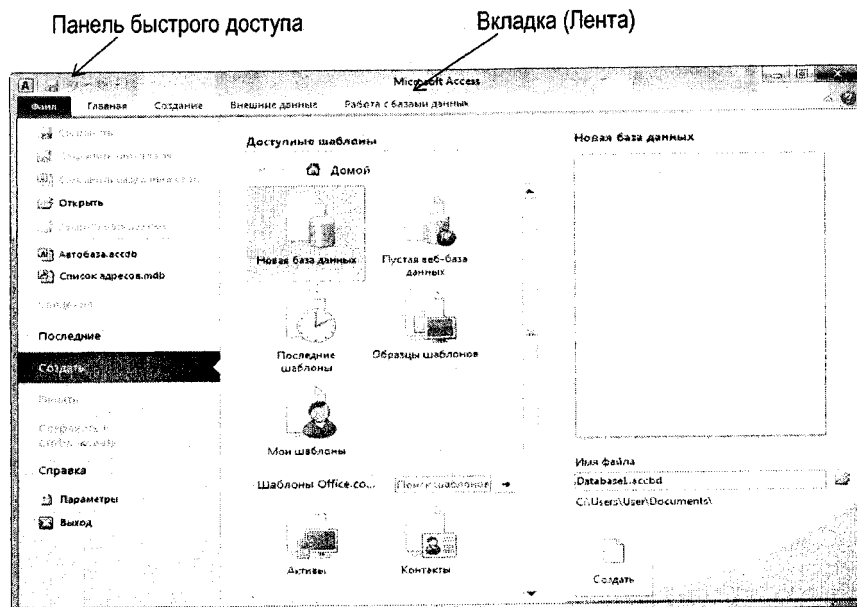


Рисунок 3.1 – Окно программы Microsoft Access 2010

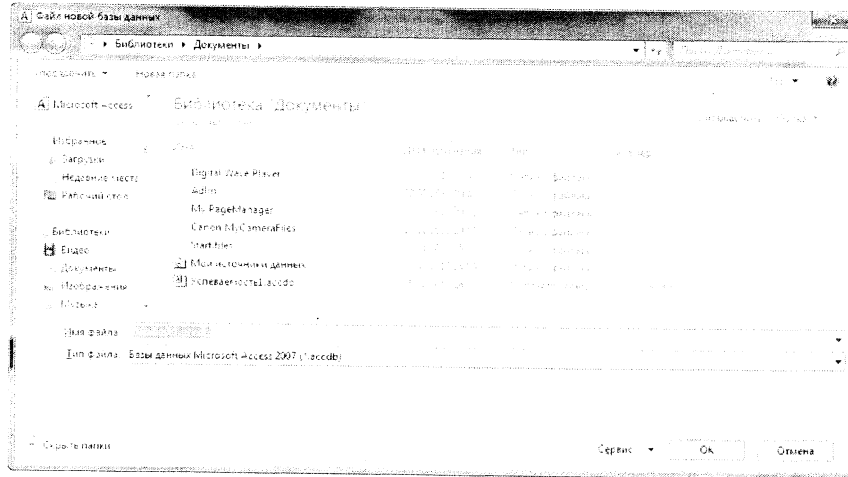


Рисунок 3.2 – Окно сохранения файла новой Базы данных

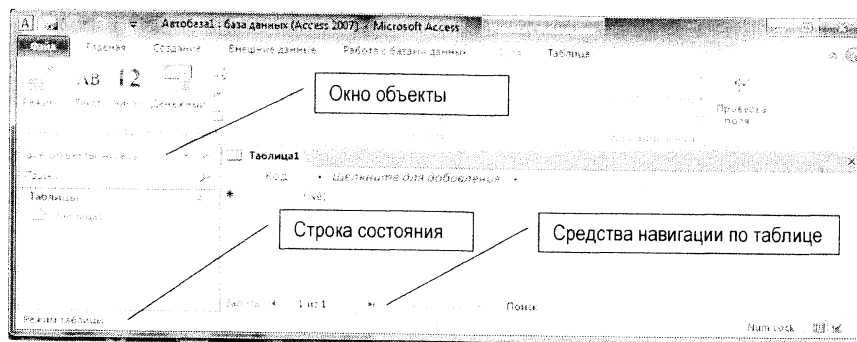


Рисунок 3.3 – Рабочее окно Базы данных

**Примечание:** Microsoft Access сохраняет базу данных сразу при её создании. При закрытии базы данных все данные сохраняются автоматически.

По умолчанию база данных создаётся на основе шаблона *нормальный*. База данных позволяет создавать пользовательские шаблоны или использовать готовые шаблоны, загружаемые из сети Интернет.

В рабочем окне расположены: Панель быстрого доступа, Лента, Окно Объекты. На панели быстрого доступа расположены кнопка системного меню "А", Сохранить – для сохранения базы данных и её объектов, кнопки Отменить и Возвратить, а также открывающийся список настройки панели быстрого доступа.

Лента имеет семь основных вкладок: Предварительный просмотр, Главная, Создание, Внешние данные, Работа с базами данных, Управление версиями, Настройки. С содержанием вкладок познакомимся в процессе работы по созданию и использованию базы данных.

Окно Объекты содержит раскрывающийся список "Все объекты базы данных", открывающий доступ к объектам базы данных. Справа от этого списка расположена кнопка "<<", которая позволяет свернуть или развернуть окно диалога Объекты. Ниже раскрывающегося списка расположена строка Поиск для быстрого поиска требуемого объекта, а ниже нее окно Таблицы, которым также можно управлять с помощью кнопки "↕".

Внизу рабочего окна расположена Строка состояния. В левой части строки состояния отображается текущий режим работы, а справа кнопки Num Lock – для переключения дополнительной клавиатуры на ввод цифр или управляющих команд, Кнопка настройки строки состояния, Кнопка режима конструктора. Могут отображаться и другие кнопки в зависимости от настройки строки состояния.

В окне базы данных по умолчанию выводится Вкладка "Работа с таблицами" Ленты и Окно объектов БД. В окне объектов базы данных открыт объект **Таблицы**. Для каждого объекта предусмотрено несколько способов создания: с помощью Конструктора, с помощью Мастера или путем ввода данных (только для таблицы). Режим конструктора – наиболее трудоемкий способ создания объектов, но он лучше позволяет познакомиться с возможностями объектов и их свойствами. Мастерами следует пользоваться после приобретения определенных навыков работы в СУБД.

#### **Совместимость с предыдущими версиями баз данных**

Microsoft Access 2010 позволяет создавать и использовать базы данных в форматах Access 2000, Access 2002-2003. Для этого необходимо выполнить небольшую настройку: введите команду **Файл, Параметры** и на закладке **Общие** в группе **Создание баз данных** выберите из списка **Формат файла по умолчанию для пустой базы данных**. Теперь база данных будет сохраняться в установленном формате.

### **3.3. Создание Таблицы**

Таблицы могут создаваться несколькими способами: в режиме **Таблица**, в режиме **Конструктора таблиц**, с помощью **шаблона таблицы**, а также с путём импортирования внешних данных: **Внешние данные, Импорт**.

Первый способ достаточно прост. Он предлагается программой по умолчанию при загрузке базы данных (рис. 3.3). В предлагаемой таблице одно поле. Вызовите контекстное меню поля "Щелкните для добавления" и выберите в нём тип добавляемого поля (рис. 3.4), Программа вставляет Поле1. Замените имя поля на требуемое, используя соответствующую команду контекстного меню. Затем вставьте следующее поле и т.д. При всей кажущейся простоте, этот способ создания таблиц ненагляден и неэффективен.

#### **Создание таблицы в Режиме конструктора таблиц**

Более удобным представляется создание таблицы в режиме Конструктора таблиц.

Перейти в режим конструктора из режима Таблица можно щелчком по кнопке Конструктор, расположенной в группе **Режимы** вкладки **Главная Ленты**. Выпадающее меню этой кнопки предлагает и другие режимы: Режим таблицы – переход из режима Конструктора в режим Таблица, Сводная таблица – создание сводных таблиц на основе существующих таблиц для их анализа или создание сводных диаграмм (рис. 3.5). Имеется и другой способ перехода в режим конструктора: откройте вкладку **Создание** Ленты и в группе **Таблицы** выберите **Конструктор таблиц**.

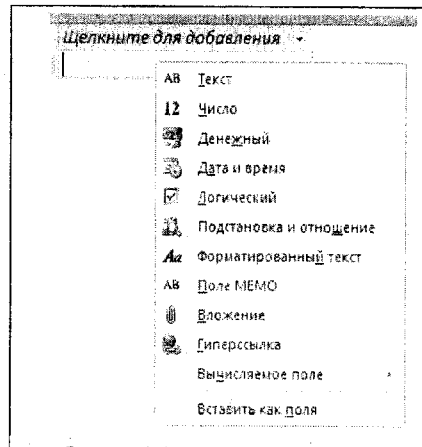


Рисунок 3.4 – Добавление поля в режиме Таблица

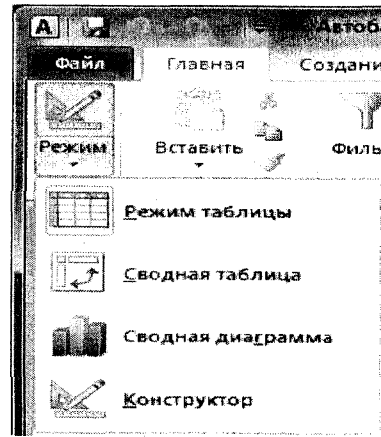


Рисунок 3.5 – Переход в режим конструктора

При переходе в режим Конструктора таблиц открывается шаблон для описания полей базы данных (рис. 3.6) – таблица с тремя столбцами: Имя поля, Тип данных и Описание.

В первой строке в столбце "Имя поля" введите название поля данных. Имя поля в Microsoft Access может быть любой комбинацией букв, цифр, пробелов и специальных символов, за исключением точки (.), восклицательного знака (!), обратного апострофа (') и квадратных скобок ([ ]), длиной до 64 символов. Кроме того, имя не может начинаться с одного или нескольких пробелов и содержать управляющие символы (символы с кодами ASCII от 0 до 31).

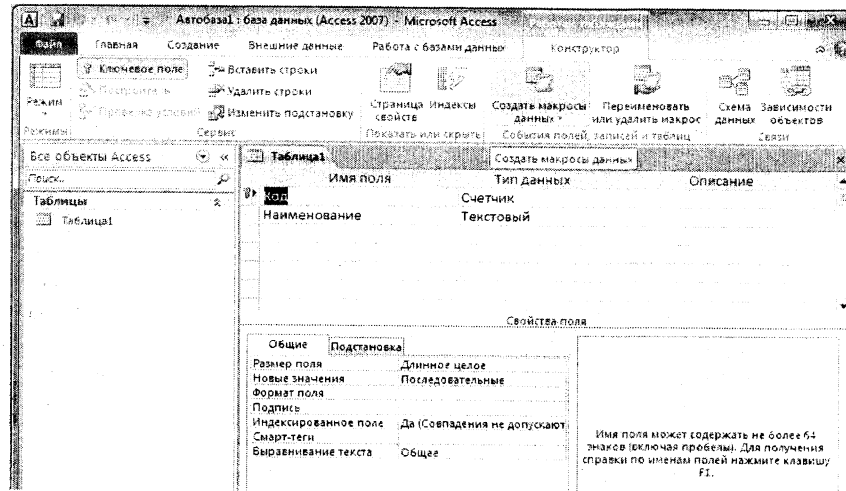


Рисунок 3.6 – Создание таблицы в режиме Конструктор



Во втором столбце выберите тип данных из раскрывающегося списка.

Внесите описание поля в столбце "Описание", при необходимости. Описание поля является необязательным, при работе с БД в режиме таблицы описание поля выводится в строку подсказки.

Настройте свойства полей в разделе "Свойства поля". Список свойств поля, выводимых на экран, определяется типом данных выбранного поля. В правом нижнем углу окна выводится описание свойства текущего поля.

Тип данных определяет вид и диапазон допустимых значений, которые могут быть введены в поле, а также объем памяти, выделяемый для этого поля. Перечень основных типов данных полей и описание значений, сохраняемых в таких полях, приведен в таблице 3.1.

Таблица 3.1 – Типы полей базы данных

Тип данных	Использование	Размер
Текстовый	Текст или комбинация текста и чисел, например, адреса, а также числа, не требующие вычислений, например, номера телефонов, инвентарные номера или почтовые индексы.	До 255 символов. Microsoft Access хранит только введенные в поле символы; незаполненные части полей типа «Текстовый» не сохраняются. Для определения максимального количества символов, которые можно ввести, используйте свойство <b>Размер поля (FieldSize)</b> .
Поле MEMO	Длинный текст или числа, например, примечания или описания. В данном поле хранится не сам текст, а только адрес места хранения текста.	До 64 000 символов.
Числовой	Числовые данные, используемые для математических вычислений, за исключением финансовых расчетов (для них следует использовать тип «Денежный»). Для более точного определения типа числа используйте свойство <b>Размер поля (FieldSize)</b> .	1, 2, 4 или 8 байтов.
Дата/время	Даты и время. Годы от 100 до 9999	8 байтов.
Денежный	Значения валют. Денежный тип используется для предотвращения округлений во время вычислений. Предполагает до 15 символов в целой части числа и 4 – в дробной.	8 байтов.
Счетчик	Автоматическая вставка последовательных (увеличивающихся на 1) или случайных чисел при добавлении записи.	4 байта.

Продолжение таблицы 3.1

Логический	Поля, содержащие только одно из двух возможных значений, таких как «Да/Нет», «Истина/Ложь», «Вкл/Выкл».	1 бит.
Поле объекта OLE	Объекты (например, документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звуки и другие двоичные данные), созданные в других программах, использующих протокол OLE. Объекты могут быть связанными или внедренными в табл. Microsoft Access. Для отображения объекта OLE в форме или отчете необходимо использовать присоединенную рамку объекта.	До 1 гигабайта (ограничено объемом диска).
Гиперссылка	Поле, в котором хранятся гиперссылки. В качестве гиперссылки может быть использован текст, адрес – маршрут или URL – адрес, файл или страница, экранный объект.	До 2048 символов.
Вложение	Любая ссылка типа файла	
Мастер подстановок	Создает поле, позволяющее выбрать значение из другой таблицы или из списка значений, используя поле со списком. При выборе данного параметра в списке типов данных запускается мастер для автоматического определения этого поля.	Тот же размер, который имеет первичный ключ, являющийся также и полем подстановок; обычно – 4 байта.

#### Создание первичного ключа

Для создания первичного ключа выделите поле, которое будет использоваться в качестве ключевого поля (если ключевое поле составное, то выделите все поля, которые будут входить в составной ключ), и щелкните по кнопке «Ключевое поле» в группе **Сервис** Конструктора работы с таблицами. При необходимости просмотреть свойства ключевых полей, а также установить первичный ключ, выделите ключевое поле и щелкните по кнопке «Индексы» в группе **Показать и скрыть**. Для первичного ключа свойствам «Ключевое поле» и «Уникальный номер» присвойте значение «ДА» (рис. 3.7.).

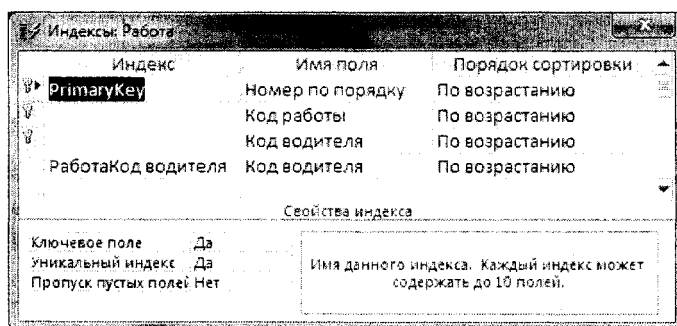


Рисунок 3.7 – Индексы ключевых полей

В качестве уникального первичного ключа можно создать поле типа Счётчик.

Сохраните таблицу. Для этого щелкните по кнопке закрытия окна таблицы. При закрытии таблицы программа запросит её имя, а также попросит определить первичный ключ, если он не был присвоен. Если ключ не был задан, то при ответе "Да" программа автоматически создаст поле типа Счётчик.

#### **Свойства полей базы данных**

Каждое поле имеет свои свойства, некоторые из этих свойств могут настраиваться. Окно диалога содержит две закладки: **Общие** и **Подстановка**. При активизации каждого поля справа появляется подсказка о назначении этого поля.

Для текстового поля обычно настраиваются свойства Размер поля, по умолчанию 255 символов, причём не введенные символы не сохраняются.

Для числовых полей настраиваются поля Размер поля, Формат поля, Число десятичных знаков, можно установить также Значение по умолчанию и Маску ввода.

Для полей типа Дата/время рекомендуется устанавливать Формат поля и Маску ввода. В этом случае при вводе данных будет предлагаться шаблон для заполнения поля.

#### **Свойства закладки общие**

**Размер поля** – определяет размер данных, хранимых в поле, которые зависят от типа данных.

Для текстовых полей свойство *Размер поля* – задает максимальный размер данных, которые могут быть помещены в данное поле. Если свойство "Тип данных" имеет значение "Текстовый", введите целое число от 0 до 255 (по умолчанию 255). Для текстового поля можно установить также маску ввода.

Если свойство "Тип данных" имеет значение "Числовой", то для свойства *Размер поля* допустимы следующие значения свойства:

- *Байт* – хранит числа от 0 до 255 (без дробной части); занимает 1 байт;
- *Целое* – хранит числа от -32 768 до 32 767 (без дробной части); занимает 2 байта;
- *Длинное целое* – хранит числа от -2 147 483 648 до 2 147 483 647 (без дробной части); занимает 4 байта;
- *Одинарное с плавающей точкой* – хранит числа с точностью до 6 знаков от -3,402823E38 до 3,402823E38; занимает 4 байта;
- *Двойное с плавающей точкой* – хранит числа от  $-1,798 \cdot 10^{308}$  до  $-4,941 \cdot 10^{-324}$  для отрицательных значений и от  $4,941 \cdot 10^{-324}$  до  $1,798 \cdot 10^{308}$  для положительных значений. Необходимо помнить, чем больше число, хранимое в памяти компьютера, тем больше ячеек памяти требуется для его хранения. Число байт, необходимых для хранения данных, определяется типом данных, а не их фактическим значением. Поэтому тип числовых полей следует выбирать в соответствии с потребностями решаемой задачи. Кроме того, обработка данных меньшего размера выполняется быстрее и требует меньше памяти.

**Формат поля** – определяет, как должно отображаться содержимое поля.

Для числовых полей *Формат поля* может принимать значения основной, денежный, процентный, экспоненциальный и др. Формат поля выбирается из раскрывающегося списка. При выборе формата поля можно просмотреть пример, воспользовавшись линейкой прокрутки в окне списка.

Для поля типа Дата/Время формат поля также выбирается из открывающегося списка с примерами шаблонов. Чаще всего используется Краткий формат даты.

**Число десятичных знаков** – устанавливает число десятичных знаков, выводимых в поля таблиц. Для числовых полей число десятичных знаков устанавливается автоматически в соответствии с выбранным форматом – Авто, но можно установить и собственное значение.

**Значение по умолчанию** – значение, которое будет появляться в поле автоматически при открытии новых записей. Для числовых полей можно ввести, например, 0. При активизации строки ввода справа появляется кнопка “троеточие”. Щелчок по ней вызывает окно диалога *Построитель выражений*, который позволяет выбирать из списка константы, операторы и функции.

**Условие на значение** – определяет область или диапазон допустимых значений данных, вводимых в поле, например:  $\geq 0$ ;  $\geq 10$  And  $\leq 100$ .

**Сообщение об ошибке** – содержит текст, который будет выводиться в сообщении при нарушении требований поля “условие на значение”.

**Подпись** – определяет тот текст, который будет отображаться рядом с полем в форме или в отчете.

**Маска ввода** – позволяет ограничить диапазон значений, которые могут вводиться в данное поле.

**Индексированное поле** – указывает СУБД, надо ли создавать для данного поля индексы для ускорения поиска, сортировки, фильтрации информации. Ключевые поля индексированы всегда.

**Обязательное поле** – если для поля установлено значение ДА, то поле должно обязательно заполняться при вводе данных.

**Смарт-теги** – встроенные программы – подсказки, замещающий текст, варианты действий, появляющиеся при выделении поля. Значения выбираются из Построителя выражений: Дата, Телефон, Контакты мгновенных сообщений, Имя.

#### **Создание маски ввода**

Маски ввода устанавливаются для текстовых полей, а также для полей типа Дата/время.

Для полей типа Дата/Время целесообразно установить Формат и Маску ввода. В этом случае при вводе данных будет предлагаться шаблон даты или времени.

Маска ввода для полей даты и времени выбираются из готовых шаблонов, а маска ввода для текстовых полей надо создавать самостоятельно. Для ввода маски активизируйте строку Маска ввода, щелкните по кнопке “троеточие” в правой части окна ввода – откроется окно диалога *Создание масок ввода*, в котором необходимо выбрать требуемый шаблон, а также установить пароль на поле для защиты от изменений.

Для создания собственного шаблона щелкните по кнопке Список – открывается окно диалога *Настройка масок ввода*, в котором необходимо ввести маску и пример заполнения (рис. 3.8).

Маска ввода может состоять из трёх частей, разделенных точками с запятой. Например, (999) 000-0000!;0;" :

*Первая часть* представляет собой собственно маску ввода: (999) 000-0000!).

*Вторая часть* маски ввода определяет режим занесения постоянных символов. Если постоянные символы должны быть включены в значение поля, введите 0; если в таблице следует занести только символы введенные пользователем, введите 1 или оставьте эту часть пустой (этот режим используется по умолчанию).

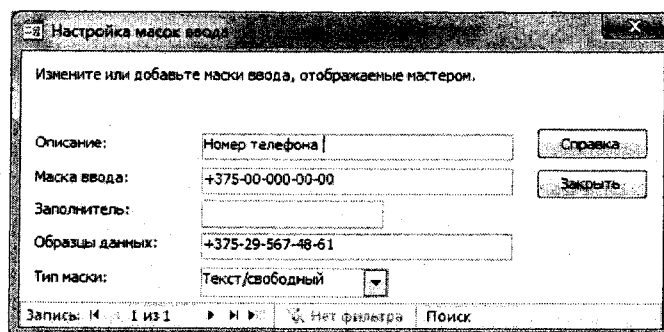


Рисунок 3.8 – Маска ввода для номера телефона

Третьим компонентом маски ввода является символ, который следует использовать для изображения пустых позиций в маске ввода. Пользователь может указать любой символ ANSI; пробел необходимо заключить в кавычки (" "). По умолчанию для этой цели используется символ подчеркивания (\_).

В таблице 3.2. перечислены символы маски и описано их назначение.

Таблица 3.2 – Символы маски

Символ маски	Назначение
0	Цифра (обязательный символ; знаки + и - не разрешены).
9	Цифра (необязательный символ; знаки + и - не разрешены).
#	Цифра, знак + или - или пробел (необязательный символ; незаполненные позиции преобразуются в пробелы).
L	Буква (обязательный символ).
?	Буква (необязательный символ).
A	Буква или цифра (обязательный символ).
a	Буква или цифра (необязательный символ).
&	Любой символ или пробел (обязательный символ).
C	Любой символ или пробел (необязательный символ).
, ; - /	Десятичный разделитель, разделитель тысяч, даты или времени; зависит от международных разделителей, установленных с помощью панели управления Windows (Control Panel).
<	Преобразует символы справа к нижнему регистру.
>	Преобразует символы справа к верхнему регистру.
!	Указывает, что маску следует заполнять справа налево; этот символ следует использовать, если позиции для заполнения находятся в левой части маски.
\	Указывает, что следующий символ следует воспринимать буквально, как постоянный символ маски ввода; этот символ следует использовать, если в маску ввода требуется включить один из перечисленных выше символов.

Для того чтобы вводимые в поле символы не отображались на экране, следует выбрать значение "Пароль" свойства "Маска ввода". Вместо каждого символа, введенного в поле, будет изображаться звездочка (\*). Этот тип маски ввода не налагает никакие ограничения на вводимые значения; он определяет только способ изображения этих значений на экране.

#### Свойства закладки Подстановка

В базе данных должна быть обеспечена минимальная избыточность данных. То есть все данные должны вводиться, как правило, один раз. Но некоторые данные могут повторяться в других таблицах, например, в качестве внешних ключевых полей. Некоторые поля базы данных могут заполняться данными, которые не меняются в процессе эксплуатации базы данных, например, города областных и районных центров, названия областей, единицы измерения материальных ценностей и т.д. Это так называемые постоянные или условно постоянные данные (перечисления). Для хранения таких данных можно создавать специальные таблицы или списки. Чтобы не вводить данные дважды или трижды или воспользоваться данными, хранящимися во вспомогательных таблицах, необходимо установить между таблицами связь.

Такая связь устанавливается с помощью закладки Подстановка (рис. 3.9).

При открытии закладки на экране присутствует одно свойство: *Тип элемента управления*. Это Свойство имеет три значения: *Поле*, *Список* и *Поле со списком*. Значение *Поле* установлено по умолчанию.

Значения *Список* и *Поле со списком* позволяют ввести список значений, разделяя их символом точка с запятой – ";", например, Да; Нет или выбрать значения из таблицы. Отличие состоит в том, что значение *Список* создаёт развёрнутый список значений, а значение *Поле со списком* создаёт раскрывающийся список.

Свойство *Тип источника строк* имеет три значения: *Таблица или запрос*, *Список значений* и *Список полей*.

При установке значения *Список значений* в свойстве *Источник строк* необходимо ввести список значений этого поля, разделяя их символом точка с запятой.

При установке значения *Таблица или запрос* программа предлагает выбрать таблицу или запрос в качестве источника значений.

Свойство *Присоединённый столбец* позволяет указать номер поля таблицы или запроса, являющихся источником данных.

Свойство *Число столбцов* указывает число столбцов, присоединяемых в качестве источника значений.

Настройку Поля подстановки можно производить вручную, но лучше поручить эту работу **Мастеру подстановок**.

**Внимание:** для создания подстановки с помощью Мастера подстановок необходимо удалить связи данной таблицы с другими таблицами в схеме данных.

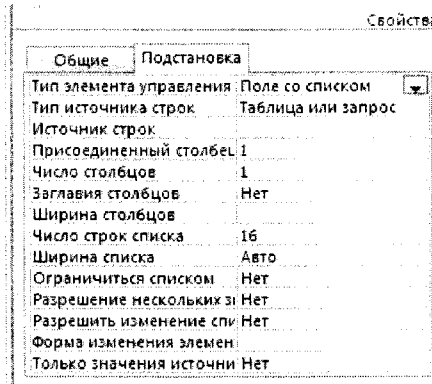


Рисунок 3.9 – Вкладка Подстановка

Алгоритм создания поля подстановки рассмотрим на примере создания поля подстановки для поля *Номер транспортного средства* таблицы Водители. В базе данных имеется таблица Транспортное средство, содержащая такое же поле.

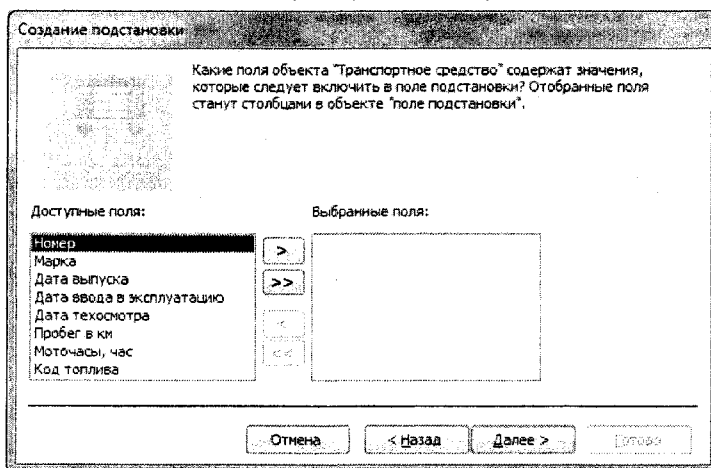
– Откроем таблицу Водители, Перейдём в режим Конструктора командой **Режим, Конструктор**.

– Откроем список Тип данных для поля *Номер транспортного средства* и выберем в списке **Мастер подстановок** – откроется окно диалога **Создание подстановки**.

– Установим переключатель операции в значение “Объект “поле подстановки” получит значения из другой таблицы или другого запроса” (предлагается по умолчанию). Щёлкнем кнопку Далее. Откроется окно диалога для выбора таблицы.

– Активизируем переключатель Таблицы (установлен по умолчанию) и выберем таблицу Транспортное средство, щёлкнем кнопку Далее.

– Следующее окно диалога предлагает выбрать поле, которое будет использоваться в качестве поля подстановки (рис. 3.10). При этом рекомендуется выбирать не одно поле, а несколько, чтобы можно было легко понять, что следует вводить в таблицы при использовании данного поля подстановки. В рассматриваемом примере в качестве полей подстановки примем поля Номер и Марка. Выделим поле Номер и перенесём его из области *Доступные поля* в область *Выбранные поля* кнопкой “Вправо” – >. Выполним данную операцию также для поля Марка. Щёлкнем кнопку Далее.



**Рисунок 3.10 – Создание подстановки, шаг третий**

– На следующем шаге программа предлагает выбрать поле для сортировки. Сортировка ускоряет выбор данных. Сортировку можно выполнять по четырём полям. При сортировке по нескольким полям необходимо следить, чтобы каждое предыдущее условие разбивало всю совокупность данных на меньшее число групп. Выберем в верхнем списке поле Номер. Щёлкнем кнопку Далее.

– На следующем шаге программа предлагает установить оптимальную ширину столбца с данными. Для этого необходимо щёлкнуть дважды по правой границе поля. Если

поле подстановки является ключевым, то необходимо снять флажок *Скрыть ключевой столбец*. Щёлкнем кнопку *Далее*.

– На следующем шаге программа сообщает, какие имеются доступные поля.

– На последнем шаге программа предлагает задать подпись, которую содержит поле подстановки. По умолчанию предлагается *Номер транспортного средства*. Для завершения работы щёлкнем по кнопке *Готово*. В появившемся окне диалога содержится рекомендация *"Перед созданием связи необходимо сохранить таблицу"* и вопрос: *"Выполнить это сейчас?"*. На этот вопрос следует ответить *"ДА"*.

В качестве источника значений полей следует выбирать главную, независимую таблицу.

### Создание Схемы данных

После создания всех таблиц базы данных необходимо установить связи между таблицами и контроль целостности и непротиворечивости данных. Для создания связей закройте все таблицы и щёлкните по кнопке *Схема данных* в группе *Отношения* вкладки *Работа с базами данных*. Открывается пустое окно схемы данных и Окно диалога *Добавление таблицы* (рис. 3.11).

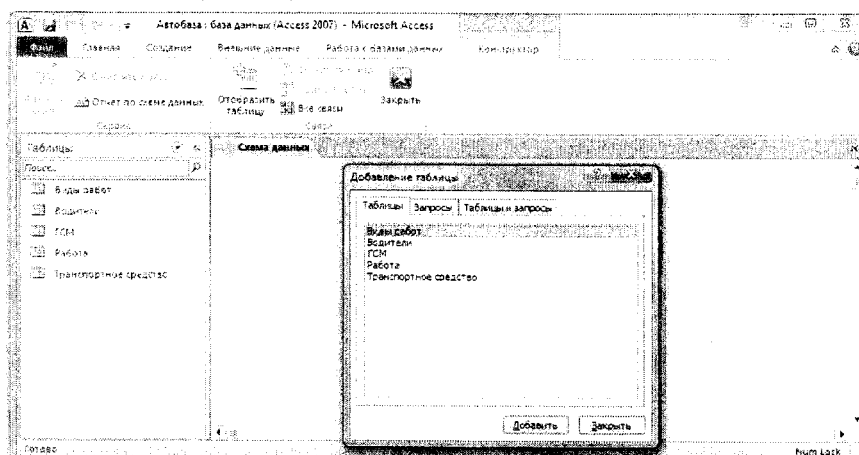


Рисунок 3.11 – Создание Схемы данных

**Примечание:** Если для создания полей подстановки использовался Мастер подстановок, то при открытии окна *Схемы данных* в нём появятся связанные таблицы с линиями связи между ключевыми полями.

В окне *Схема данных* выводится макет связей между таблицами и запросами в текущей базе данных. В этом окне пользователь имеет возможность просматривать или изменять существующие связи или определять новые связи между таблицами и запросами. При создании макета допускается перемещение таблиц и запросов в окне схемы данных.

Связанные таблицы соединяются в окне линией объединения. Если для отношений между таблицами наложены условия обеспечения *целостности данных*, линия объе-



динения выводится с жирными концами, а на каждую сторону помещаются символы, указывающие тип отношений между таблицами (один и многие (∞)) (см. рис. 3.13).

Добавьте все необходимые таблицы в макет схемы данных (для выделения всех таблиц в окне диалога (рис. 3.11) выделите щелчком мыши первую таблицу, нажмите и удерживайте клавишу Shift на клавиатуре и выделите мышкой последнюю таблицу в списке).

После этого необходимо установить связи между таблицами. Все таблицы по своему статусу можно разделить на главные и подчинённые. Главные таблицы не зависят от других таблиц. В проектируемой базе данных независимыми таблицами являются таблицы Виды работ, ГСМ. Таблицы Транспортные средства, Водители и Работа подчинённые. Одна и та же таблица может быть главной по отношению одной таблице и подчинённой по отношению к другой таблице.

Установление связей между таблицами осуществляется путём перетаскивания ключевых полей из главной таблицы на соответствующие поля подчинённой таблицы. Например, Код работы таблицы Виды работ перетащите на поле Код работы таблицы Работа – открывается окно диалога *Изменение связей* (рис. 3.12). Установите флажки *Обеспечение целостности данных*, *Каскадное обновление связанных полей* и *Каскадное удаление связанных записей*

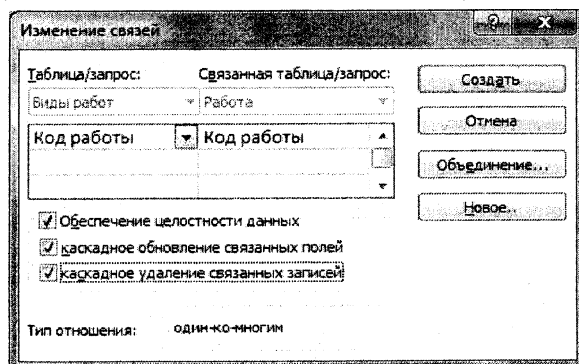


Рисунок 3.12 – Окно Базы данных

ное удаление связанных записей и щёлкните по кнопке *Создать*. Окно диалога закрывается. После установки связей между всеми таблицами разместите их в окне Схема данных наиболее удобным образом и закройте окно Схема данных (рис.3.13). При этом программа попросит Вас сохранить Схему данных.

Пользователь имеет возможность добавлять таблицы или запросы в окно схемы данных. Для этого следует вызвать контекстное меню Схемы данных и выбрать команду "Добавить таблицу".

Для конкретной таблицы или запроса возможен вывод только таблиц или запросов, с которыми они имеют прямые связи. Для этого следует добавить таблицу или запрос в пустое окно схемы данных и выбрать в группе *Связи* команду *Прямые связи*. Для вывода всего макета выберите в группе *Связи* команду *Все связи*.

Допускается очистка окна схемы данных с помощью команды *Очистить макет* из группы *Сервис*, не приводящая к уничтожению связей между таблицами. Допускается

также сохранение измененного пользователем макета схемы данных с помощью команды **Сохранить макет** контекстного меню. Каждый из пользователей базы данных, работающих в сети, имеет возможность сохранить собственный макет.

#### Установление связи между таблицами

Связь между таблицами устанавливает отношения между совпадающими значениями в ключевых полях – обычно между полями разных таблиц. В большинстве случаев связывают ключевое поле одной таблицы с соответствующим ему полем (часто имеющим то же имя) во второй таблице, которое называют полем внешнего ключа.

*Таблица, содержащая ключевое поле, называется главной, а таблица, содержащая внешний ключ – связанной или подчиненной.*

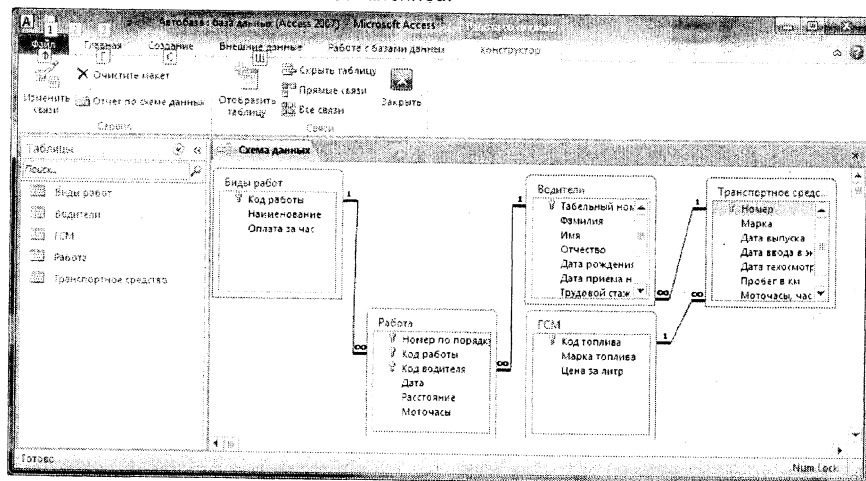


Рисунок 3.13 – Схема данных

#### Обеспечение целостности данных

Поддержание целостности данных гарантирует сохранение существующих связей между таблицами при вводе и удалении записей и недопущение случайного удаления связанных данных. Обеспечение целостности данных обеспечивается активизацией флажка "Обеспечение целостности данных" в окне диалога **Изменение связей** (рис.3.12). При наличии поддержания целостности данных:

- в связанное поле подчиненной таблицы можно вводить только те значения, которые имеются в связанном поле главной таблицы;
- невозможно удалить из главной таблицы запись, с которой связаны одна или несколько подчиненных записей.

#### Каскадное обновление и удаление связанных полей

Microsoft Access может автоматически выполнять каскадное удаление и обновление связанных записей. Это означает, что при удалении главной записи или изменении значения ключа, внесенные изменения будут автоматически отражены в подчиненных таблицах.

Для изменения связей или обеспечения целостности данных щелкните мышью по связи – откроется окно диалога Изменение связей (рис. 3.12).

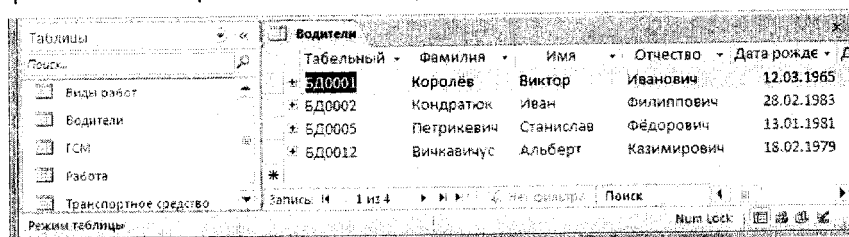
Если установлен флажок **Каскадное обновление** связанных полей, то выполняется каскадное обновление ключевых полей для операции, которая в другом случае была бы запрещена. При любом изменении ключевого поля или записи в главной таблице будут автоматически приведены в соответствие все значения в связанных записях.

Если установлен флажок **Каскадное удаление** связанных записей, то при любом удалении записей из главной таблицы будут автоматически удалены все связанные записи в связанных таблицах. При удалении записей в режимах формы или таблицы в этом случае выводится предупреждение о возможности удаления связанных записей. Однако при удалении записей с помощью запроса на удаление, записи в связанных таблицах удаляются автоматически без вывода предупреждения.

#### Ввод и редактирование данных

После создания Схемы данных можно приступить к вводу данных в таблицы. *Ввод данных необходимо начинать с главных таблиц.*

Для ввода данных в таблицу откройте таблицу двойным щелчком мыши по имени таблицы. Появляется таблица (рис. 3.14). В верхней части таблицы выведены имена полей базы данных. Внизу таблицы расположены кнопки навигатора для перемещения по записям базы данных. Если данные не умецаются в окне таблицы, то появляются горизонтальные и вертикальные линейки прокрутки.



Табельный	Фамилия	Имя	Отчество	Дата рожд
5D0001	Королёв	Виктор	Иванович	12.03.1965
5D0002	Кондратюк	Иван	Филиппович	28.02.1983
5D0005	Петрикевич	Станислав	Фёдорович	13.01.1981
5D0012	Вичкавичус	Альберт	Казимирович	18.02.1979

Рисунок 3.14 – Ввод данных в таблицу

Данные вводятся, как обычно, с клавиатуры, переход от одного поля к другому осуществляется с помощью мыши или клавиши Tab. Ширину столбцов можно регулировать, перетаскивая границу поля мышью. В этом режиме можно также форматировать текст в полях таблицы с помощью команд группы **Форматирование текста** вкладки **Главная Ленты**.

При открытии таблицы активизируется вкладка **Работа с таблицам**, которая имеет две вкладки: **Поля** и **Таблица**. Вкладка **Поля** позволяет редактировать свойства полей базы данных, а вкладка **Таблица** – позволяет настраивать свойства таблицы, изменять схему данных, назначать макросы, которые должны быть выполнены при соответствующих изменениях таблицы, просматривать зависимости таблицы.

Ввод данных в подчинённые таблицы удобно производить с использованием главных таблиц. В главных таблицах слева от каждой записи имеется кнопка "+". При щелчке мышью по данной кнопке открывается связанная запись подчинённой таблицы. Кнопка "+" заменяется при этом на кнопку "-". Чтобы закрыть подчинённую таблицу, теперь надо щёлкнуть мышью по кнопке "-".

### Просмотр базы данных

Для просмотра таблицы базы данных щёлкните по ней дважды мышью. Для перехода из режима Таблицы в режим Конструктора удобно пользоваться кнопкой **Режим** в группе **Режимы** вкладки **Главная**.

### Вставка и удаление полей, изменение ключей

Вставка или удаление полей, изменение ключей выполняется в режиме Конструктор. В режиме конструктора можно также изменить структуру уже созданной таблицы (изменить названия полей, их тип, свойства и т.п.).

### Удаление и переименование таблиц

Для удаления или переименования таблиц следует воспользоваться контекстным меню таблицы, вызываемом щелчком правой кнопки мыши по таблице в окне объектов.

### Изменение макета таблицы

Изменить шрифты, высоту строк и ширину столбцов, удалить и добавить строки, и изменить порядок расположения столбцов и т.п. можно в режиме таблицы.

Выделение строк осуществляется щелчком мыши по полю выделения – левое поле таблицы. Выделение столбцов осуществляется щелчком мыши по заголовку поля.

Для добавления записи щёлкните по кнопке **►\*** в строке навигатора. Новая запись добавляется в конце таблицы.

Для удаления записи выделите ее и нажмите клавишу Del. После этого на экран будет выведен запрос на подтверждение удаления записи. Удаленные записи не восстанавливаются.

### Печать таблицы и форм в ACCESS

Откройте таблицу в режиме **«Таблица»**. Введите команду **Файл, Печать**. В этом режиме можно предварительно просмотреть таблицу и отправить на печать. При выборе команды **Быстрая печать** таблица выводится на печать без настройки параметров печати. При выборе команды **Печать** отрывается окно диалога для настройки параметров печати, которое позволяет установить что печатать (всё, указанные страницы или выделенные записи, число копий). Команда **Настройка** позволяет установить дополнительные параметры печати. Печать можно отправить на принтер или в файл. При необходимости можно настроить параметры принтера (рис. 3.15).

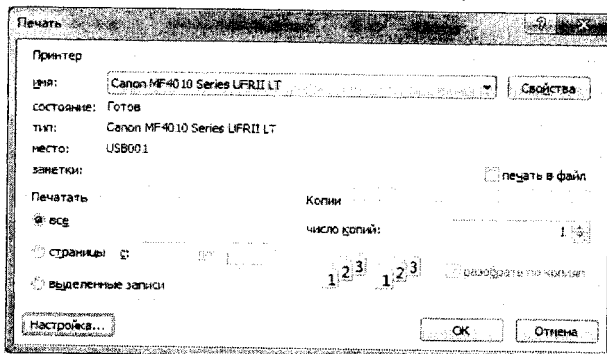


Рисунок 3.15 – Печать таблиц

### Использование баз данных

Использование баз данных заключается в поиске и выборке информации по определенному критерию, а также добавлению и изменению данных. Критерий включает в себя имя поля и значение.

При поиске информации программа позволяет только найти и просмотреть информацию, интересующую пользователя. Поиск информации можно сочетать с внесением изменений в базу данных. При этом новые объекты базы данных не создаются.

При выборке информации осуществляется поиск и извлечение информации из базы данных с созданием нового объекта. Одновременно с выборкой данных могут осуществляться различные арифметические и логические операции над данными.

### Поиск данных

Поиск данных осуществляется в режиме таблицы. Откройте таблицу и введите команду *Найти* в одноименной группе вкладки *Главная ленты*. Отрывается окно диалога (рис. 3.16), в котором надо указать контекст для поиска в строке *Образец* и настроить параметры поиска: указать поле, по которому будет выполняться поиск, условия совпадения текста (целиком, с любой частью поля, с началом поля), направление просмотра (вверх, вниз, все), а также установить, при необходимости, флажки для учёта регистра и формата полей. При необходимости выполнить замену текста, откройте закладку "Замена", укажите текст для поиска и текст для замены.

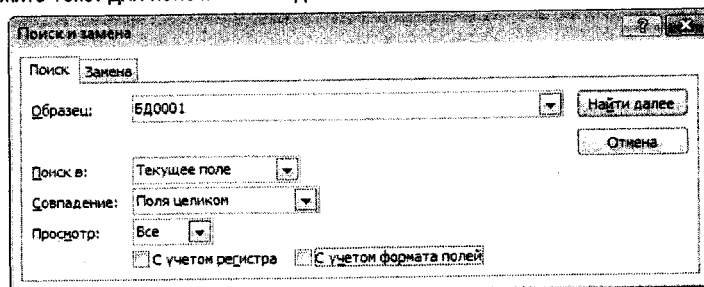


Рисунок 3.16 – Поиск и замена данных

При записи контекста можно использовать маску. В качестве маски допускается использовать следующие символы:

\* – заменяет все слово или его часть, может использоваться в начале или в конце слова;

? – заменяет один символ в той позиции, где он установлен;

# – подставляется вместо любой значащей цифры;

[ ] – допускается подстановка одного любого символа из указанных в квадратных скобках. В квадратных скобках можно указать диапазон допустимых символов в порядке возрастания, разделяя их символом "-", например, [2-3] или символ "!", который определяет исключение из допустимых символов, например [!2].

### Сортировка записей в таблицах и формах

Сортировка производится с целью упорядочивания данных. Кроме того, в операциях поиска и выборки данных операции осуществляются быстрее, если данные предварительно отсортированы.

Сортировку можно проводить по конкретному полю таблицы или формы. Сортировка может проводиться по нескольким полям. Для этого поместите столбцы, по которым проводится сортировка, рядом. Выделите их и в группе Сортировка и фильтр вкладки Главная, введите требуемый режим сортировки: по возрастанию или по убыванию. При выполнении сортировки по нескольким полям сортировка проводится сначала по первому полю, затем по второму полю и т.д.

#### Фильтрация данных

Фильтрация записей таблиц осуществляется для поиска данных, удовлетворяющих определенному критерию или нескольким критериям. Критерии для поиска могут быть простыми и составными. При этом различают несколько схем объединения критериев по схеме ИЛИ, по схеме И и смешанная схема ИЛИ-И (рис. 3.17). Принцип простой: если условия относятся к одной записи, то критерии объединены по схеме И. В других случаях критерии объединены по схеме ИЛИ. Для объединения критериев используются функции OR – ИЛИ и AND – И.

Простой критерий	Составной критерий по схеме ИЛИ	Составной критерий по схеме ИЛИ	
Поле	Поле	Поле 1	Поле 2
Значение	Значение 1 Значение 2	Значение	Значение
Наименование	Наименование	Наименование	Цена
Минск	Минск Гродно	Горизонт	< 3200000
= Минск	Минск OR Гродно	Горизонт AND <3200000	

Рисунок 3.17 – Примеры критериев для поиска и фильтрации данных

Логический оператор AND может использоваться для выборки значений и из одного поля. Например, Цена >2000 AND <2500. Но Цена <2000 и Цена >2500 должны уже объединяться с помощью оператора OR: Цена <2000 OR >2500.

Графическая интерпретация данных условия представлена на рис. 3.18.

1. Access выполняет логическую операцию **AND** над условиями отбора, находящимися в одной строке *Бланка запроса (И-запрос, схема И)*. Т.е. запись будет выбрана, если оба условия будут возвращать значение Истина.

2. Access выполняет логическую операцию **OR** над условиями отбора, находящимися в разных строках *Бланка запроса (ИЛИ-запрос, схема ИЛИ)*. Т.е. запись будет выбрана в том случае, если хотя бы одно условие будет возвращать значение Истина.

СУБД предоставляет несколько способов задания фильтров:

- фильтр по выделенному;
- фильтр по форме;
- расширенный фильтр.

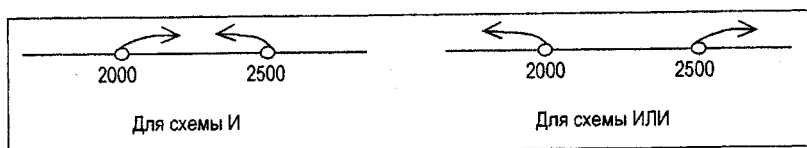


Рисунок 3.18 – Графическая интерпретация логических выражений И и ИЛИ

Для ввода команд используются команды группы **Сортировка и фильтр** вкладки **Главная ленты**.

#### Фильтр по выделенному

Откройте таблицу, например, Работа.

Выделите в соответствующем поле таблицы значение, необходимое для поиска, например, в поле *Код работы* выделите значение ВР1.

Откройте раскрывающийся список *Выделение* и выберите условие для выбора: равно, не равно, содержит, не содержит, например, равно. Сразу же после выбора условия в таблице останутся только те записи, которые удовлетворяют условию. Содержание списка зависит от типа данных.

Для отмены фильтра щёлкните по кнопке **Фильтр (Удалить фильтр)** (справа, внизу группы **Сортировка и фильтр**).

Последний примененный фильтр автоматически сохраняется и может быть применен вновь в последующем кнопкой **Фильтр (Применить фильтр)**.

Для удаления фильтра с поля откройте раскрывающийся список поля *Код работы* и выберите команду **Снять фильтр с Код работы** – кнопка **Фильтр** становится недоступной.

#### Автофильтр

Этот фильтр вводится кнопкой **Фильтр** в группе **Сортировка и Фильтр** вкладки **Главная ленты** (слева).

- Откройте таблицу, например, Работа.
- Выделите поле, например, Номер по порядку.

Щёлкните по кнопке **Фильтр** или откройте раскрывающийся список поля *Номер по порядку*. Откроется список, содержащий все значения данного поля (рисунок 3.19).

Снимите все флажки, кроме тех, которые Вам нужны, и щёлкните по кнопке **ОК** (реализуется объединение критериев по схеме ИЛИ).

В этом окне диалога дополнительно можно установить числовой (или текстовый фильтр): *равно*, *не равно*, *больше*, *меньше* или *между* (в последнем случае реализуется объединение критериев по схеме И).

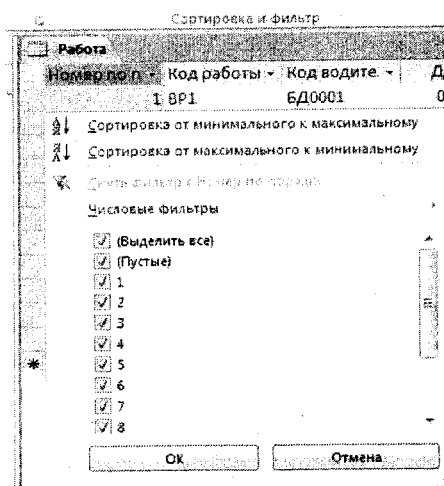


Рисунок 3.19 – Автофильтр

### Фильтр по форме

Откройте таблицу, например, Работа. Откройте список *Дополнительно* в группе *Сортировка и фильтр* и выберите команду *Изменить фильтр* – откроется шаблон для ввода условий отбора (рис. 3.20).

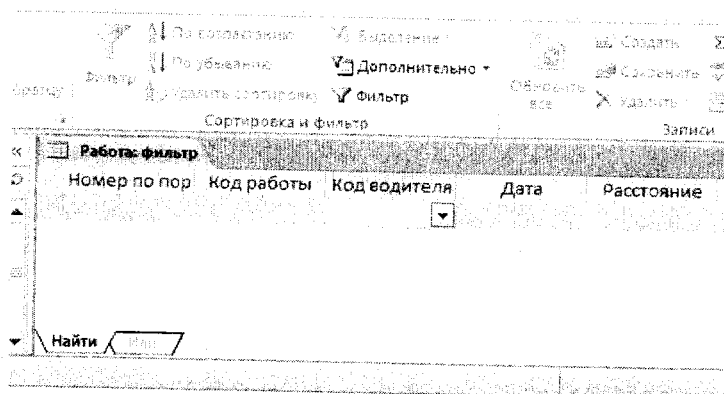


Рисунок 3.20 – Фильтр по форме

При активизации любого поля на нём появляется раскрывающийся список. Откройте его и выберите нужное Вам значение. Если значения выбраны в нескольких полях, например, Код работы и Код водителя, то реализуется схема И.

После выбора первого значения в нижней строке активизируется вкладка ИЛИ. Щёлкните по ней мышью – откроется новая строка для выбора условия. Значения можно выбирать для того же поля или другого поля таблицы. При этом критерии отбора будут объединены по схеме ИЛИ.

Для применения фильтра введите команду *Фильтр, Применить фильтр*.

### Расширенный фильтр

Расширенный фильтр предоставляет окно для задания условий отбора, отображаемых записей таблицы или запроса. Окно расширенного фильтра приведено на рис. 3.21.

Окно расширенного фильтра состоит из двух частей: схемы данных и условий отбора. В раздел *Схема данных* выводится структура таблицы. В раздел *Условия отбора* выводится список полей, используемых для отбора, задаются условия сортировки, при необходимости, и собственно условия отбора.

Условие отбора представляет собой выражение, которое состоит из операндов и операторов.

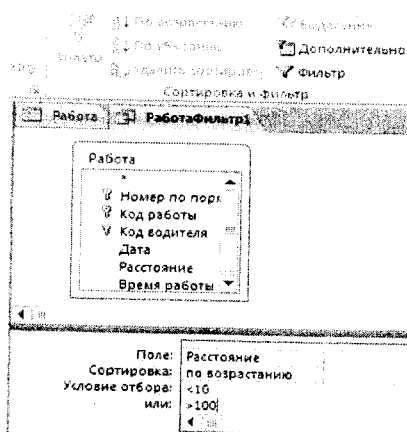


Рисунок 3.21 – Расширенный фильтр



В качестве операндов могут использоваться литералы, константы, идентификаторы (ссылки).

Литералы – это числа, строки символов, заключенные в кавычки, а также даты, заключенные в символы #: Например, 2673, "январь", #01.01.2013#.

Константы – это не изменяемые значения, например: True, False, Да, Нет, Null.

Идентификаторы представляют собой ссылку на значение поля, элемент управления или свойства. Идентификаторами могут быть поля таблиц, форм, отчетов и т.д. Идентификаторы заключаются в квадратные скобки. Если идентификатор содержит имя таблицы, то он отделяется от имени поля восклицательным знаком. Например:

- ссылка на имя поля – [Имя таблицы]![Имя поля]
- ссылка на свойство – Forms![Автобаза]![Расстояние].Значение

Операторами сравнения и логическими операторами в условных выражениях могут быть: =, <, >, <>, <=, >=, Between, In, Like, And, Or, Not.

В операторах можно использовать символы шаблона \* и ?.

Оператор Between позволяет задать интервал изменения параметра:

- Between 10 And 100 – задает интервал изменения параметра от 10 до 100.

Оператор Like позволяет использовать образцы, использующие символы шаблона: Like "Иванов\*".

Для формирования выражений можно использовать построитель выражений (см. следующий раздел).

Расширенный фильтр можно сохранить как запрос. Для сохранения расширенного фильтра следует воспользоваться командой Сохранить как запрос контекстного меню бланка запроса. Чтобы уничтожить текущий запрос, связанный с таблицей и автоматически выполняющийся по команде Применение фильтра, выберите в контекстном меню команду Очистить бланк. Для использования существующего фильтра откройте бланк запроса и выберите в контекстном меню команду Загрузить из запроса. Упомянутые выше команды имеются также и в списке Дополнительно группы Сортировка и фильтр.

### **Создание запросов, форм и отчетов**

#### **Типы запросов**

Запросы служат для отбора записей из одной таблицы или связанных таблиц на основе условия, заданного пользователем, а также создания вычисляемых полей. Запросы используются для просмотра, анализа и изменения данных. Запросы могут служить источниками записей для форм, отчетов и страниц доступа к данным. Запросы могут быть однотабличные и многотабличные. В однотабличных запросах источником данных является одна таблица. Многотабличные запросы строятся на основе связанных таблиц или запросов.

В Access можно создавать несколько различных типов запросов:

- запрос на выборку;
- запрос на выборку с вычисляемыми полями;
- запрос с параметрами;
- перекрестные запросы;
- запросы на изменение;
- SQL запросы.

Запросы на выборку строятся на основе одной или нескольких взаимосвязанных таблиц, при этом могут использоваться таблицы базы данных, а также созданные и сохра-

ненные таблицы на основе запросов. Результаты выборки помещаются в новую, как правило, временную таблицу. В запросах можно выполнять вычисления, помещая результаты вычислений в новое поле.

Запрос с параметрами представляет собой обычный запрос на выборку, но значение параметров для отбора задается пользователем в режиме диалога. Запросы с параметрами позволяют производить выборку из связанных таблиц, при этом результирующий запрос представляет собой таблицу зависимости одного параметра от другого.

Запросы на изменение позволяют добавлять, удалять записи или вносить в них изменения.

### Создание запросов

Запросы создаются с помощью команд группы **Запросы** вкладки **Создание** ленты (рис. 3.22).

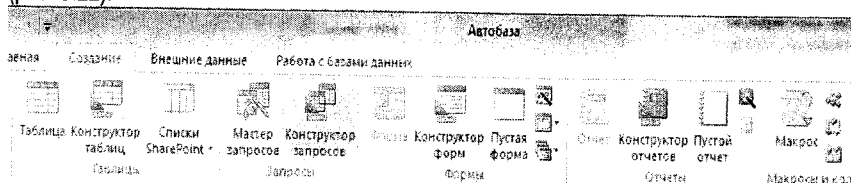


Рисунок 3.22 – Вкладка **Создание** Ленты

Запросы могут создаваться с помощью Мастера запросов или Конструктора запросов.

Для создания запроса на выборку с помощью Конструктора запросов выберите команду **Конструктор запросов** в группе **Запросы** вкладки **Создание**. При этом на экран выводится диалоговое окно конструктора запроса по образцу, и откроется окно диалога **Добавление таблицы** (рис. 3.23). Выберите нужные таблицы или запросы и щелкните по кнопке **Добавить**. Выбранные таблицы помещаются в окно Конструктора запросов. Связи между таблицами устанавливаются автоматически по ключевым полям. Окно конструктора запросов разделено на две панели: верхнюю и нижнюю (рис. 3.24).

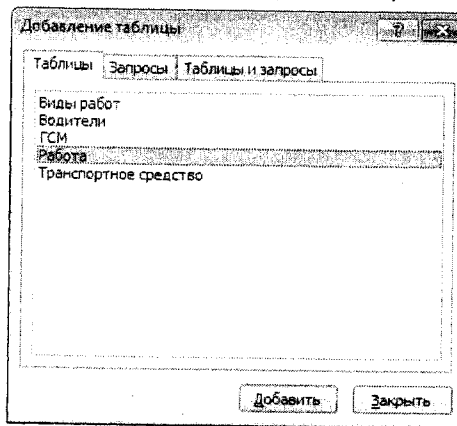


Рисунок 3.23 – Окно диалога **Добавление таблицы**

Верхняя панель содержит схему данных, которая включает все таблицы, используемые для выполнения запроса и связи между ними. Таблицы представлены списками полей. Нижняя панель QBE (Query by example) является бланком запроса по образцу, который нужно заполнить.

### Однотабличный запрос

Простой однотабличный запрос, сформированный в режиме конструктора, приведён на рис. 3.24. В запрос включены три поля: **Фамилия**, **Имя** и **Отчество**. Выполнена сорти-

ровка по всем полям по возрастанию. Все поля выводятся на экран. Тип запроса **Выборка** установлен по умолчанию на вкладке **Работа с запросами**, группа **Тип запроса**. Вкладка **Работа с запросами** открывается автоматически сразу же после закрытия окна диалога **Добавление таблицы**.

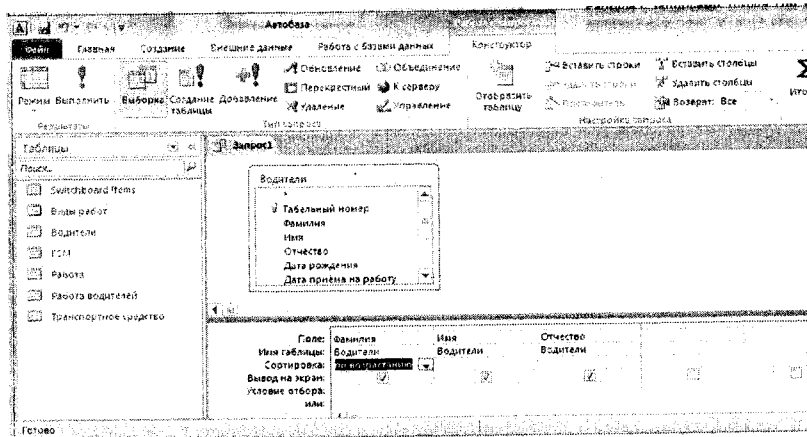


Рисунок 3.24 – Создание однотобличного запроса

#### Правила включения полей в запрос

Выбор полей из таблицы-источника (из верхней части окна *Запрос на выборку*) в *Бланк QBE* (нижнюю часть окна) можно выполнить следующими способами:

- Путём "перенести – и-бросить";
- путем выбора поля таблицы в верхней части окна двойным щелчком мыши;
- с помощью выбора нужного поля из списка, находящегося непосредственно в ячейке строки *Поле Бланка запроса* (или комбинацией клавиш **Alt + ↓**).

В строке *Поле* появится имя поля, а в строке *Имя таблицы* автоматически появится имя таблицы-источника; в строке *Вывод на экран* будет установлен флажок.

**Замечание:** для включения всех полей в бланк QBE нужно выбрать пункт \*, при этом отключить режим вывода на экран для дополнительных полей, по которым задается критерий отбора.

Для выполнения запроса щёлкните по кнопке **Выполнить** в группе **Результаты** вкладки **Работа с запросами**. Сохраните запрос, для чего закройте окно Конструктора запроса. Программа выведет на экран окно диалога с предложением изменить имя запроса. Другой способ сохранения любого объекта: Введите команду **Файл, Сохранить объект как**. При этом откроется окно диалога, в первой строке которого необходимо ввести имя запроса, а во второй строке будет указан тип объекта – Запрос.

**Замечание:** имя запроса должно отличаться от имени любого другого объекта в исходной базе данных.

#### Многотабличный запрос

На рис. 3.25 приведён пример многотабличного запроса. Запрос основан на трёх таблицах: *Водители*, *Работа*, *Вид работ*. Сортировка выполнена по двум полям: *Дата* и *Фа-*

милля. При этом вначале таблица будет отсортирована по датам, а потом в пределах дат по фамилиям.

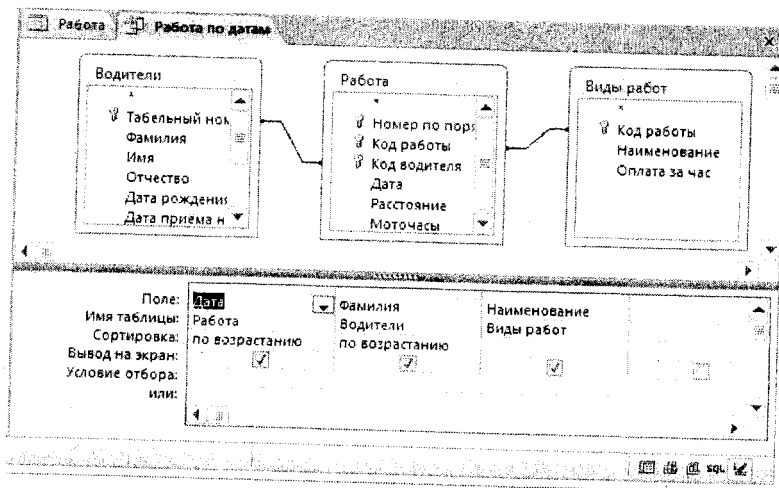


Рисунок 3.25 – Многотабличный запрос

**Примеры запросов на выборку с условиями**

**Пример 3.1.** Требуется вывести по датам сведения о водителях, которые совершали поездки на расстояния от 10 до 100 км включительно.

Запрос формируется на основе таблиц Работа и Водители. При формировании условия использован оператор Between (рис. 3.26).

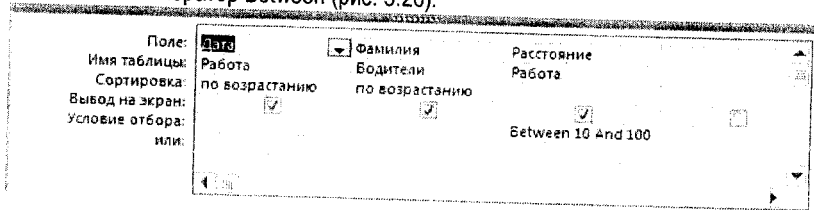


Рисунок 3.26 – Запрос на выборку с условием

**Пример 3.2.** Необходимо составить список водителей, трудовой стаж которых менее 3 лет или больше 20 лет.

Пример бланка запроса приведён на рис 3.27. Критерии отбора объединены по схеме ИЛИ. Запрос создан на основе таблицы Водители.

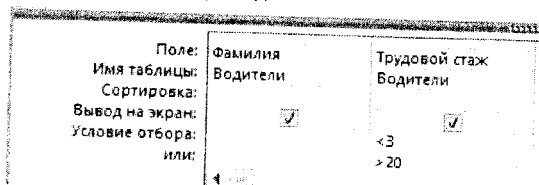


Рисунок 3.27 – Запрос на выборку с условием

**Пример 3.3.** Составить список водителей, возраст которых на 1 января 2013 года был 50 и более лет.

Пример запроса приведён на рис. 3.28. Под указанный критерий отбора подпадают все водители, которые родились первого января 1963 года или раньше. Запрос сформирован на основе таблицы Водители.

Поле:	Фамилия	Дата рождения
Имя таблицы:	Водители	Водители
Сортировка:	по возрастанию	
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:		<=#01.01.1963#

Рисунок 3.28 – Запрос на выборку с условием

**Пример 3.4.** На рис. 3.29 сформирован запрос на выборку водителей, которые были в дальних рейсах в мае 2013 года. Запрос сформирован на основе трёх таблиц: *Водители*, *Работа* и *Виды работ*. Критерии отбора объединены по схеме И.

Поле:	Фамилия	Дата	Наименование
Имя таблицы:	Водители	Работа	Виды работ
Сортировка:			
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:		Between #01.05.2013# And #31.05.2013#	Like "Дальние *"
или:			

Рисунок 3.29 – Запрос на выборку с условием

### Параметрические запросы

**Запрос с параметрами** обеспечивает ввод данных в режиме диалога с пользователем. При выполнении запроса с параметрами на экран последовательно выводится одно или несколько окон диалога, в которые пользователь должен ввести значения параметров (условий) отбора.

Преимущества данного вида запросов состоят в следующем:

- не нужно постоянно модифицировать запрос в режиме *Конструктора*;
- удобно использовать в формах и отчетах, т. к. каждый раз при их открытии Access запрашивает у пользователя требуемый параметр.

Чтобы установить параметр, нужно вместо конкретных данных в *Бланк QBE* в строку **Условие отбора** ввести имя или фразу, заключенную в квадратные скобки [ ]. Например, для поля, которое выводит даты, чтобы задать границы диапазона значений, можно ввести приглашения следующего вида «Введите начальную дату:» и «Введите конечную дату:».

Параметрический запрос может быть легко создан на основе простого запроса.

**Пример 3.5.** Преобразовать запрос из примера 3.4 в параметрический для задания границ проверяемого диапазона даты с клавиатуры.

Скопируем сначала предыдущий запрос: находясь в окне БД на панели объектов Запросы, щелкнем правой клавишей мыши по имени запроса, сформированного в примере 3.4, и из появившегося контекстного меню выберем команду *Копировать*. Далее в области панели объектов с запросами вызовем контекстное меню и выберем команду *Вставить*, после чего в появившемся диалоговом окне *Вставка* введем другое имя

запроса. Откроем созданный запрос в режиме Конструктора и для поля *Дата* в строке *Условие отбора* изменим выражение *Between #01.05.2013# And #31.05.2013#* на *Between [Введите начальную дату:] And [Введите конечную дату:]*. После выполнения запроса необходимо ввести в появившиеся последовательно диалоговые окна начальную и конечную даты временного диапазона, как указано, например, на рис. 3.30.

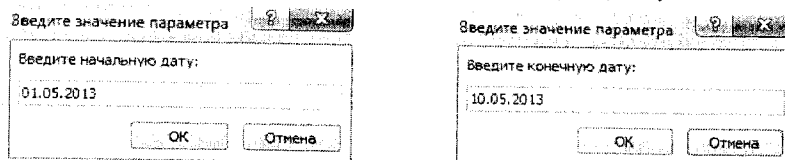


Рисунок 3.30 – Параметрический запрос

### Запросы с вычисляемыми полями

При работе с базами данных часто приходится выполнять различные расчёты на основе имеющихся данных. Вычислительные операции могут выполняться как с использованием полей базы данных, так и с использованием встроенных функций. В этом случае неоценимую помощь может оказать *Построитель выражений*. Построитель выражений находится на вкладке *Работа с запросами* в группе *Настройка запроса* (рис. 3.31).

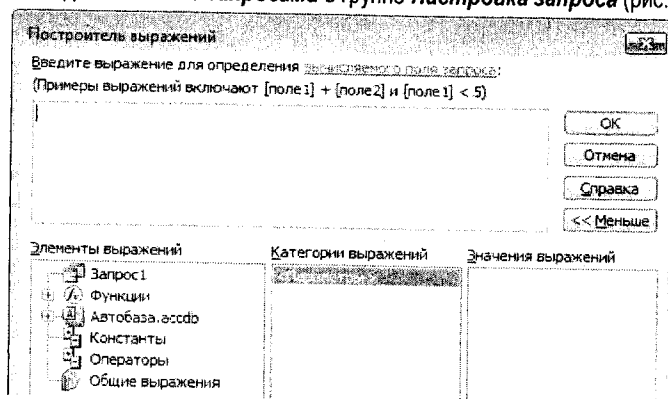


Рисунок 3.31 – Построитель выражений

Построитель выражений содержит несколько окон. Верхнее окно без названия служит для ввода выражений. Выражение начинается с Имени поля, заканчивающегося двоеточием. Например, *Стоимость топлива:*, после которого следует собственно выражение (имена полей, функции, константы и т.п., объединяемые знаками операций).

Список **Элементы выражений** содержит источники данных:

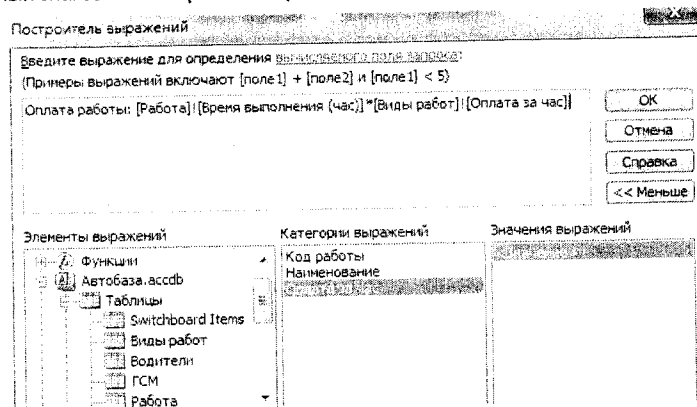
- Текущий запрос – Запрос1. Содержит список всех полей текущего запроса, который появляется после сохранения запроса;
- Функции. В этом списке хранятся встроенные функции СУБД, функции пользователя, созданные для текущей базы данных, функции с веб-служб Интернета;
- Текущая база данных (Автобаза.accdb). Содержит все объекты текущей базы данных;
- Константы – содержит список встроенных логических констант;

- Операторы – содержит список арифметических и логических операторов;
- Общие выражения – содержит список нескольких наиболее часто используемых выражений, например, текущая дата и время, номер страницы и др.

Список **Категории выражений** – содержание списка зависит от выделенного объекта в списке Элементы выражений.

При выделении текущего запроса в списке будут отражены все поля, включённые в текущий запрос. Однако этот список появляется только после сохранения запроса. Из этого списка можно выбирать мышью поля для включения их в выражения. Эти же поля можно выбирать и из списка объектов базы данных (Автобаза), но в этом случае в выражение включается и имя таблицы, из которой выбрано поле, что увеличивает длину выражения и затрудняет его чтение, просмотр и редактирование.

**Пример 3.6.** Вычислить размер оплаты за выполненную работу. На рис. 3.32 приведен пример формирования вычисляемого поля в Построителе выражений. На рис. 3.33 сформированный запрос отображен в режиме Конструктора. После выполнения данного запроса можно заметить, что полученные значения вычисляемого поля не приведены к какому-либо формату. Для того, чтобы изменить формат вычисляемого поля, необходимо, находясь в бланке Конструктора, вызвать контекстное меню для этого поля и выбрать команду **Свойства**, в котором изменить значения нужных свойств. Для поля *Оплата работы* изменяем следующие свойства: Формат поля – денежный, Число десятичных знаков – 0. Сохраним запрос с именем «Оплата работы».



**Рисунок 3.32 – Формирование выражения в Построителе**

Результат выборки для нескольких записей приведен на рис. 3.34.

**Рекомендация.** При создании вычисляемых полей соблюдайте следующий порядок:

- создайте запрос и включите в него все поля, необходимые для вычислений;
- сохраните запрос. Теперь все поля текущего запроса будут находиться в списке **Категорий выражений**.

При выборе Встроенных функций в списке категорий будут указаны категории функций, что ускоряет поиск нужной функции. Однако если Вы не знаете, к какой категории относится интересующая Вас функция, выберите категорию **Все**.

В списке **Значения выражений** выводятся значения функций, констант.

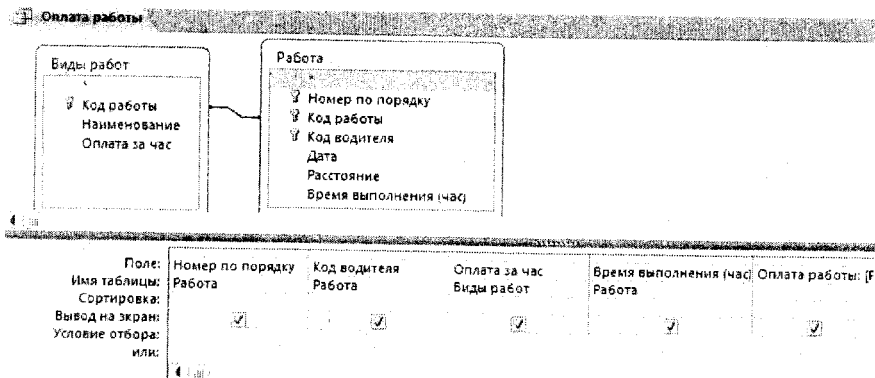


Рисунок 3.33 – Отображение сформированного запроса в режиме Конструктора

Номер по порядку	Код водителя	Оплата за час	Время выполнения (час)	Оплата работы
1	БД0001	20 000р.	1,8	36 000р.
3	БД0002	20 000р.	0,25	5 000р.
5	БД0002	20 000р.	0,3	6 000р.

Рисунок 3.34 – Фрагмент результата выборки

**Пример 3.7.** Вычислить стоимость топлива, затраченного на выполнение работы. Стоимость топлива вычисляется по формуле:  
 $Стоимость\ топлива: [Расстояние] * [Цена\ за\ литр] * [Норма\ расхода\ топлива\ на\ 100\ км] / 100$   
 Сформируем запрос. В запрос включим поля, необходимые для вычисления стоимости топлива: Расстояние, Цена за литр и Норма расхода, а также другие поля, необходимые для идентификации водителя, который выполнял работу, дату выполнения работы, марку топлива (рис. 3.35).

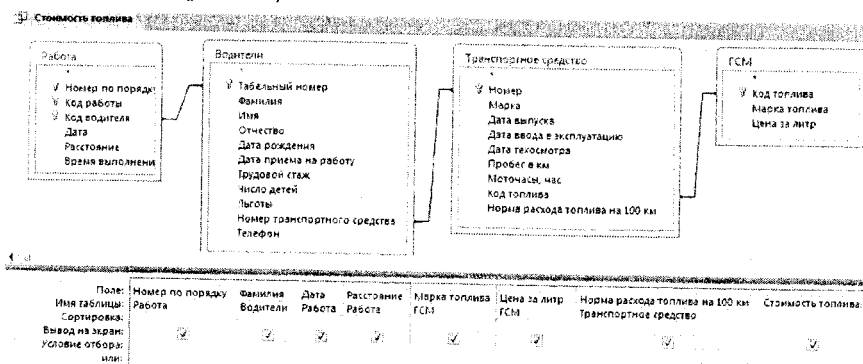


Рисунок 3.35 – Запрос с вычисляемым полем

Рассмотрим примеры использования ряда выражений и встроенных функций.



**Пример 3.8.** Сформировать список водителей с указанием фамилии, имени, отчества и возраста в годах.

**Решение.**

- Сформируем запрос и включим в него необходимые нам поля: Фамилия, Имя, Отчество, Дата рождения. Сохраним запрос и снова откроем его в режиме Конструктора.

- Вычислим число полных лет: Возраст. Для вычисления воспользуемся приближенной формулой:

*Возраст: Int((Date()-[Дата рождения])/366),*

где *Int* – функция, возвращающая целое, меньшее, чем значение аргумента,

*Date()* – текущая дата.

- Для формирования списка сложим поля базы данных. При сложении текстовых полей используется оператор конкатенации – & (амперсанд), можно использовать также оператор сложения "+".

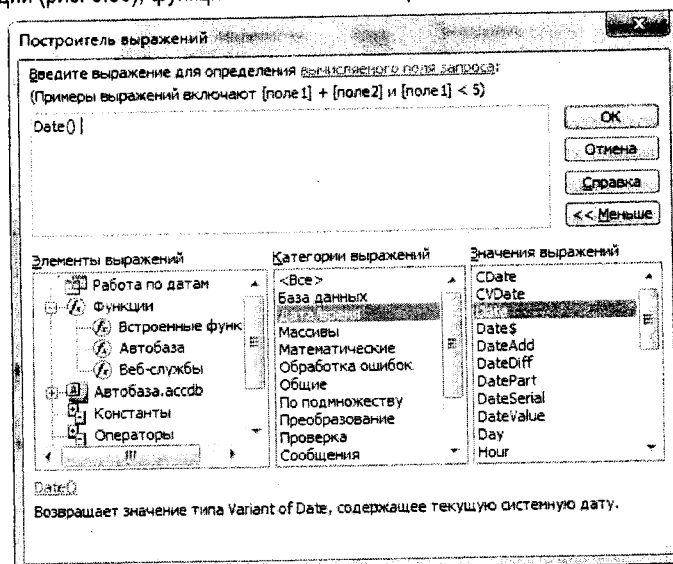
*Список: [Фамилия] & " " & [Имя] & " " & [Отчество] & " " & [Возраст].*

Пробелы между именами полей вставлены с целью исключения слияния значений полей базы данных.

#### Функции работы с датами

Access имеет большое число встроенных функций для работы с датами, которые широко применяются при различных вычислениях: Date, Now, DateAdd, DateDiff, DatePart, Month, MonthName, WeekDay, WeekdayName, Year, Format.

Все функции работы с датами размещаются в категории *Дата/время* меню Встроенные функции (рис. 3.36), функция *Format* – в категории *Текстовые*.



**Рисунок 3.36 – Функции категории Дата/время**

*Date()* – функция не имеет параметров, возвращает текущую дату.

**Now()** – функция не имеет параметров, возвращает текущую дату и время.

**Month(«date»)** – возвращает номер месяца. Цифры от 1 до 12.

**MonthName(«month»; «abbreviate»)** – возвращает название месяца по его номеру. Первый параметр – название месяца, второй параметр – логическое выражение истина (True) или ложь (False) – если "истина", то вывести название в краткой форме, по умолчанию – "ложь", то есть выводить название месяца в полной форме.

**Weekday(«date»; «firstdayofweek»)** – возвращает день недели от 1 до 7. По умолчанию первый день недели – воскресенье, однако можно изменить настройку и указать, какой день недели считать первым, в дополнительном параметре firstdayofweek. Для понедельника в позиции свойства firstdayofweek необходимо указать значение 2.

**WeekdayName(«weekday»; «abbreviate»; «firstdayofweek»)** – возвращает название дня недели по его номеру. Первый параметр – день недели, второй параметр – вывести название недели в краткой форме, третий параметр – то же значение, что и в функции Weekday.

**Year(Дата)** – возвращает год по указанной дате.

**Format(«expression»; «format»; «firstdayofweek»; «firstweekofyear»).**

Здесь «expression» – выражение типа даты, «format» – маска, указывающая, в какой форме выводить результат, например:

- mm – номер месяца двумя цифрами;
- mmmm – полное название месяца;
- d или dd – день месяца;
- dddd – полное название дня недели;
- q – номер квартала;
- ww – номер недели, от 1 до 51;
- y – день года, от 0 до 355;
- yyyy – номер года, от 100 до 9666;
- "firstdayofweek" – первый день недели года;
- "firstweekofyear" – первая неделя в году.

Например: *Format([Дата рождения]; "mm")* – выводится номер месяца.

**Пример 3.9.** Вывести день недели выполнения работы. Можно воспользоваться либо функциями из категории Дата/время (рис. 3.36), либо функцией *Format: ДеньНедели: Format([Дата]; "dddd")*. Запрос представлен на рис. 3.37, а результат – на рис. 3.38.

Поле:	Номер по порядку	Код работы	Код водителя	Дата	День недели: WeekdayName(Weekday([Дата];2))
Имя таблицы:	Работа	Работа	Работа	Работа	
Сортировка:					
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:					

**Рисунок 3.37 – Использование функций работы с датами**

День недели					
Ном	Код работы	Код водителя	Дата	День недели	
1	BP1	БД0001	05.05.2013	воскресенье	
3	BP1	БД0002	17.05.2013	пятница	
4	BP2	БД0002	18.05.2013	суббота	
5	BP1	БД0002	20.05.2013	понедельник	

**Рисунок 3.38 – Использование функций работы с датами**

**DateAdd**(«interval»; «number»; «date») – возвращает новую дату, равную заданной дате, увеличенной на указанное число в соответствии с установленной маской.

Здесь «interval» – маска возвращаемых результатов: *d* – день; *m* – месяц; *q* – квартал; *y* – год; «number» – число, которое необходимо прибавить к дате; «date» – дата, относительно которой ведётся отсчёт.

**Пример 3.10.** Определить дату очередного техосмотра, если он должен проводиться через 6 месяцев.

Решение: `DateAdd("m";6;[Дата техосмотра])`.

**DateDiff**(«interval»; «date1»; «date2»; «firstdayofweek»; «firstweekofyear») – определяет интервал между двумя датами в указанных единицах измерения (днях, месяцах, кварталах, годах). Здесь значение «interval» имеет то же значение, что и в функции `DateAdd`:

Значение	Описание	Значение	Описание
yyyy	Год	w	День недели
q	Квартал	ww	Неделя
m	Месяц	h	Часы
y	Дней в году	n	Минуты
d	День	s	Секунды

«date1» – начальная дата, «date2» – конечная дата.

Параметры «firstdayofweek»; «firstweekofyear» имеют то же значение, что и в функции `Format`.

**Пример 3.11.** Определить, сколько дней прошло после последнего техосмотра.

Решение: `DateDiff("d";[Дата техосмотра];Date())`.

**DatePart**(«interval»; «date»; «firstdayofweek»; «firstweekofyear») – возвращает число дней, недель, месяцев и т.д. в соответствии с значением «interval» для указанной даты.

**Пример 3.12.** Определить номер месяца даты последнего техосмотра.

Решение: `DatePart("m";[Дата техосмотра])`. Результатом вычисления является номер месяца даты проведения техосмотра.

#### Логические функции

Функция `IIf`(«expression»; «truepart»; «falsepart») – возвращает один из двух аргументов в зависимости от результата вычисления выражения. Находится в категории *Управление встроенных функций*.

Здесь *expression* – условное выражение; *truepart* – выражение, соответствующее значению условного выражения «истина»; *falsepart* – выражение, соответствующее значению условного выражения «ложь». Иными словами:

**Если** условное выражение Истинно, **Тогда** функция возвращает результат согласно первому выражению, **Иначе** функция возвращает значение согласно второму выражению.

Функция допускает использование вложенных функций, что позволяет анализировать сложные условия.

В условном выражении могут использоваться логические операторы = равно, < – меньше, > – больше, >= больше или равно, <= меньше или равно, <> – не равно, а также функции OR – ИЛИ (логическое сложение), AND – И (логическое умножение), NOT – НЕ (отрицание). Логические операторы записаны в соответствии с их приоритетами, то есть при сравнении двух переменных сначала будет выполнена проверка на равенство, затем «на меньше» и т.д.

Примеры уловных выражений:

$a = b$ ;  $b > c$ ;  $b < d$ ;  $a \leq b + c$ ;  $d \geq g * u$ ;  $b <> g$ ;  $OR(a < b; c > d)$ ;  $AND(a = b; c = d)$ ;  $NOT g$ .

**Пример 3.13.** Если дата последнего техосмотра более одного года, вывести сообщение о необходимости очередного техосмотра (рис. 3.39).

Поле:	Номер	Марка	Дата техосмотра	Сообщение: Иф(DateDiff("d";[Дата техосмотра];Date()))>=365; Нужен техосмотр," "
Имя таблицы:	Транспорт	Транспортнс	Транспортное средс	
Сортировка:				
Выход на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:				
Илк:				

Рисунок 3.39 – Применение функции If

**Иф(DateDiff("d";[Дата техосмотра];Date()))>=365;"Нужен техосмотр"," "**

В приведённой функции Функция DateDiff возвращает значение дней из текущей даты, которое сравнивается с константой 365. Если условие истинно, то будет выведено сообщение "Нужен техосмотр", иначе будет выведено значение "пусто".

#### Создание запросов на основе других запросов

**Пример 3.14.** Начислить надбавку к оплате за работу в зависимости от трудового стажа: если стаж от 10 до 20 лет, надбавку вычислить в размере 20% от суммы оплаты, если от 20 до 30 лет – 25% от суммы оплаты, если свыше 30 лет – 30% от суммы оплаты.

При создании данного запроса необходимо включить в бланк Конструктора источники данных – это таблица Водители и запрос "Оплата работы" из примера 3.6. (в окне Добавление таблицы его можно найти на вкладке Запросы). Однако этих двух источников будет недостаточно, поскольку в них нет полей, связывающих эти две таблицы, что приведет к избыточности данных в результате выполнения запроса. Поэтому рекомендуется добавить в верхнюю часть бланка Конструктора дополнительно таблицу, поля которой будут связанными с полями обеих таблиц-источников. В нашем примере такой таблицей будет являться подчиненная таблица "Работы". Заметим, что при проектировании запросов, которые впоследствии будут являться источником данных для других запросов, необходимо включать в них ключевые поля используемых таблиц, поэтому при создании запроса "Оплата работы" в примере 3.6 было добавлено в бланк QBE поле Номер по порядку, входящее в составной ключ таблицы «Работы» (рис. 3.40). Более того, после добавления дополнительной таблицы и в случае, когда Access самостоятельно не отобразил связи (линии объединения) между таблицами, их необходимо выполнить «вручную»: находясь в верхней части бланка Конструктора, перетащить поле из подчиненной таблицы или запроса на соответствующее поле подчиненной таблицы, т.е. поле Номер по порядку в запросе «Оплата работы» зажать левой кнопкой мыши, перетащить на поле Номер по порядку таблицы «Работа».

Далее, перед созданием вычисляемого поля, желательно отметить на оси проверяемых значений интервалы критериев, поскольку условий проверки здесь более двух и функция If будет вложена сама в себя несколько раз (количество проверяемых интервалов – 1).

В результате получим вычисляемое поле, для каждого поля можно изменить денежный формат по примеру 3.6:

Надбавка: **Иф([Трудовой стаж] <10; 0; Иф([Трудовой стаж] <20; [Оплата работы] \*0,1; Иф([Трудовой стаж] <30; [Оплата работы] \*0,15; [Оплата работы] \*0,2)))**.

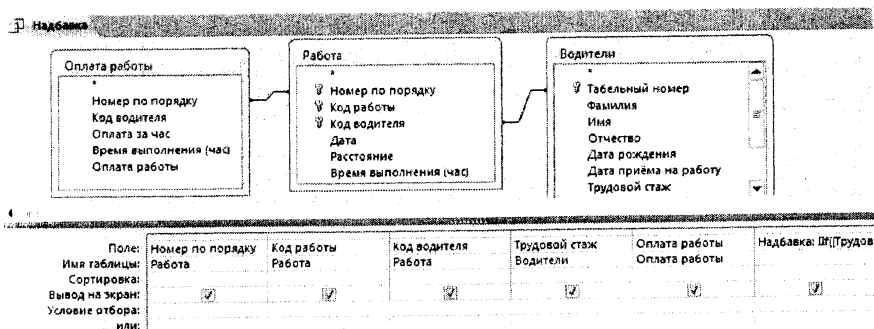


Рисунок 3.40 – Создание запроса на основе других запросов

**Пример 3.15.** Рассчитать общую стоимость работы с учётом стоимости оплаты с надбавкой и стоимости топлива. В данном запросе, по аналогии с предыдущим запросом из примера 3.14, необходимо для добавляемых источников-запросов в верхней части бланка Конструктора «вручную» установить связь между запросами по полю Номер по порядку. Можно добавить главную таблицу «Виды работ» для использования поля Наименование работы в текущем запросе. В этом случае Access может установить линию объединения по полю Код работы самостоятельно.

Вычисляемое поле будет в результате выглядеть следующим образом (рис. 3.41):  
 Полная стоимость работы: [Стоимость топлива]+[Оплата работы]+[Надбавка].

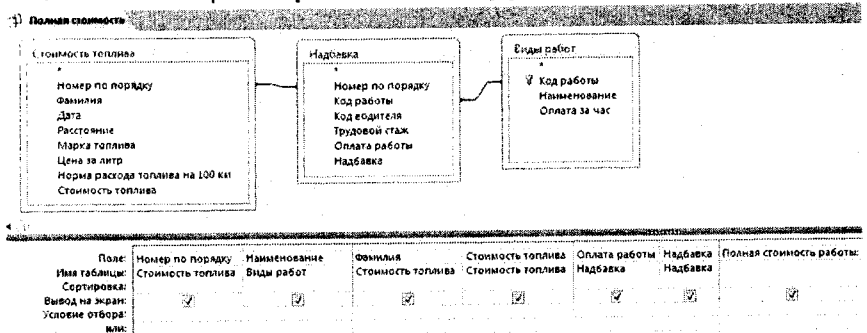


Рисунок 3.41 – Создание вычисляемого поля на основе данных нескольких таблиц

### Групповые операции

Групповые операции позволяют производить подсчёт итоговых значений по столбцам в разрезе параметров, по которым произведена группировка или общий итог.

Порядок создания групповой операции:

- выберите в окне **Объекты Запросы**,
- введите команду **Создание, Конструктор запросов** и выберите таблицу (таблицы), на основании которых будет создаваться запрос – открывается вкладка **Работа с запросами** ленты;
- выберите команду **Итоги** из группы **Показать или скрыть** вкладки **Работа с запросами**. В области QBE бланка запроса появляется опция **Групповая операция** (рис. 3.42).

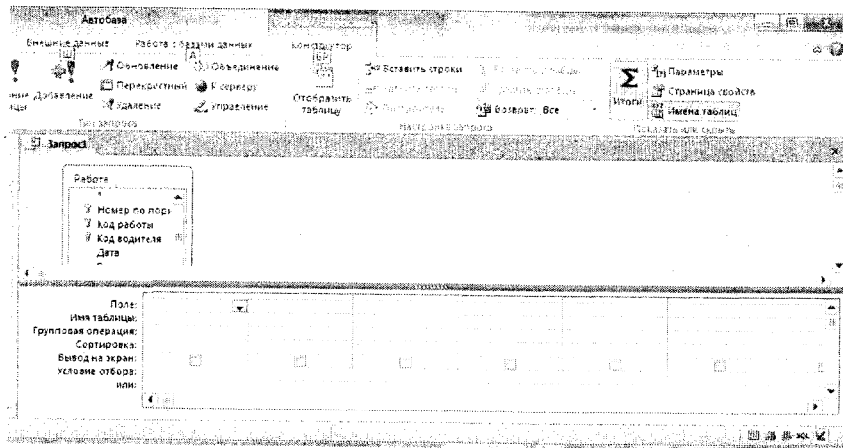


Рисунок 3.42 – Бланк для формирования группового запроса

При активизации поля в строке *Групповая операция* создаётся открывающийся список, в котором содержатся команды и функции: *Группировка* – опция, которая устанавливается автоматически для поля, по которому осуществляется группировка; *Sum* – сумма; *Avg* – среднее значение; *Max* – максимальное значение; *Min* – минимальное значение; *Count* – количество, подсчитывает количество непустых строк, подсчёт ведётся как по числовым, так и по текстовым полям; *StDev* – среднее квадратичное отклонение, *Var* – дисперсия, *First* – первый, *Last* – последний, опция *Выражение* – позволяет использовать вычисляемые поля, опция *Условие* – позволяет задавать условия отбора в групповых операциях.

При использовании опции *Группировка* записи группируются на основе одинаковых значений поля, в котором указана данная опция, и Access в этом случае выполняет вычисления отдельно для каждой группы.

Групповые операции удобнее рассмотреть на примерах.

#### Группировка записей по одному полю

**Пример 3.16.** Подсчитать, сколько всего водителей в автобазе.

– Создать запрос по таблице Водители с одним полем Табельный номер (рис.3.43).

– Ввести команду **Итоги** – в бланке QBE – запроса появляется дополнительная строка – *Групповая операция*, а в поле Табельный номер этой строки появляется значение *Группировка*. Активизируйте ячейку *Группировка* и выберите в списке функцию *Count*. Можно также выполнить сортировку.

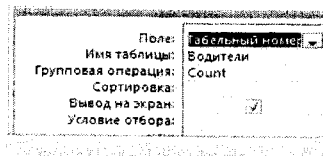


Рисунок 3.43 – К примеру 3.16

**Пример 3.17.** Подсчитать общее, минимальное, максимальное и среднее время, затраченное на выполнение работы каждым водителем.

Запрос формируется на основании таблицы Работа и для подсчёта общего значения времени используется групповая функция *Sum*, для минимального – функция *Min*, для

максимального – Max и для среднего – Avg. После выполнения запроса, для полей, обработанных групповыми функциями (Sum, Min, Max, Avg, Count, StDev, Var), в заголовке столбца выводится подпись, которая включает имя использованной функции и название обрабатываемого поля как, например, в текущем запросе, – Sum-Время выполнения(час). Можно изменить заголовки таких полей, для чего необходимо вызвать контекстное меню для данного поля и изменить свойство Подпись желаемым заголовком, например – Общее время. Можно также в этих свойствах изменить Формат числовых полей – С разделителями разрядов и изменить свойство Число десятичных знаков на 2 (см. Пример 3.6). В результате получим выборку (рис. 3.44) с количеством записей, равным количеству кодов водителей, участвовавших в перевозках.

Код водителя	Общее время	Минимальное время	Максимальное время	Среднее время работы
5Д0001	5,29999995231628	2,3	3	2,64999997615814
5Д0002	10,0999999046326	0,5	7,5	3,36666663487752
5Д0005	14,2999997138977	1,5	8,2	4,76666657129924
5Д0012	23,5	5,5	10	7,83333333333333

Рисунок 3.44 – Групповой запрос с вычисляемыми полями (к примеру 3.17)

**Примечание:** число установленных десятичных знаков не применяется к полям, в которых дробная часть числа является периодической.

#### Группировка записей по нескольким полям

Можно произвести расчеты над сгруппированными данными из нескольких полей и/или из нескольких таблиц, при этом последовательность размещения полей в бланке QBE определяет порядок вложения групп: в первую очередь группировка будет выполнена по крайнему левому полю.

**Пример 3.18.** Какое количество работ по категориям было выполнено в мае месяце?

- Создадим запрос на основе таблицы Работа.
- Вставим в запрос поле Код работы.
- Добавим в запрос вычисляемое поле Месяц, вычисленное по полю Дата таблицы Работа (рис. 3.45).
- Введём команду Итоги и установим Группировку для полей Код работы и Месяц.

Код работы	Месяц: Month([Дата])	Код работы
Работа	Группировка	Работа
Группировка	Группировка	Count

Рисунок 3.45 – Групповая операция по двум полям (к примеру 3.18)

- В Условиях отбора для поля Месяц укажем номер месяца – 5.
  - Добавим в запрос поле Код работы и выберем для него в строке Групповая операция функцию Count.
  - Сохраним запрос.
- Фрагмент результата выполнения запроса приведён на рис. 3.46.

Код работы	Месяц	Count-Код работы
BP1	5	5
BP2	5	1
BP3	5	2

Рисунок 3.46 – Результат запроса (к примеру 3.18)

**Пример 3.19.** Подсчитать, сколько раз выполнялась работа за каждый день по каждому виду работ.

В данном запросе для наилучшего отображения информации выведены не коды работ, а их наименования, для чего при создании запроса в бланк Конструктора добавлены две связанные таблицы: Работа и Виды работ. Запрос и его результат для нескольких записей приведен на рис.3.47. Результат выполнения запроса приведён на рис. 3.48.

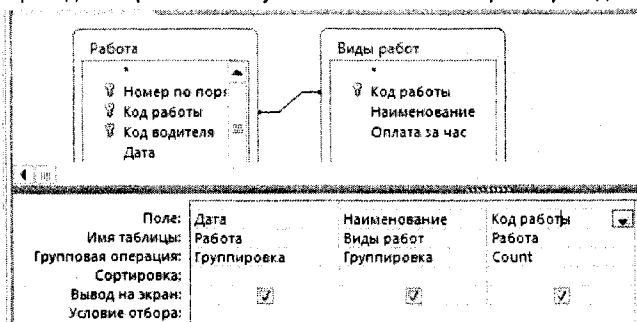


Рисунок 3.47 – Групповая операция по двум полям связанных таблиц (к примеру 3.19)

Дата	Наименование	Количество работ
05.05.2013	Перевозка грузов в городе	1
06.05.2013	Перевозка грузов в городе	2
16.05.2013	Перевозка грузов в городе	1
17.05.2013	Дальние рейсы	1
17.05.2013	Перевозка грузов в городе	1

Рисунок 3.48 – Групповая операция по двум полям связанных таблиц (к примеру 3.19)

#### Обобщающие запросы по всем записям

Такие запросы используют групповые функции без участия опции группировка и подводят итоги по всем записям. В результате выполнения такого вида запросов выводится одна запись. Пример запроса данного вида приведён на рис. 3.43.



### Групповые запросы с условиями

Для формирования условий отбора в групповых запросах могут использоваться поля, по которым выполняется группировка (поля обработанные групповой операцией), а также поля с опциями *Условие* и *Выражение* (к этим полям не применяется групповая операция).

**Пример 3.20.** Подсчитать, сколько водителей в автобазе старше 50 лет?

- Сформируем запрос на базе таблицы Водители и запроса Возраст в годах.
- Включим в запрос поле *Табельный номер* из таблицы Водители и поле *Возраст* из запроса Возраст в годах. Установим в строке *Групповая операция* для поля *Табельный номер* функцию *Count*, а для поля *Возраст* команду *Условие*.
- В строке *Условие отбора* поля *Возраст* установим значение  $>50$  (рис. 3.49). На экран данное поле не выводится.

Поле:	Табельный номер	Возраст
Имя таблицы:	Водители	Возраст в годах
Групповая операция:	Count	Условие
Сортировка:	по возрастанию	
Вывод на экран:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:		$>50$
или:		

Рисунок 3.49 – Групповой запрос с условием (к примеру 3.20)

**Пример 3.21.** Подсчитать, сколько сотрудников имеют двое детей. Пример запроса приведён на рис. 3.50.

Поле:	Табельный номер	Число детей
Имя таблицы:	Водители	Водители
Групповая операция:	Count	Условие
Сортировка:		
Вывод на экран:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:		2

Рисунок 3.50 – Групповой запрос с условием (к примеру 3.21)

**Пример 3.22.** Определить количество поездок в выходные дни, а также общее время их выполнения.

Запрос сформируем на основе таблицы Работа. Для определения дня недели можно использовать функцию `Weekday([Работа]![Дата])`, можно использовать также функции `DatePart(): DatePart("w"; [Работа]![Дата])`; или `Format: Format([Работа]![Дата]; "w")`.

Данные функции возвращают по умолчанию значение 1 для первого дня недели – Воскресенья, и значение 7 для дня недели Суббота. Два критерия (суббота или воскресенье) здесь объединены по схеме ИЛИ (рис. 3.51).

Поле:	Чод работы	Время выполнения ( Weekday([Работа]![Дата])	
Имя таблицы:	Работа	Работа	
Групповая операция:	Count	Sum	Условие
Сортировка:			
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:			1 Or 7
или:			

Рисунок 3.51 – Групповой запрос с условием (к примеру 3.22)

**Пример 3.23.** Сколько было выполнено работ за каждый день заданного месяца?

Данный запрос можно создать как параметрический, вводя номер или название месяца с клавиатуры. При выполнении запроса в первом случае в появившемся окне диалога "Введите значение параметра" вводится число – номер месяца (например, 6), а в другом случае – название месяца (например, июнь) (рис. 3.52, 3.53).

Поле:	Дата	Код работы	Месяц: Month([Работа];[Дата])
Имя таблицы:	Работа	Работа	
Групповая операция:	Группировка	Count	Выражение
Сортировка:	по возрастанию	<input checked="" type="checkbox"/>	
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:			[Введите номер месяца]

**Рисунок 3.52 – Групповой запрос с выражением (к примеру 3.23)**

Поле:	Дата	Код работы	Название_Месяца: Format([Работа];[Дата];"mmmm")
Имя таблицы:	Работа	Работа	
Групповая операция:	Группировка	Count	Выражение
Сортировка:	по возрастанию	<input checked="" type="checkbox"/>	
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:			[Введите название месяца]
или:			

**Рисунок 3.53 – Групповой запрос с выражением (к примеру 3.23)**

**Пример 3.24.** Определить количество поездок отдельно в суббота и в воскресные дни, а также время их выполнения.

Данный запрос подобен запросу в примере 3.22, для формирования условия будем использовать поле группирования, то есть условие отбора применяется к полю обработанному группировкой (рис. 3.54).

Поле:	День недели: DatePart("w";[Работа];[Дата])	Код работы	Время выполнения (час)
Имя таблицы:		Работа	Работа
Групповая операция:	Группировка	Count	Sum
Сортировка:		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Вывод на экран:		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:	1 Or 7		

**Рисунок 3.54 – Групповой запрос с выражением (к примеру 3.24)**

Пример выполнения запроса приведён на рис. 3.55.

День недел	Количество работ	Общее время работ
1	2	2,29999995231628
7	10	10,0999999046326

**Рисунок 3.55 – Результат выполнения запроса (к примеру)**

**Перекрестный запрос**

Перекрёстный запрос создаётся так же, как и запрос на выборку, однако после создания структуры запроса необходимо изменить тип запроса на перекрёстный в группе **Tun** запроса вкладки **Работа с запросами**.

Перекрёстный запрос представляет данные в виде таблицы. В первой колонке указывается поле, которое берется в виде заголовка строк; во второй колонке указывается

поле, которое берется в качестве заголовка столбцов; в третьей колонке указывается вычисляемое поле (значение) и в четвертой колонке указывается, что будет выводиться в качестве заголовков строк результата (рис. 3.56). При выборе типа запроса Перекрестный запрос в шаблоне запроса появляется две строки: групповая операция и перекрестный запрос. В строке Групповая операция для первого и второго столбца указывается Группировка, в третьем и четвертом столбцах – функция. В строке Перекрестный запрос для первого столбца записывается Заголовки строк, второго столбца – Заголовки столбцов, третьего столбца – Значение и четвертого столбца – Заголовки строк.

Четвертый столбец необязателен. Он позволяет получить итоговые данные по вычисляемым полям (сумма, среднее, минимум, максимум и др.).

**Пример 3.25.** Составить отчет пробега транспортных средств по датам.

Для выполнения запроса необходимы две таблицы: Транспортное средство и Работа. Однако между этими таблицами нет связи. Для обеспечения связи между таблицами добавим таблицу Водители. Расположение данных в бланке запроса приведено на рис. 3.56. По строкам выводятся марки транспортных средств, по столбцам – даты выполнения работ. В ячейки таблицы выводится сумма значений пробега в километрах на текущую дату. В четвертом столбце подсчитывается общая сумма пробега автомобиля. Результат запроса приведен на рис. 3.57. Итоговые суммы выводятся во втором столбце. Имя столбцу программа присваивает автоматически.

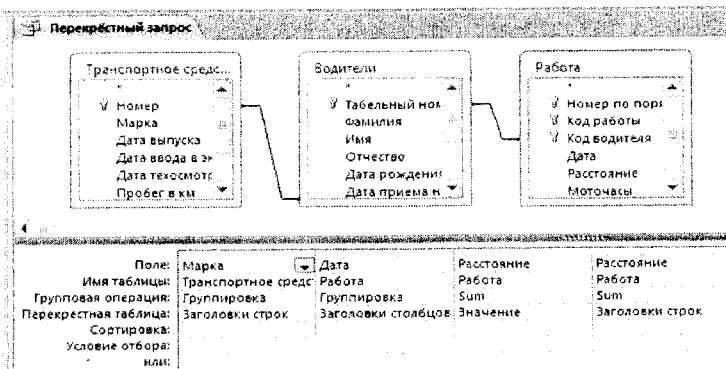


Рисунок 3.56 – Создание перекрестного запроса

Марка	Sum-Рассто	05_05_2013	06_05_2013	15_05_2013	16_05_2013	17_05_2013
Беларусь-80	31,5					6,5
Газель	78					
МАЗ-546	2081				1200	
УАЗ-476	175		150	25		

Рисунок 3.57 – Перекрестный запрос

### Запросы на изменение

К этой группе относятся четыре запроса: Обновление, Добавление, Создание таблицы, Удаление.

### Запрос на обновление

Создать запрос на обновление стоимости бензина Б-72. Старая цена – 6800, новая цена – 7250. Для этого необходимо создать сначала запрос на выборку по таблице ГСМ, в который включить поля *Цена за литр* и *Марка топлива* с условием отбора бензина Б-72 (рис. 3.58). Проверить работу запроса и сохранить его на диске.

Поле:	Цена за литр	Марка топлива
Имя таблицы:	ГСМ	ГСМ
Обновление:	7250	
Условие отбора:		Б-72
или:		

Рисунок 3.58 – Запрос на обновление

Снова откройте запрос в режиме Конструктора и измените *Тип запроса* на *Обновление* – в бланк запроса добавляется строка *Обновление*. Для поля *Цена за литр* в строке *Обновление* укажите новую цену бензина – 7250. Выполните запрос. Программа выведет на экран сообщение сколько записей будет обновлено и предупреждение о том, что после нажатия кнопки "Да" отмена изменений станет невозможна. Сохраните запрос.

### Запрос на создание таблицы

Запрос на создание таблицы ничем не отличается от процедуры создания запроса на выборку. Предположим, что нас интересует, какие работы выполняли водители по датам. В новую таблицу должны войти фамилии водителей, дата выполнения работы и наименование выполненной работы. Для создания запроса потребуется три таблицы: *Водители*, *Работа* и *Виды работ*. Создадим запрос выборку и проверим его (рис. 3.59), а затем изменим тип запроса на *Выборку* на тип запроса на *Создание* таблицы. При выборе команды *Создание таблицы* программа запрашивает имя таблицы. Сохраним запрос (имя созданной таблицы и запроса *не должны совпадать*), а затем выполним его. При выполнении запроса на создание таблицы программа предупреждает, что таблица будет изменена, и после подтверждения на изменение создаёт таблицу и вводит сообщение, в котором сообщает о числе записей в созданной таблице. Результатом выполнения запроса будет новая таблица, которая будет помещена в список таблиц. Полученная таблица может использоваться для подготовки отчётов или участвовать в создании новых запросов.

Поле:	Фамилия	Дата	Наименование
Имя таблицы:	Водители	Работа	Виды работ
Сортировка:			
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:			

Рисунок 3.59 – Запрос на создание таблицы

### Запрос на удаление

Access позволяет удалять из указанной таблицы записи, соответствующие заданным критериям. Пусть требуется удалить запись о выполнении работы водителем Королёвым, код водителя БД0001, 6 мая 2013 года. Создадим запрос на выборку по таблице *Работа*. В запрос включим все поля таблицы *Работа\**, а также условия отбора для полей *Код водителя* и *Дата выполнения работы* (рис. 3.60).

Поле:	Работа*	Код водителя	Дата
Имя таблицы:	Работа	Работа	Работа
Удаление:	Из	Условие	Условие
Условие отбора:		БД0001	#06.05.2013#
или:			

Рисунок 3.60 – Запрос на удаление

Преобразуем запрос-выборку в запрос на удаление, для чего в группе **Тип запроса** укажем **Удаление**. В запросе появляется строка Удаление. В поле Работа\* указано "Из", а в полях *Код водителя* и *Дата* – "Условие". Выполним запрос. Программа выводит на экран предупреждение, что после выполнения запроса восстановление записей будет не возможно.

Запрос на **Добавление** выполняется по похожему сценарию, но требует наличия дополнительной таблицы, из которой будут дополняться записи в текущую таблицу. Поэтому пример не рассматривается.

### Формы

Формы являются основой разработки приложений для диалога. С помощью форм можно загружать данные во взаимосвязанные таблицы, базы данных с документов-источников, просматривать данные, а также корректировать их.

Формы создаются с помощью команд группы **Формы** вкладки **Создание** (рис. 3.61). Имеется несколько режимов создания форм: быстрое создание формы, с помощью **Мастера форм**, с помощью **Конструктора форм**, **Пустая форма**, а также можно использовать специальные **шаблоны форм**.

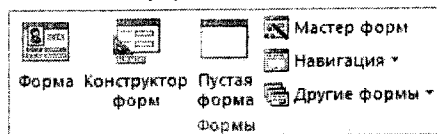


Рисунок 3.61 – Группа **Формы** вкладки **Создание**

### Быстрое создание форм

Это самый простой способ для быстрого создания форм для ввода данных. В форме данного типа отображаются сведения об одной записи. Для создания формы таким способом необходимо выделить таблицу в окне объектов базы данных и выбрать команду **Форма** из группы **Формы** вкладки **Создание**. В созданную форму будут включены все поля базового источника данных, и она откроется в **режиме Макета**. Для перехода в режим формы необходимо ввести команду **Режим формы** в группе **Режимы** вкладки **Главная**. Пример "автоформы" для подчиненной таблицы *Работа* приведен на рис. 3.62. В форме отображаются поля только одной таблицы, хотя существует несколько таблиц, связанных с ней отношением один-ко-многим.

Рисунок 3.62 – Автоформа *Работа*

Если Access обнаруживает таблицу, связанную отношением один-ко-многим с таблицей или запросом, которые будут использоваться для создания формы, то в форму, основанную на связанной таблице или запросе, добавляется *подтаблица* данных. Например, если создается простая форма, основанная на таблице ГСМ (рис. 3.63), и между таблицами ГСМ и Транспортные средства определено отношение один-ко-многим, то автоматически будет создана *составная* форма, в которой в подтаблице данных <sup>2</sup> будут отображаться все записи таблицы Транспортные средства, относящиеся к текущей записи, т.е. к выбранному виду топлива <sup>1</sup>.

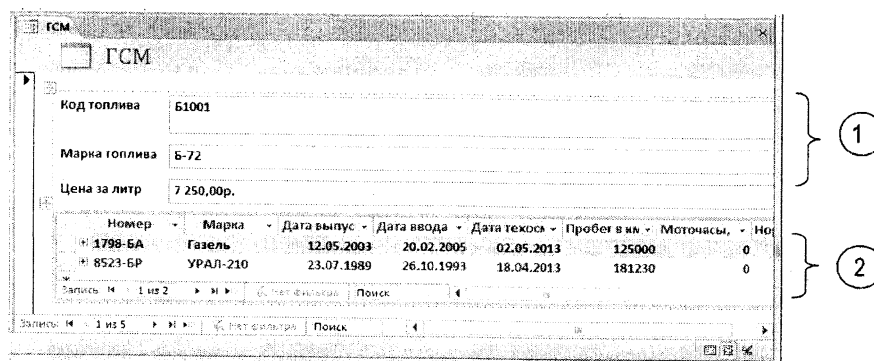


Рисунок 3.63 – Составная форма

Если подтаблица в форме не нужна, ее можно удалить, для чего необходимо перейти в режим Макета, выделить подтаблицу и нажать клавишу DELETE.

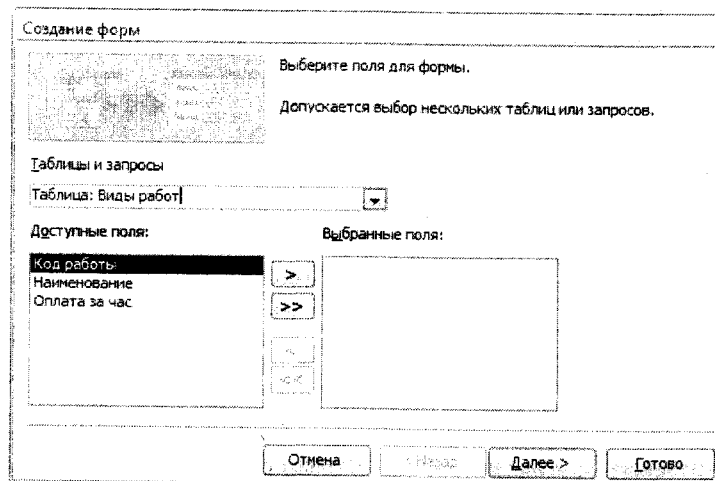


Рисунок 3.64 – Окно диалога Мастера форм

### Создание форм с помощью мастера форм

Это самый простой способ создания форм. Формы, созданные в этом режиме, можно затем доработать в режиме конструктора. После ввода команды на экране появляется окно диалога Мастера форм (рис. 3.64). Мастер создаёт формы за несколько шагов.

На первом шаге необходимо выбрать таблицу или запрос и перенести нужные поля из окна *Доступные поля* в окно *Выбранные поля*. Открывающийся список *Таблицы и запросы* позволяет выбрать требуемую таблицу. В окне *Доступные поля* отображаются поля выбранной таблицы или запроса. Одиночная стрелка служит для переноса выделенного поля из окна *Доступные поля* в окно *Выбранные поля*. Двойная стрелка переносит в окно *Выбранные поля* все поля таблицы или запроса. Стрелки, недоступные на рисунке, служат для переноса полей из окна *Выбранные поля* в окно *Доступные поля*. Для продолжения работы следует нажать кнопку *Далее*. Для завершения работы нажмите кнопку *Готово*.

На втором шаге программа предлагает выбрать внешний вид формы: *в один столбец*, *ленточный*, *табличный* или *выровненный*, а на третьем шаге можно задать имя формы (по умолчанию предлагается то же имя, что и у загруженной таблицы). Активизировать переключатель *Открыть форму для просмотра и ввода данных* и завершить работу, нажав кнопку *Готово*. Пример формы в *Один столбец* приведён на рис. 3.65. Слева выводятся имена полей, а справа от них значения полей. Размеры полей устанавливаются в соответствии с теми параметрами, которые были выбраны на этапе создания таблицы. Внизу формы отображена панель навигации. Стрелки служат для перемещения по записям таблицы. Крайняя правая стрелка ► ☼ служит для добавления записей.

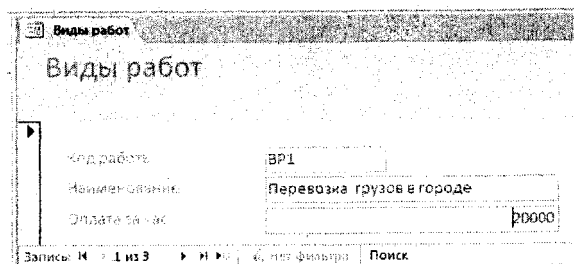


Рисунок 3.65 – Форма в один столбец

### Создание формы в режиме Конструктора

Режим Конструктор позволяет создавать формы "с нуля". После ввода команды *Создание*, *Форма*, *Конструктор форм* открывается окно диалога *Form1* и открывается вкладка *Ленты Конструктор* (рис. 3.66).

Конструктор форм имеет три области: *Область данных* (открыта по умолчанию), *Заголовок* и *Примечание*. Последние две области открываются командой *Заголовок* из группы *Колонтитулы*.

Область *Заголовка* предназначена для размещения заголовка формы, логотипа (эмблемы), даты и времени, итоговых данных.

Область *Примечаний* предназначена для размещения также итоговых данных, номера страницы, итоговых данных и других вспомогательных сведений.

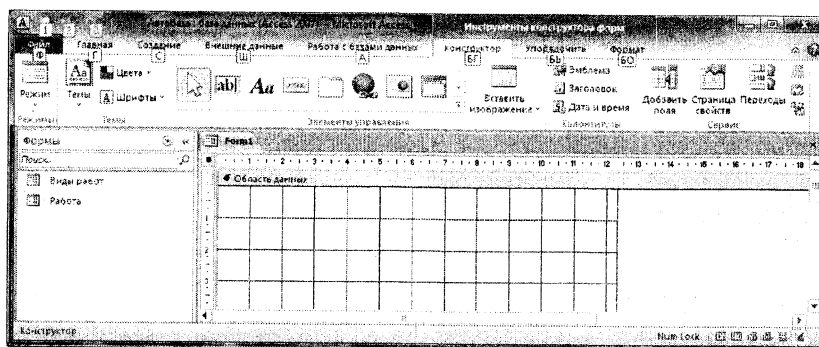


Рисунок 3.66 – Конструктор форм

Область Данных является основной областью данных и предназначена для размещения полей и их имён. В этой области могут размещаться также и вычисляемые поля на основании полей текущей формы.

Во всех областях могут размещаться элементы управления, коллекция которых расположена на панели в группе Элементы управления вкладки Конструктор (рис. 3.67).

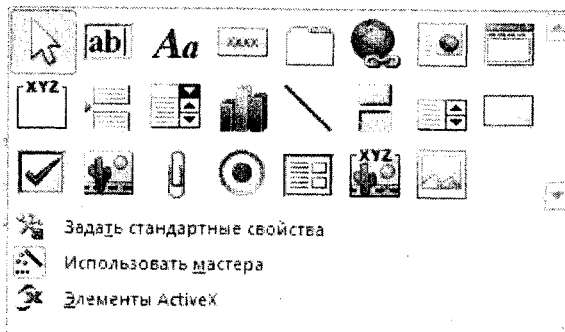








Рисунок 3.67 – Панель элементов управления Access

Рассмотрим назначение некоторых из них.

	<b>Выбор</b> – при активации позволяет выделять объекты, обводя их прямоугольной рамкой.
	<b>Поле</b> – предназначено для ввода формул и текста.
	<b>Надпись</b> – предназначена для ввода текста, используется преимущественно для создания надписей, заголовков и т.п.
	<b>Кнопка</b> – предназначена для создания кнопок управления формой.
	<b>Рамка</b> – служит для группировки объектов.
	<b>Рисунок</b> – предназначена для вставки рисунков.



### Порядок разработки Формы

Форму можно разрабатывать в произвольном порядке. Однако целесообразно вначале связать ее с источником данных – таблицей или запросом.

1. На вкладке **Конструктор** в группе **Сервис** щелкните мышью по кнопке **Страница свойств** – откроется Окно свойств выделенного объекта, в данном случае формы. На вкладке **Все** или **Данные** откройте список в строке **Источник записей** и выберите нужную Таблицу, например, **Водители**.

2. Щелкните по кнопке **Добавить поля** – откроется список полей таблицы. Перетащите поля из списка в область данных формы и упорядочите их. При перетаскивании полей одновременно устанавливаются имена полей и значения полей (рис.3.68). Положением имен полей и их значений можно управлять с помощью маркеров (чёрные квадратик).



Рисунок 3.68 – Поле таблицы, установленное в форму путём перетаскивания

маркеры позволяют управлять отдельно положением имени поля и его значением. Если зацепить объект мышью за границу, то можно перемещать весь объект.

3. Напишите заголовок формы (шаблон заголовка уже присутствует на форме). Отредактируйте заголовок, используя команды группы **Форматирование текста** вкладки **Главная** (рис.3.69).

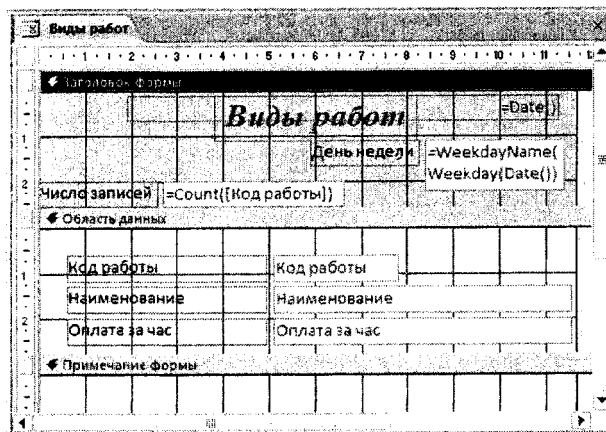


Рисунок 3.69 – Разработка формы в режиме Конструктора

4. Вставьте изображение используя кнопку **Вставить изображение** из группы **Элементы управления** вкладки **Конструктор** или **Эмблему** из группы **Колонтитулы**. При установке даты можно выбрать формат представления данных.

5. Вставьте текстовое поле для вывода дня недели:

- добавьте *Свободное поле* в область заголовка формы, Название поля можно удалить или записать в него текст *День недели*;

- вызовите контекстное меню этого поля и в свойство *Данные* запишите формулу **=WeekdayName(Weekday(Date()))** вручную или с использованием **Построителя выражений**, который вызывается кнопкой “треточие” (“эллипс”).

6. Выведите также в заголовок формы число записей в таблице:
- добавьте Свободное поле в область заголовка формы. В название поля запишите текст *Число записей*;
  - вызовите контекстное меню этого поля и в свойство *Данные* запишите формулу **=Count([Код работы])** вручную или с использованием *Построителя выражений*.
- Примеры разработки формы в режиме Конструктора и в режиме Формы приведены на рис. 3.69 и 3.70 соответственно.

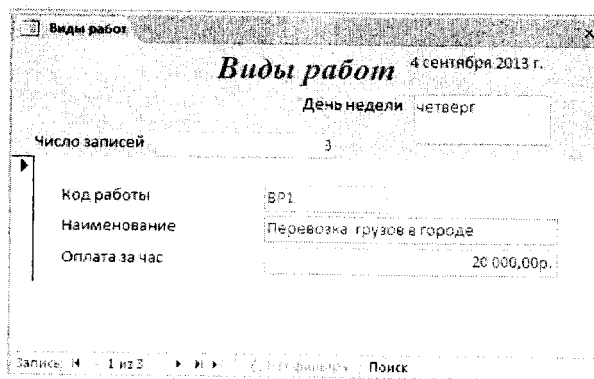


Рисунок 3.70 – Разработанная форма

#### Создание отчетов

Отчёты, так же как и формы, можно разрабатывать несколькими способами, но наиболее просто выполнить это с помощью Мастера форм.

Мастер выполняет работу за несколько шагов:

- откройте вкладку Создание и выберите в группе Отчёты команду **Мастер отчётов**. Появляется окно диалога Создание отчетов;
- выберите из списка таблицу или запрос, отберите поля, необходимые для включения в отчет, и нажмите кнопку Далее;
- следующее окно предлагает выбрать уровни группировки. Выберите, по какому полю будет осуществляться группировка, и щёлкните по кнопке > – в окне отображается поле, выбранное в качестве поля группировки;
- в следующем окне выберите при необходимости порядок сортировки, а также выберите поля, по которым будут подводиться итоги, и функцию для вычисления Итогов, щёлкнув по кнопке **Итоги** (кнопка Итоги присутствует в окне диалога в том случае, когда в списке полей есть числовые поля);
- в следующем окне выберите макет и ориентацию страницы;
- в следующем окне диалога программа предлагает ввести имя отчёта. Имя отчёта предлагается по умолчанию. Если оно не устраивает Вас, введите новое имя. При необходимости можно просмотреть отчёт и вернуться назад для изменения макета отчёта.
- Для просмотра отчёта щёлкните по кнопке Готово.

Отчет, подготовленный с помощью мастера отчетов, можно отредактировать в режиме конструктора.

Пример отчёта приведён на рис. 3.71.

Код работы	Наименование	Оплата за час
BP1	Перевозка грузов в городе	20 000,00р.
Итого для "Код работы" = BP1 (1 запись)		20 000,00р.
BP2	Перевозка грузов в районе	15 000,00р.
Итого для "Код работы" = BP2 (1 запись)		15 000,00р.
BP3	Дальние рейсы	25 000,00р.
Итого для "Код работы" = BP3 (1 запись)		25 000,00р.

17 декабря 2013 г. Стр. 1 из 1

Рисунок 3.71 – Пример отчёта "Виды работ"

#### Автоматизация вычислений в Access

Для решения практических задач в Access имеются макросы и язык программирования Visual Basic for Application (VBA). Макросы – макрокоманды, выполняющие определенные команды языка программирования Access. Язык программирования макросов позволяет решать многие задачи, не прибегая к программированию. В Access более 40 макросов, которые позволяют выполнять практически все команды меню. Кроме того, для решения текущих задач пользователь может разрабатывать собственные макросы.

Язык программирования VBA позволяет разрабатывать сложные вычислительные программы, которые могут использоваться в разрабатываемых приложениях. В рамках данного пособия этот вопрос не рассматривается.

#### Создание макроса

Создание макроса рассмотрим на примере.

**Пример 3.26.** Создать макрос для обновления стоимости топлива.

Расчет выполняется на основе запроса Изменение цены топлива. Поэтому для выполнения задания достаточно создать макрос с одной командой для вызова соответствующего запроса. Если команд в макросе несколько, то они будут выполняться в той последовательности, как они записаны в макросе.

- Откройте вкладку **Создание** ленты, и в группе **Макросы и код** щелкните по кнопке **Макрос**. Открывается окно заготовки макроса Макрос1, справа от которого открыт Каталог макрокоманд, сгруппированных по назначению.
- В группе **Макрокоманды** откройте список **Фильтр, запрос или поиск**, щелкнув по кнопке "+".
- Найдите в списке макрос ОткрытьЗапрос и активизируйте его двойным щелчком мыши. Макрос помещается в макрос пользователя (рис. 3.72).
- Выберите в строке **Имя запроса** из раскрывающегося списка запрос **Изменение цены топлива**.
- Сохраните макрос. При сохранении присвойте ему имя, например, Изменение цены.

#### Использование макроса

Созданный макрос пользователя является объектом Access, и поэтому после создания его имя помещается в список объектов приложения Макросы. Для запуска макроса Изменение цены откройте в списке объектов объект Макросы, найдите и запустите двойным щелчком мыши нужный вам макрос.

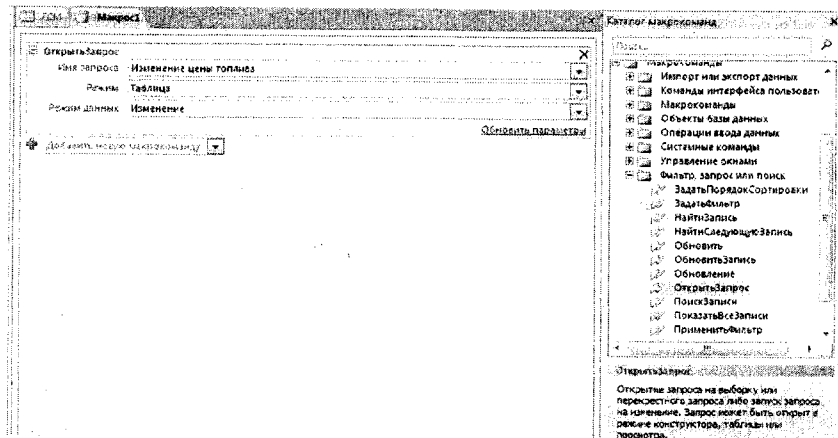


Рисунок 3.72 – Создание макроса пользователя

#### Использование команд управления в макросе

Для управления вычислительным процессом в Каталоге макрокоманд имеется группа Управление. Она содержит четыре группы макросов:

- **Вложенный макрос.** Он разрешает использование в макросе именованных наборов макрокоманд, которые могут быть выполнены только макрокомандой ЗапускМакроса или ПриОшибке;
- **Группа** разрешает группирование макрокоманд и операторов в именованный свёртываемый невыполняемый блок;
- **Если** позволяет создавать условный оператор, выполняемый в случае, если условие Истинно. Дополнительные команды "Добавить блок "Иначе"" и "Добавить блок "Иначе если"" позволяют создавать расширенный оператор Если. Имеется возможность при написании условного оператора использовать Построитель выражения (рис. 3.73);
- **Примечание** позволяет включать в текст программы невыполняемые текстовые комментарии.

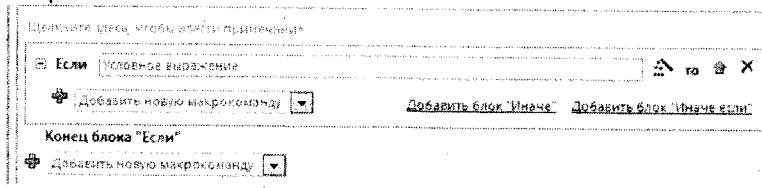


Рисунок 3.73 – Вставка в макрос условного оператора

#### Создание кнопочной формы вручную

Кнопочная форма выполняет роль панели управления приложением. Она содержит кнопки, обеспечивающие вызов других кнопочных форм, а также отдельных объектов – отчетов, форм, макросов.

Создать кнопочную форму можно достаточно просто: открыть пустую форму, поместить в заголовок формы рисунок и установить кнопки для управления открытием нужных объектов. Пример создания кнопочной формы приведен на рис. 3.74.

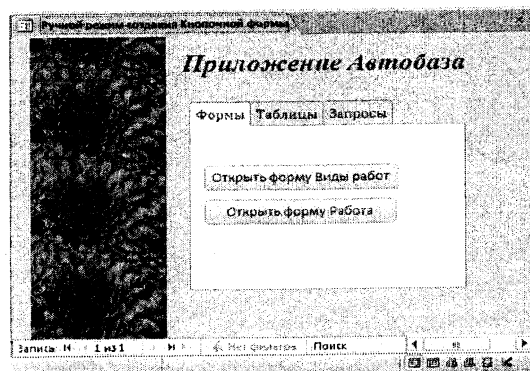


Рисунок 3.74 – Создание кнопочной формы вручную

Откройте вкладку Конструктор.

Установите на форму небольшой растровый рисунок, например цветок, и растяните его на всю высоту формы. Если рисунок заполняет только часть шаблона, то для заполнения всего поля шаблона откройте через контекстное меню свойства рисунка и свойству "Мозаичное заполнение" присвойте значение "Да".

Установите на форму элемент управления **Надпись** и напишите текст названия формы. Выделите объект Надпись и установите параметры шрифта: цвет, начертание, высоту.

Установите на форму элемент управления **Вкладка**. По умолчанию ЭУ содержит две вкладки. Для добавления вкладки вызовите контекстное меню и введите команду **Вставить вкладку**. Откройте через контекстное меню вкладки свойства вкладки и заполните свойство Подпись. Свойство индекс вкладки устанавливается автоматически по мере добавления вкладок.

Следующая операция – установка кнопок.

#### Установка кнопок

Установите элемент управления Кнопка на Вкладку формы. Сразу же после установки кнопки открывается окно диалога *Создание кнопок* (рис. 3.75).

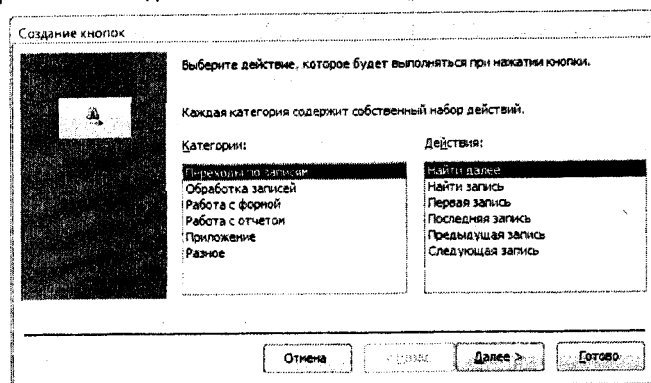


Рисунок 3.75 – Окно диалога Создание кнопок

В группе Категории указаны операции, которые могут быть выполнены с помощью кнопок, а в правой группе указаны соответствующие им действия. Если среди предложенных категорий нет необходимой операции, то выберите категорию *Разное*, которая содержит дополнительные действия: Автонабор номера, Выполнить запрос, Выполнить макрос, Печать таблицы. Среди этих действий есть поистине универсальное действие: Выполнить макрос. То есть, если заблаговременно создать нужный макрос, то его можно назначить кнопке. Выберите нужную категорию и действие, они будут назначены установленной кнопке, и перейдите к следующему шагу кнопкой *Далее*.

Выберите один из предложенных вариантов действий: установите на кнопку рисунок или напишите текст, который будет отображаться на кнопке. Текст на кнопке должен соответствовать выполняемому действию. Перейдите к следующему шагу.

Программа предложит установить имя кнопки. Эту операцию целесообразно опустить. Программа автоматически нумерует все кнопки, устанавливаемые на форму. Это имя используется программой при обращении к кнопкам.

Завершите операцию установки кнопки, щелкнув по кнопке *Готово*.

Порядок создания макросов был рассмотрен в разделе «Создание макроса». Рассмотрим ещё один пример создания макроса для открытия Таблицы для изменений:

- откройте в окне диалога Объекты объект Макросы;
- в группе Макросы и код вкладки Создание ленты введите команду Макрос;
- в окне диалога *Каталог макрокоманд* в группе *Макрокоманды* откройте группу

**Объекты базы данных;**

- выберите из предложенного списка макросов макрос *Открыть таблицу* и в шаблоне Открыть таблицу *выберите таблицу* из списка, например, Водители. В строке *Режим данных* установите "Изменение";

- сохраните макрос. При сохранении присвойте ему имя.

#### **Создание кнопочной формы с помощью Диспетчера кнопочных форм**

##### **Добавление Диспетчера кнопочных форм на Ленту**

Программа Access имеет встроенные средства для создания кнопочных форм – *Диспетчер кнопочных форм*. Однако по умолчанию Диспетчер кнопочных форм не установлен на Ленту. Для добавления Диспетчера кнопочных форм на Ленту необходимо выполнить следующее:

- введите команду *Файл, Настройка ленты*;
- в списке *Основные вкладки* (правый список окна диалога) разверните вкладку

**Создание** щелчком по кнопке "+";

- создайте в этой вкладке новую группу кнопкой *Создать группу* и присвойте ей название, например, *Диспетчеры*;

- в списке *Выбрать команды* (левый список окна диалога) выберите команду *Все команды*;

- найдите в этом списке команду *Диспетчер кнопочных форм*;

- активизируйте щелчком мыши вновь созданную группу *Диспетчеры*;

- активизируйте *Диспетчер кнопочных форм* и добавьте его в активную группу щелчком по кнопке *Добавить*.

##### **Создание главной кнопочной формы**

- Введите команду *Диспетчер кнопочных форм* из группы *Диспетчеры* вкладки **Создание**. Диспетчер кнопочных форм осуществляет проверку наличия кнопочной формы в базе данных и, при отсутствии таковой, выдаёт на экран соответствующее сооб-

щении и предложение создать Кнопочную форму. На запрос программы ответьте "Да". В окне диалога **Диспетчер кнопочных форм** открывается область **Страницы кнопочной формы** со строкой "Главная кнопочная форма" (в этой области формируется список кнопочных форм разных уровней) (рис. 3.76).

- Для создания подчиненных кнопочных форм выберите команду **Создать** в окне диспетчера кнопочных форм. Открывается окно диалога **Создание** (рис. 3.77). В окне диалога **Создание** в поле **Имя страницы кнопочной формы** введите имя подчиненной формы **Таблицы**.
- Создайте аналогично подчиненные кнопочные формы: **Формы** и **Запросы**.
- Закройте Страницу кнопочной формы кнопкой **Заккрыть**.

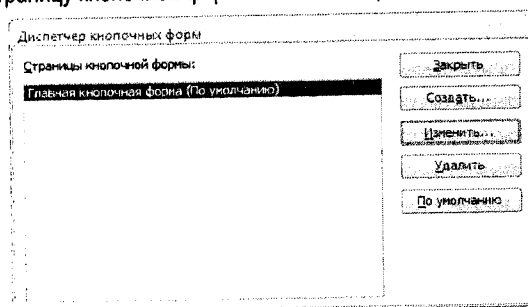


Рисунок 3.76 – Создание подчинённых форм

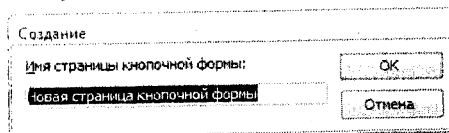


Рисунок 3.77 – Создание подчинённой формы

#### Формирование элементов взаимосвязи кнопочных форм

- В окне диалога **Диспетчер кнопочных форм** (рис. 3.76) выделите строку **Главная кнопочная форма** и щелкните по кнопке **Изменить**. Откроется окно диалога **Изменение страницы кнопочной формы** (рис. 3.78). В окне диалога **Изменение страницы кнопочной формы** измените имя главной кнопочной формы на "Приложение Автобаза".

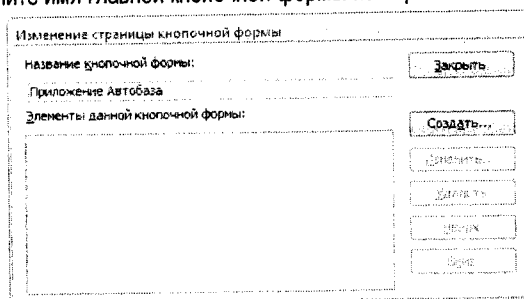


Рисунок 3.78 – Окно диалога **Изменение страницы кнопочной формы**

- Создайте кнопки вызова подчиненной формы в главной форме:
  - щёлкните по кнопке **Создать**. Открывается окно диалога *Изменение элемента кнопочной формы* (рис. 3.79);
  - в окне диалога *Изменение элемента кнопочной формы* выберите в списке **Команда** команду **Перейти к кнопочной форме**, а и в списке **Кнопочная форма** выберите **Таблицы**.

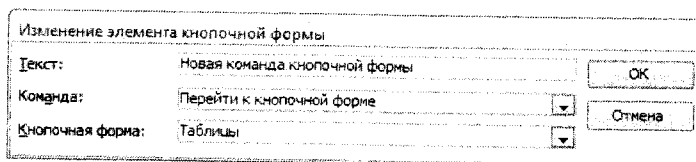


Рисунок 3.79 – Создание кнопки в Главной форме

- В поле **Текст** вместо надписи “Новая команда кнопочной формы” введите подпись для кнопки – **Открытие таблиц**.
- Аналогичным образом создайте кнопки вызова других подчиненных форм и присвойте им названия **Открытие форм** и **Открытие запросов**.
- Для обеспечения возможности редактирования главной кнопочной формы создайте в главной кнопочной форме кнопку с именем **Редактирование кнопочной формы**. Для этого введите в окне диалога *Изменение страницы Кнопочной формы* команду **Создать** и в окне диалога *Изменение элемента кнопочной формы* выберите команду **Конструктор приложения**. В строке **Текст** напишите **Редактирование кнопочной формы**.
- Аналогично создайте кнопку **Выход из приложения** для завершения работы с приложением и присвойте ей соответствующее имя.

В результате выполненной работы в окне диалога *Изменение страницы кнопочной формы* будет сформировано пять команд (рис. 3.80).

Для завершения редактирования главной кнопочной формы нажмите кнопку **Закрыть**.

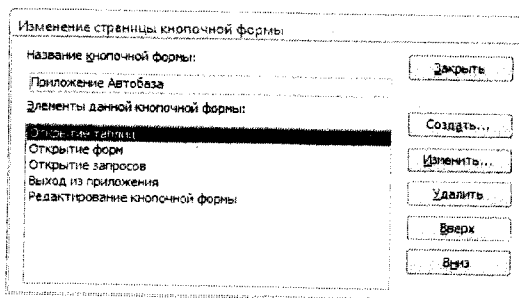


Рисунок 3.80 – Сформированные команды управления Главной кнопочной формы

### Создание команд управления для подчинённых форм

В подчиненных кнопочных формах создайте кнопки вызова соответствующих объектов (таблиц, форм или запросов), а также кнопок для возврата к главной кнопочной форме. Алгоритм создания кнопок в подчинённых кнопочных формах аналогичен созданию кнопок в Главной кнопочной форме с незначительными отличиями.



Выделите подчинённую форму в окне диалога Диспетчер кнопочных форм, например, Таблица и введите команду **Изменить**. В открывшемся окне диалога *Изменение структуры кнопочных форм* введите команду **Создать**. В окне *Изменение элемента кнопочной формы* в строке **Команда** выберите необходимую команду. Для связи создаваемой кнопки с нужным объектом надо выбрать одну из команд, представленных в списке: *открыть форму для добавления, открыть форму для изменения, открыть отчет, выполнить макрос, выполнить программу* и др. (Для открытия Запроса или Таблицы следует выбрать команду **Выполнить макрос**). В следующей строке ввода укажите имя Макроса или соответствующего объекта. На рис. 3.81 приведен пример создания кнопки для открытия таблицы Водители.

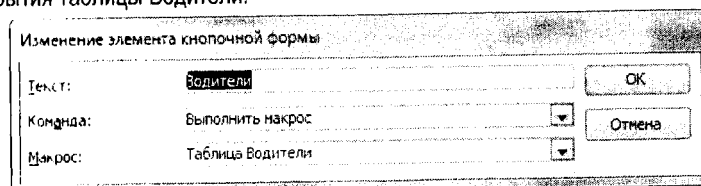


Рисунок 3.81 – Создание кнопок управления объектами в подчинённых кнопочных формах

Для создания кнопки возврата к главной кнопочной форме в строке **Команда** выберите команду **Перейти к кнопочной форме**, в строке **Кнопочная форма** укажите **Приложение Автобаза**, а в строке **Текст** укажите **“Возврат в главную кнопочную форму”** (рис. 3.82).

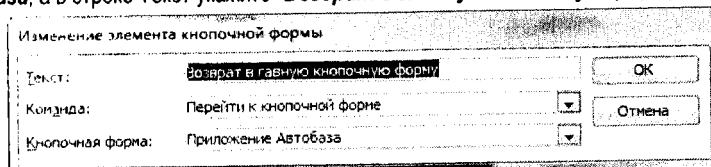


Рисунок 3.82 – Создание кнопки возврата в главную кнопочную форму

Аналогично создаются команды управления для других подчинённых форм.

#### Редактирование кнопочной формы в режиме Конструктора

Откройте контекстное меню кнопочной формы и выберите команду **Конструктор**. В режиме конструктора можно настраивать все элементы формы. Вызовите контекстное меню формы. В списке параметров указаны Свойства формы и Свойства (рис. 3.83). Через данное меню можно настраивать Макет формы. Последняя команда в списке открывает доступ к настройкам всех объектов, установленных на форму. Вид Окна свойств приведён на рис. 3.84. В верхней части окна Расположен раскрывающийся список, в котором содержится список объектов формы.

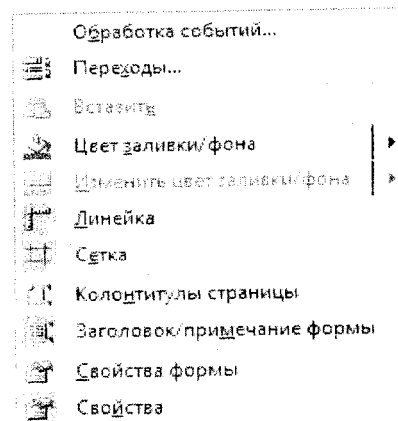


Рисунок 3.83 – Контекстное меню формы

**Примечание:** В список таблиц добавляется таблица *SwitchboardItems*, которая содержит описание свойств кнопок установленных на форму.

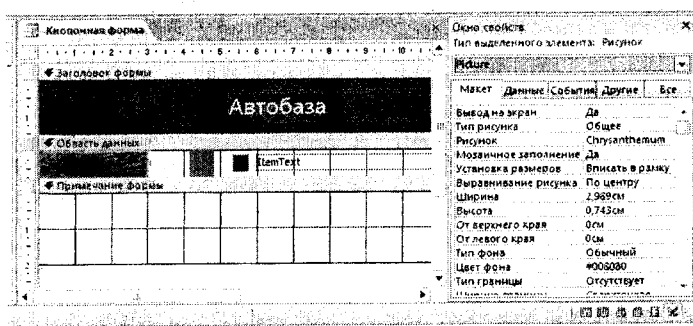


Рисунок 3.84 – Окно свойств объектов, установленных на форму

#### Добавление рисунков на кнопочную форму

- переключите кнопочную форму в режим Конструктора;
- щелкните по области формы слева и нажмите правую кнопку мыши;
- вызовите контекстное меню;
- в окне свойств на вкладке Макет в строке рисунок укажите маршрут для загрузки рисунка;
- измените тип рисунка с внедренный на связанный.

Возможно удобнее установить рисунок в заголовок кнопочной формы, как было рассмотрено в разделе «Разработка формы в режиме конструктора».

Пример разработанной кнопочной формы приведен на рис. 3.85.

После разработки кнопочной формы в окне Объекты в группе объектов Формы появится Кнопочная форма, а в группе объектов Таблица – таблица Switchboard Items (дословно – распределительный пункт). Данная таблица содержит описание структуры Кнопочной формы.

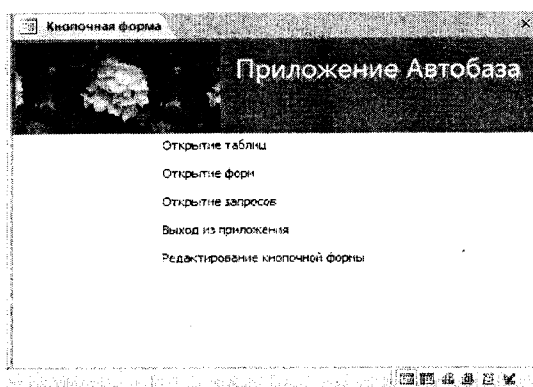


Рисунок 3.85 – Приложение, созданное с помощью диспетчера кнопочных форм и частично доработанное в режиме Конструктора

### Определение параметров запуска приложением

Если вы постоянно или довольно часто пользуетесь созданным приложением, то целесообразно, чтобы при загрузке Access оно загружалось автоматически. Настройка параметров:

- Введите команду **Файл, Параметры, Текущая база данных**.
- В группе **Параметры приложений** запишите в строке Заголовок приложения: **Автобаза**.
- Выберите при необходимости значок для приложения. Выбранный значок будет отображаться в строке заголовка Windows. Если при этом установить флажок "Значок форм и отчетов", то значок будет отображаться на всех вкладках формы и отчета текущей базы данных.
- В списке Форма просмотра укажите имя Кнопочной формы.
- Остальные параметры приложения рекомендуется оставить по умолчанию.

Для вступления в силу установленных параметров перезагрузите базу данных: закрыть базу данных через меню **Файл, Закрыть базу данных**, а затем открыть базу данных через меню **Файл, Открыть**.

- Кроме того, чтобы при запуске Access открывалась последняя использовавшаяся база данных, введите команду **Файл, Параметры, Параметры клиента** и в группе **Дополнительно** установите флажок "Открывать последнюю использовавшуюся базу данных при запуске Access". Выполненные установки вступают в силу после перезагрузки базы данных, как описано выше.

### Список рекомендуемой литературы

1. Бекаревич, Ю.Б. MS Access 2000 за 30 занятий. – СПб.: БХВ-Санкт-Петербург, 2000. – 512 с.: ил.
2. Бекаревич, Ю.Б. Самоучитель Access 2010 / Ю.Б. Бекаревич, Н.В. Пушкина. – СПб.: БХВ-Петербург, 2011. – 432 с.: ил.
3. Гедранович, В.В. Технологии организации, хранения и обработки данных: уч. методич. комплекс. 2-е изд., стереотип / В.В. Гедранович, Ю.В. Змеева – Мн.: Изд-во МИУ, 2006.
4. Гончаров, А.Ю. Access 2003? 2004. – М.: КУДИЦ-ОБРАЗ, 2004. – 272 с.
5. Каратыгин, С.А. Access 2000. – М.: Лаборатория Базовых знаний, 2000. – 376 с.: ил.
6. Кузин, А.В. Разработка баз данных в системе Microsoft Access: учебник / Кузин А.В., Демин В.М. – М.: ФОРУМ: ИНФРА-М, 2005.
7. Левкович, О.А. Основы компьютерной грамотности. – Мн.: ТетраСистема, 2005. – 528 с.: ил.
8. Левчук, Е.А. Технологии организации, хранения и обработки данных. – Мн.: Высш. школа, 2005.
9. Оскерко, В.С. Практикум по технологиям баз данных: учебное пособие / В.С. Оскерко, З.В. Пунчик. – Мн.: БГЭУ, 2004.
10. Попов В.Б. Основы информационных и телекоммуникационных технологий. Системы управления базами данных: учеб. пособие. – М.: Финансы и статистика, 2005.
11. Романова, Ю.Д. Информатика и информационные технологии: учеб. пособие. – 4-е изд. – М.: Эксмо, 2010. – 688 с.: ил.
12. Рудикова, Л. Базы данных. Разработка приложений. – СПб.: БХВ-Петербург, 2006.
13. Савицкий Н.И. Технологии организации, хранения и обработки данных: учеб. пособ. – М.: ИНФРА-М, 2001.