

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Воронежский государственный лесотехнический университет  
имени Г.Ф. Морозова»

Т.П. Новикова    К.В. Зольников

УПРАВЛЕНИЕ ДАННЫМИ

Лабораторный практикум

Воронеж 2017

УДК 004

Печатается по решению учебно-методического совета  
ФГБОУ ВО «ВГЛУ» (протокол № ... от ..... г.)

Рецензенты: ОАО НИИЭТ;

заведующий лабораторией к.т.н. А.И. Яньков  
кафедра электротехники и автоматики ВГАУ имени  
императора Петра I, заведующий кафедрой,  
д.т.н. профессор Д.Н. Афоничев

**Новикова, Т.П.**

Управление данными [Текст] : лабораторный практикум / Т.П. Новикова,  
К.В. Зольников; М-во образования и науки РФ, ФГБОУ ВО «ВГЛУ». –  
Воронеж, 2017. – 124 с.

ISBN .....

В лабораторном практикуме приведены методические рекомендации к проведению лабораторных работ по дисциплине управление данными, практикум содержит краткий теоретический обзор проектирования баз данных, принципы работы в различных реляционных СУБД, основные возможности манипулирования данными, обзор средств администрирования баз данных, а также комплекс лабораторных заданий для усвоения теоретического материала и его практического применения.

Лабораторный практикум предназначен для студентов по направлению подготовки 09.03.02 Информационные системы и технологии (уровень бакалавриата).

УДК 004

ISBN .....

© Новикова Т. П., 2017

© ФГБОУ ВО «Воронежский государственный  
лесотехнический университет  
имени Г.Ф. Морозова», 2017

## ОГЛАВЛЕНИЕ

Введение .....	4
ЛАБОРАТОРНАЯ РАБОТА № 1. Введение в базы данных.....	5
ЛАБОРАТОРНАЯ РАБОТА № 2. Основные объекты баз данных.....	16
ЛАБОРАТОРНАЯ РАБОТА № 3. Модели данных. Проектирование баз данных.....	28
ЛАБОРАТОРНАЯ РАБОТА № 4. Операции реляционной алгебры.....	39
ЛАБОРАТОРНАЯ РАБОТА № 5. Системы управления базами данных. Создание базы данных .....	45
ЛАБОРАТОРНАЯ РАБОТА № 6. Создание связей между таблицами реляционной базы данных .....	57
ЛАБОРАТОРНАЯ РАБОТА № 7. Создание простых запросов .....	62
ЛАБОРАТОРНАЯ РАБОТА № 8. Функции агрегирования .....	65
ЛАБОРАТОРНАЯ РАБОТА № 9. Создание подзапросов .....	68
ЛАБОРАТОРНАЯ РАБОТА № 10. Запросы на изменение .....	72
ЛАБОРАТОРНАЯ РАБОТА № 11. Организация многопользовательского приложения архитектуры «файл-сервер».....	75
ЛАБОРАТОРНАЯ РАБОТА № 12. Защита приложений по управлению данными на уровне рабочих групп .....	86
ЛАБОРАТОРНАЯ РАБОТА № 13. Репликация баз данных .....	93
ЛАБОРАТОРНАЯ РАБОТА № 14. Публикация данных в корпоративной сети и Интернете .....	99
ЛАБОРАТОРНАЯ РАБОТА № 15. Приложения архитектуры «клиент-сервер».....	107
Библиографический список.....	119
Приложение 1 .....	122

## **Введение**

Лабораторный практикум содержит дидактический аппарат, способствующий работе студентов над освоением дисциплины «Управление данными», и соответствует требованиям федерального государственного образовательного стандарта высшего образования.

Лабораторный практикум содержит теоретический обзор средств и методов управления и манипулирования данными средствами реляционных СУБД Microsoft Access и SQL Server, а также комплекс практических заданий, направленных на закрепление теоретического материала.

Каждая работа содержит ряд упражнений, при выполнении которых студенты приобретают опыт работы с перечисленными выше программными средствами.

## ЛАБОРАТОРНАЯ РАБОТА № 1

### Введение в базы данных

**Цель работы:** Ознакомиться с основными терминами и закрепить (получить) навыки работы с реляционными базами данных.

#### **2. Теоретический материал для домашнего изучения.**

**База данных (БД)** – это совокупность данных, организованная по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными.

**Реляционная структура** – это совокупность таблиц, связанных между собой определенными отношениями и предназначенных для хранения данных. В ее основе лежит идея (Эдвард Кодд, сотрудник фирмы IBM, 1970) о том, что любой набор данных можно представить в виде двумерной таблицы. Простейшая реляционная БД может состоять из единственной таблицы, в которой будут храниться все необходимые данные. На практике реляционная БД состоит из нескольких таблиц, связанных между собой по определенным критериям.

#### **Объекты БД**

*Таблицы* – основные объекты БД, в которых хранятся данные и структура базы (поля, их типы и свойства).

В таблицах элементы данных взаимосвязаны между собой в соответствующих строках и столбцах. Строки таблицы соответствуют данным об объекте, информация о котором хранится в БД и называются записями (record). Столбцы называются полями (fields), образующими структуру БД, а информация, содержащаяся в ней, представляется записями. Полям задаются свойства, определяющие вносимые типы данных и операции, совершаемые над данными.

*Запросы* – средство отбора данных из таблиц по определенным критериям.

*Формы* – предназначены для ввода, просмотра или изменения данных в таблицах БД.

*Отчеты* – служат для представления данных в печатном виде и включают специальные элементы их оформления (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т.д.).

*Макросы и модули* – категории объектов, предназначенные для автоматизации повторяющихся операций при работе с системой управления БД и для создания новых функций путем программирования.

*Страницы* – специальные объекты БД, реализованные в последних версиях СУБД Microsoft Access. Физически это объект, выполненный в коде HTML, размещаемый на Web-странице и передаваемый клиенту вместе с ней. Сам объект не является БД, а лишь осуществляет связь Web-страницы с БД, расположенной на сервере.

Любая таблица реляционной базы данных обладает свойствами:

- каждое поле записи имеет единственное значение, а не состоит из группы значений;
- отсутствуют одинаковые записи;
- порядок следования полей и записей не имеет значения.

В таблице могут быть одно или несколько полей, которые однозначно идентифицируют запись таблицы, то есть определяют значения других полей, и называются *ключевыми полями*.

Объявление *первичного ключа* обеспечивает уникальность строк и препятствует вводу повторяющихся блоков данных. Это поле не может содержать одинаковую величину в двух различных записях. *Ключевое поле* помогает *Microsoft Access* наиболее активно организовать поиск, хранение и объединение данных. В *Microsoft Access* можно выделить три типа ключевых полей: *счетчик*, *простой ключ* и *составной ключ*. Указание поля *счетчика* в качестве ключевого является наиболее простым способом создания ключевых полей. Если до сохранения созданной таблицы ключевые поля не были определены, то при сохранении будет выдано сообщение о создании ключевого поля. При нажатии кнопки Да будет создано ключевое поле счетчика. Ключевое поле *Счетчик* автоматически упорядочивает записи по возрастанью.

*Простой ключ* определяется полем, содержащим уникальные значения, такие как коды или инвентарные номера. Ключевое поле не может содержать повторяющиеся или пустые значения. Если устранить повторы путем изменения значений невозможно, то следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ.

В случаях, когда невозможно гарантировать уникальность значений каждого поля, существует возможность создать *составной ключ*, состоящий из нескольких полей. Чаще всего такая ситуация возникает для таблицы,

используемой для связывания двух таблиц в отношении «многие-ко-многим». Если определить подходящий набор полей для составного ключа сложно, следует добавить поле счетчика и сделать его ключевым. Обычно в качестве ключа используются числовые поля.

При определении *связи* между таблицами ключ в одной таблице содержит ссылки на конкретные записи в другой таблице. Ключ, на который имеется ссылка в другой таблице, называют *внешним ключом*. Поле внешнего ключа определяет способ связывания таблиц. Содержимое поля внешнего ключа (тип данных и размер) должно совпадать с содержимым ключевого поля. Эти поля также могут иметь одинаковые имена.

Отношение «*один к одному*» создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы. (Уникальный индекс – индекс, определенный для свойства Индексированное поле значением «Да (Совпадения не допускаются)»).

Отношение «*один ко многим*» создается в том случае, когда только одно из полей является ключевым или имеет уникальный индекс. В отношении «один ко многим» главной таблицей является таблица, которая содержит первичный ключ и составляет часть «один» в этом отношении. Таблица со стороны «много» является подчиненной таблицей. Связующее поле (или поля) в ней с таким же типом информации, как в первичном ключе главной таблицы, является полем внешнего ключа.

Связь с отношением «*многие ко многим*» фактически представляет две связи с отношением «один ко многим» через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, которые являются полями внешнего ключа в двух других таблицах.

Другими словами, связи между таблицами в реляционных базах данных могут быть двух типов:

- *один к одному* – образуется в случае, когда одной записи первой таблицы соответствует одна запись второй таблицы, а одной записи второй таблицы соответствует одна запись первой таблицы;

- *один ко многим* – означает, что одной записи первой таблицы соответствует несколько записей второй таблицы, а одной записи второй таблицы соответствует только одна запись первой таблицы. При этом первая таблица является главной, а вторая – подчиненной.

Для облегчения работы с БД используются системы управления базами данных (СУБД).

**СУБД** – это инструментальные программные средства, предназначенные для создания и ведения баз данных на внешних носителях, а также организации доступа к данным и их обработки.

СУБД обеспечивает сохранность и перемещение данных, возможность их использования с другими программными средствами. Наиболее популярные СУБД: *Microsoft Access*, *FoxPro*, *Oracle Database*, *Microsoft SQL Server*, *MySQL*, *ЛИНТЕР* и др.

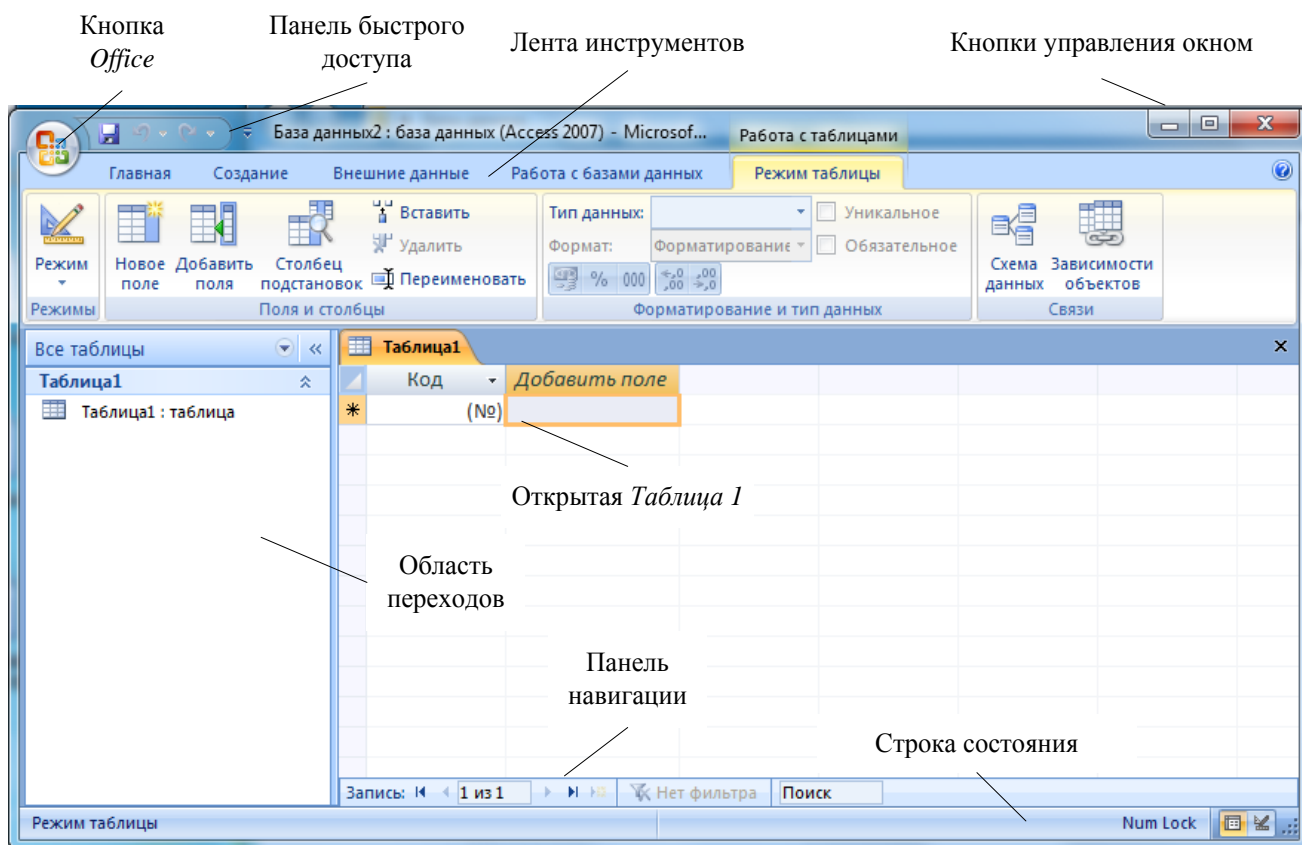


Рис. 1.1. Основные элементы окна *Microsoft Access 2007*

### Начало работы в *Microsoft Access 2007*

Запуск программы *Microsoft Access 2007* осуществляется двойным нажатием соответствующего ярлыка, либо нажатием кнопки *Пуск* и выбором в главном меню Windows пунктов *Все программы* | *Microsoft Office* | *Microsoft Office Access 2007*.

## 3. Лабораторные задания.

### Упражнение 1

Пусть необходимо построить базу данных, содержащую информацию об учебном процессе текущего семестра. База данных будет включать в себя таблицы: *Студенты*, *Дисциплины*, *Учебный план*, *Практика* и *Стипендия*.



## Построение таблиц в режиме конструктора

1. Для того чтобы **создать базу данных** необходимо открыть программу *Microsoft Access 2007*, нажать на кнопку «Office», затем выбрать вкладку *Создать*.

Новой БД присвоить имя *Студенты* (рис. 1.2).

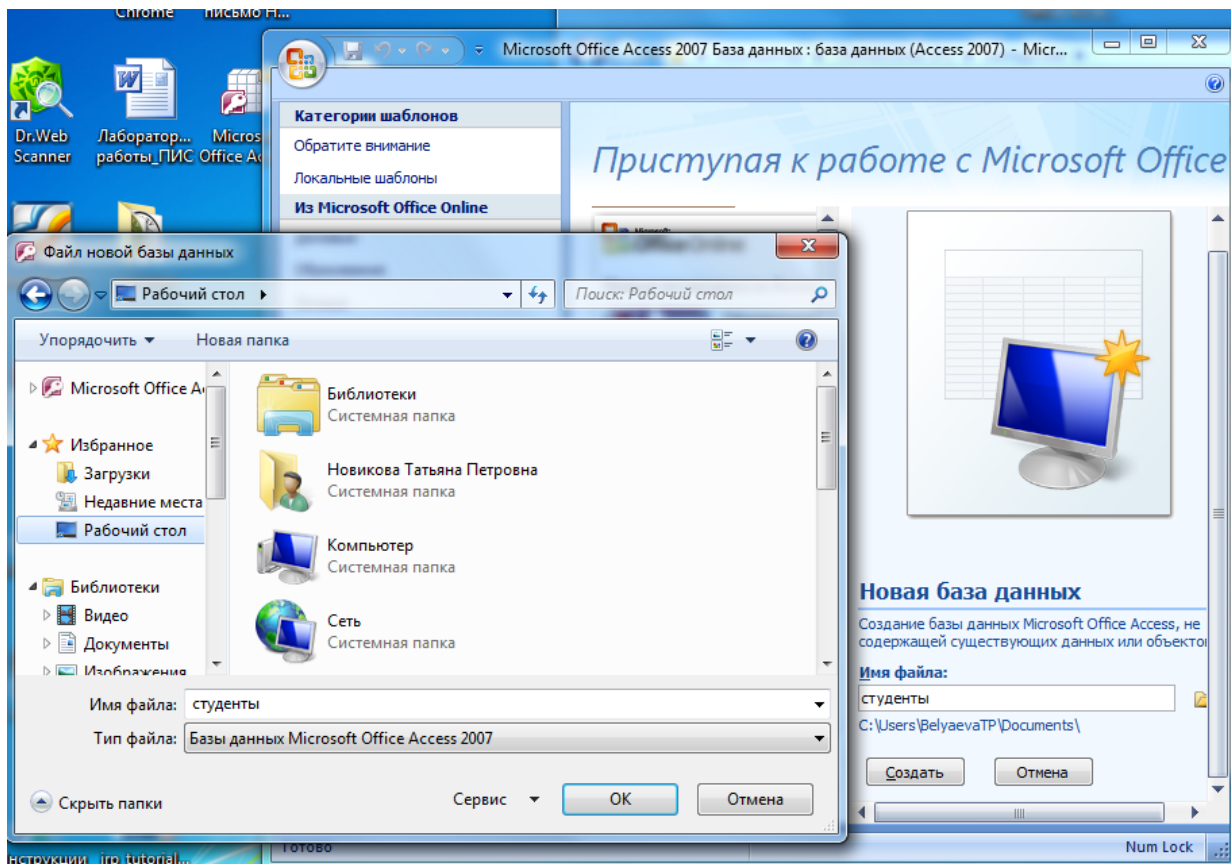


Рис.1.2. Создание БД Студенты

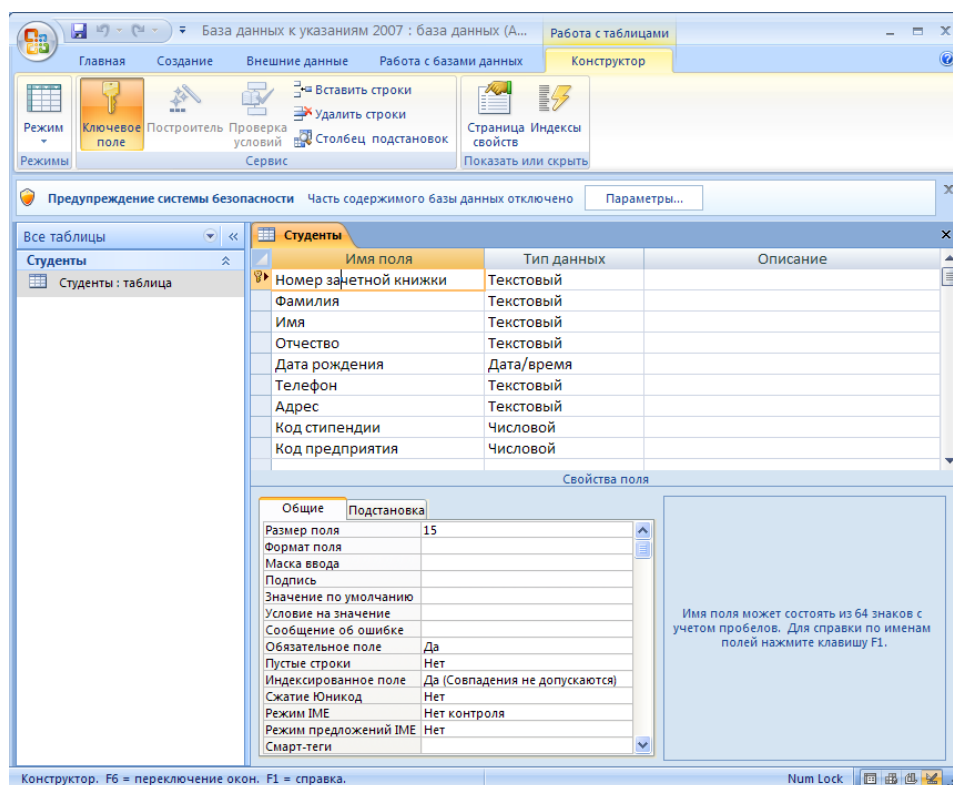
2. Создайте таблицу *Студенты* в режиме *Конструктора*, со следующими полями (рис. 1.3).

Таблица 1

Описание таблицы *Студенты*

Имя поля	Тип данных	Свойство поля
Номер зачетной книжки	Текстовый	<b>Ключевое поле</b>
		Размер поля - 15
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Да (Совпадения не допускаются)
Фамилия	Текстовый	Размер поля - 45
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Да (Допускаются совпадения)
Имя	Текстовый	Размер поля - 20
		Обязательное поле: Да

		Пустые строки: Нет
		Индексированное поле: Нет
Отчество	Текстовый	Размер поля - 20
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
Дата рождения	Дата/время	Формат поля: Краткий формат даты
		Обязательное поле: Да
		Индексированное поле: Нет
		Отображать элемент выбора даты: Для дат
Телефон	Текстовый	Размер поля: 14
		Маска ввода: (999)000-00-00;0;-
		Обязательное поле: Нет
		Пустые строки: Да
		Индексированное поле: Нет
Адрес	Текстовый	Размер поля - 70
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
Код стипендии	Числовой	Размер поля: Длинное целое
		Обязательное поле: Да
		Индексированное поле: Нет
Код предприятия	Числовой	Размер поля: Длинное целое
		Обязательное поле: Да
		Индексированное поле: Нет

Рис. 1.3. Таблица *Студенты* в режиме Конструктора

**3. Аналогичным образом создаются в режиме Конструктора таблицы Дисциплины, Учебный план, Практика и Стипендия (таблицы 2-5).**

Таблица 2

Описание таблицы *Дисциплины*

<i>Имя поля</i>	<i>Тип данных</i>	<i>Свойство поля</i>
Код дисциплины	Счетчик	<b>Ключевое поле</b>
		Размер поля: Длинное целое
		Новые значения: Последовательные
		Индексированное поле: Да (Совпадения не допускаются)
Название дисциплины	Текстовый	Размер поля - 45
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Да (Допускаются совпадения)
Кафедра	Текстовый	Размер поля - 20
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
ФИО преподавателя	Текстовый	Размер поля - 20
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
Количество часов	Числовой	Размер поля: Целое
		Обязательное поле: Да
		Индексированное поле: Нет

Таблица 3

Описание таблицы *Учебный план*

<i>Имя поля</i>	<i>Тип данных</i>	<i>Свойство поля</i>
Код дисциплины	Числовой	Размер поля: Длинное целое
		Обязательное поле: Да
		Индексированное поле: Нет
Номер зачетной книжки	Текстовый	Размер поля - 15
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
Вид итогового контроля	Текстовый	Размер поля - 20
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет

Таблица 4

Описание таблицы *Практика*

<i>Имя поля</i>	<i>Тип данных</i>	<i>Свойство поля</i>
Код предприятия	Счетчик	<b>Ключевое поле</b>
		Размер поля: Длинное целое
		Новые значения: Последовательные
		Индексированное поле: Да (Совпадения не допускаются)
Название предприятия прохождения	Текстовый	Размер поля - 40
		Обязательное поле: Да
		Пустые строки: Нет

практики		Индексированное поле: Нет
Адрес предприятия	Текстовый	Размер поля - 70
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет

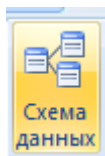
Таблица 5

Описание таблицы *Стипендия*

Имя поля	Тип данных	Свойство поля
Код стипендии	Счетчик	<b>Ключевое поле</b>
		Размер поля: Длинное целое
		Новые значения: Последовательные
		Индексированное поле: Да (Совпадения не допускаются)
Вид стипендии	Текстовый	Размер поля - 40
		Обязательное поле: Да
		Пустые строки: Нет
		Индексированное поле: Нет
Сумма	Денежный	Число десятичных знаков: 2
		Обязательное поле: Да
		Индексированное поле: Нет

**4. Установите связи между созданными таблицами.**

4.1. Для установки связей между таблицами в *Microsoft Access 2007* существует окно *Схема данных*, которое открывается нажатием кнопки



на вкладке *Работа с базами данных*.

Как правило, связывается ключевое поле одной таблицы с соответствующим ему полем другой таблицы, которое называется полем внешнего ключа. Связанные поля могут иметь разные имена, однако у них должны быть одинаковые типы данных и одинаковые значения свойств.

4.2. Используя команду *Отобразить таблицу* на вкладке *Связи*, добавьте все созданные таблицы.

4.3. Установите курсор в любую из таблиц на поле, по которому будет установлена связь, и «перетащите» это поле на связующее поле другой таблицы (рис. 1.4).

4.4. Активизируйте флажок *Обеспечение целостности данных*. Данное действие позволит предотвратить случайное удаление или изменение связанных данных.

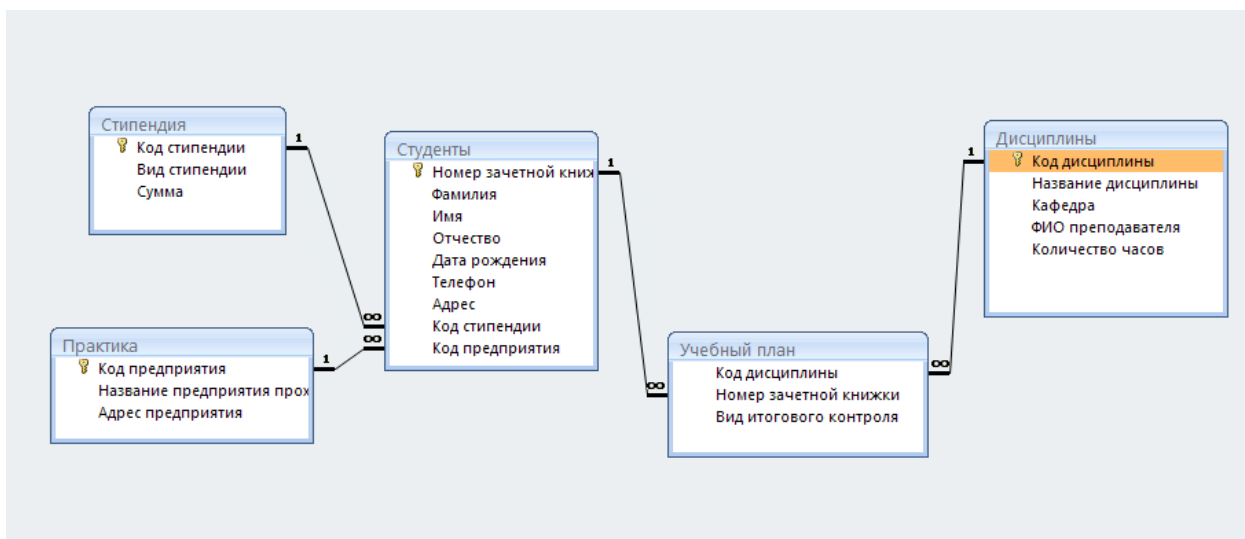


Рис. 1.4. Схема данных разрабатываемой БД

**5. Заполните данными таблицы *Стипендия* и *Практика*, выполнив следующие действия:**

5.1. В области переходов дважды щелкните по таблице *Стипендия*. Выбранная таблица будет открыта в режиме таблицы, заполните ее согласно данным из таблицы 6.

Обратите внимание, что у *Кода стипендии* тип данных **Счетчик**, данное поле заполняется автоматически.

Таблица 6

Данные таблицы *Стипендия*

Код стипендии	Вид стипендии	Сумма
1	Обычная	1000
2	Повышенная	2000
3	Социальная	3540
4	Президентская	9500

5.2. Аналогично откройте таблицу *Практика* в режиме таблицы и заполните ее согласно данным из таблицы 7.

Обратите внимание, что у *Кода предприятия* тип данных **Счетчик**, данное поле заполняется автоматически.

Таблица 7

Данные таблицы *Практика*

Код предприятия	Название предприятия прохождения практики	Адрес предприятия
1	ОАО «Ангстрем»	ул. Торпедо д.43
2	ОАО ХК «Мебель Черноземья»	ул. Богдана Хмельницкого, 51а
3	ООО «ДОЗ Придонской»	п. Придонской, ул. Латненская, 13

**6. Заполните таблицу *Студенты* в соответствии с данными таблицы 8.**

Таблица 8

Данные таблицы *Студенты*

Номер зачетной книжки	Фамилия	Имя	Отчество	Дата рождения	Телефон	Код стипендии	Код предприятия
					Адрес		
1	2	3	4	5	6	7	8
ПМ2- 151.01	Авдеев	Игорь	Игоревич	05.08.1998	(950)325-62-56 ул. Докучаева д.6 кв. 85	01	01
ПМ2- 151.02	Варава	Анна	Викторовна	05.03.1998	(920)520-13-10 ул. Морозова д. 31 кв. 20	02	02
1	2	3	4	5	6	7	8
ПМ2- 151.03	Гаврилов	Семен	Иванович	02.10.1999	(920)528-85-10 ул. Морозова д. 30б кв. 30	03	03
ПМ2- 151.04	Светлова	Анна	Игоревна	02.03.1998	(950)560-44-52 ул. Лермонтова д.45 кв.15	04	01
ПМ2- 151.05	Иванов	Петр	Сергеевич	01.10.1998	(950)560-33-52 ул. Ломоносова д.5 кв.15	01	01
ПМ2- 151.06	Светлов	Иван	Алексеевич	10.05.1999	(920)333-15-82 ул. Ломоносова д.87 к.555	02	02
ПМ2- 151.07	Кротов	Алекс андр	Сергеевич	07.08.1999	(904)214-15-82 ул. Плеханова д.87 к.11	03	03
ПМ2- 151.08	Бурмист рова	Юлия	Андреевна	04.05.1998	(910)218-44-02 ул. Театральная д.7 к.25	01	01
ПМ2- 151.09	Старых	Ирина	Сергеевна	02.05.1999	(910)152-48-53 ул. Театральная д.52 к.5	02	02
ПМ2- 151.10	Юрьева	Алла	Виктор на	02.09.1998	(904)125-08-82 ул. Ломоносова д.87 к.187	03	03
ПМ2- 151.11	Беликов	Серге й	Иванович	02.10.1999	(920)528-85-10 ул. Морозова д. 10 кв. 12	01	01
ПМ2- 151.12	Сизов	Виктор	Петрович	05.07.1998	(915)105-02-62 ул. Ломоносова д.87 к.253	02	02
ПМ2- 151.13	Андреев	Василий	Иванович	08.09.1998	(950)144-33-82 ул. Ломоносова д.145б кв.15	03	03
ПМ2- 151.14	Козлова	Юлия	Борисовна	01.10.1999	(910)112-91-02 ул. Тимирязева д.8 к.25	04	01
ПМ2- 151.15	Орлов	Иван	Федорович	02.09.1998	(904)105-42-39 ул. Ломоносова д.87 к.136	01	02

7. В результате заполнения таблиц *Стипендия*, *Практика*, *Студенты*, при открытии связанной таблицы *Стипендия* получаем раскрывающиеся списки (рис. 5).

Код стипен.	Вид стипен.	Сумма	Имя	Отчество	Дата рожд.	Телефон	Адрес	Код предпр.
1 обычная 1 000,00р.								
PM2-151.05	Иванов	Петр	Сергеевич	18.05.1999	(920)526-66-77	ул. Тимирязев		1
PM2-151.01	Авдеев	Игорь	Игоревич	05.08.1998	(950)325-62-56	ул. Докучаева		1
2 повышенная 2 000,00р.								
PM2-151.02	Варава	Анна	Викторовна	05.03.1998	(920)520-13-10	ул. Морозова		2
3 социальная 3 540,00р.								
4 президентская 9 500,00р.								
* (№)								

Рис. 5. Вид таблицы *Стипендия* после заполнения связанных таблиц

8. Заполните таблицу *Дисциплины* в соответствии с данными таблицы 9. Обратите внимание, что у *Кода дисциплины* тип данных **Счетчик**, данное поле заполняется автоматически.

Таблица 9

Данные таблицы *Дисциплины*

Код дисциплины	Название дисциплины	Кафедра	ФИО преподавателя	Количество часов
1	Математика	Математики	Еремин П.П.	72
2	ИТ в менеджменте	Информатики	Петрова М.И.	72
3	Менеджмент	Экономики	Федоров К.А.	144
4	Экономическая теория	Экономики	Исакова А.В.	72
5	Английский	Иностранного языка	Рагозина Л.И.	54
6	Психология	Гуманитарных наук	Коротких А.И.	36
7	История	Гуманитарных наук	Власов А.Н.	54

9. Заполните таблицу *Учебный план* в соответствии с данными таблицы 10.

Таблица 10

Данные таблицы *Учебный план*

Код дисциплины	Номер зачетной книжки	Вид итогового контроля
1	PM2-151.01	Экзамен
2	PM2-151.02	Экзамен
3	PM2-151.03	Экзамен
4	PM2-151.04	Зачет
5	PM2-151.05	Зачет
6	PM2-151.06	Зачет
7	PM2-151.07	Экзамен
1	PM2-151.08	Экзамен

2	ПМ2-151.09	Экзамен
3	ПМ2-151.10	Экзамен
4	ПМ2-151.11	Зачет
5	ПМ2-151.12	Зачет
6	ПМ2-151.13	Зачет
7	ПМ2-151.14	Экзамен
1	ПМ2-151.15	Экзамен
2	ПМ2-151.01	Экзамен
3	ПМ2-151.02	Экзамен
4	ПМ2-151.03	Зачет
5	ПМ2-151.04	Зачет
6	ПМ2-151.05	Зачет
7	ПМ2-151.06	Экзамен
1	ПМ2-151.07	Экзамен

#### 10. Сохраните созданную базу данных.

#### 4. Контрольные вопросы:

1. Дайте определения базе данных и системе управления базами данных
2. Дайте определение реляционной БД.
3. Функции СУБД *Microsoft Access*.
4. Этапы проектирования и создания БД.
5. Основные объекты БД.
6. Назначение и определение первичного ключа.
7. Какие типы данных можно задать полям таблицы в *Access*?
8. Понятие связей «один-к-одному», «один-ко-многим» и «многие-ко-многим».
9. Как установить связь между таблицами?
10. Как определяется структура таблицы в программе *Microsoft Access*?

### ЛАБОРАТОРНАЯ РАБОТА № 2

#### Основные объекты баз данных

**Цель работы:** Изучить процесс реализации различных видов запросов и освоить основные приемы работы с формами и отчетами в *Microsoft Access*.

#### 2. Теоретический материал для домашнего изучения.

Запросы используются для отбора, анализа и изменения данных из одной или нескольких таблиц базы данных *Microsoft Access*. Например,



можно использовать запрос для выбора данных из таблицы по определенному условию, выполнения расчетов, объединения данных из разных таблиц или добавления, изменения или удаления данных в таблице.

Запросы, используемые для извлечения данных из таблицы или выполнения расчетов, называются *запросами на выборку*. Запросы, используемые для добавления, изменения или удаления данных, называются *запросами на изменение*.

### Создание запросов с параметром

*Запрос с параметром* – это запрос, при выполнении которого появляется диалоговое окно для ввода конкретного значения, используемого для отбора данных.

### Работа с формами

*Форма* – это настраиваемое диалоговое окно, которое используется для ввода, изменения и отображения данных из таблицы или запроса базы данных.

По сравнению с режимом таблицы, формы предоставляют более удобный способ просмотра и правки данных в таблицах базы данных. Они позволяют выполнять проверку корректности данных при вводе, проводить вычисления, обеспечивают доступ к данным в связанных таблицах.

В *Access* существуют следующие способы создания новых форм в уже существующей базе данных:

- автоматическое создание простой формы;
- автоматическое создание разделенной формы;
- создание формы с помощью *Мастера форм*;
- создание формы с дополнительными элементами;
- создание пустой формы;
- создание формы в режиме конструктора.

Форму можно создавать с помощью инструментов, находящихся на вкладке *Создание* в группе *Формы* (рис. 2.1).

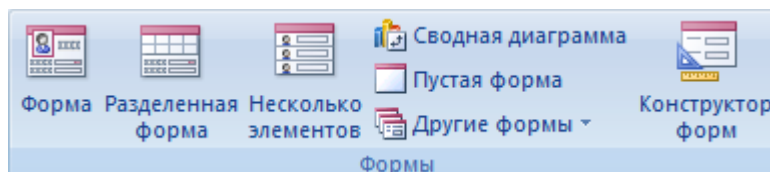


Рис. 2.1. Группа *Формы*

### *Создание простой формы*

Для автоматического создания *простой формы* для существующей таблицы базы данных нужно воспользоваться инструментом *Форма* на вкладке *Создание* в группе *Формы*. При использовании этого средства в новой форме размещаются все поля выбранной таблицы базы данных. Сразу после создания новой формы можно начать ее использование либо при необходимости изменить ее в режиме макета или конструктора.

### ***Создание разделенной формы***

*Разделенная форма* отображает данные, содержащиеся в таблице, одновременно в двух представлениях – в режиме формы и в режиме таблицы. Эти два представления связаны с одной и той же таблицей базы данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в каждой части формы.

### **Создание отчетов**

*Отчет* – это способ представления данных в удобном формате в виде печатного документа, доступного только для чтения.

С помощью отчета можно отображать данные, содержащиеся в таблицах или запросах, но их нельзя изменять, в отличие от других объектов базы данных (в частности, форм и запросов). Наряду с данными, извлекаемыми из таблиц и запросов, в любом отчете обязательно содержится информация о структуре и свойствах страниц отчета и его отдельных элементов (подписей, заголовков, рисунков).

К основным способам создания нового отчета в текущей базе данных *Access* относятся:

- автоматическое создание отчета с помощью средства *Отчет*;
- создание отчета с помощью *Мастера отчетов*;
- создание отчета с использованием средств *Пустой отчет*;
- создание отчета в конструкторе отчетов;
- создание наклеек с помощью *Мастера наклеек*.

Отчет можно создавать с помощью инструментов, находящихся на вкладке *Создание* в группе *Отчеты* (рис. 2.2)

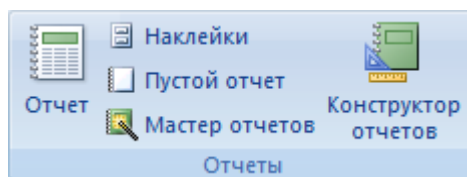


Рис. 2.2. Группа *Отчеты* на вкладке *Создание*

### 3. Лабораторные задания.

#### Упражнение 1

##### Создание простого запроса на выборку

Создайте запрос, выводящий список студентов, отсортированный по фамилии, имени, дате рождения и номеру группы .

1. На вкладке *Создание* в группе *Другие* нажмите кнопку *Мастер запросов*. На экране появится диалоговое окно *Мастера запросов* (рис. 3).

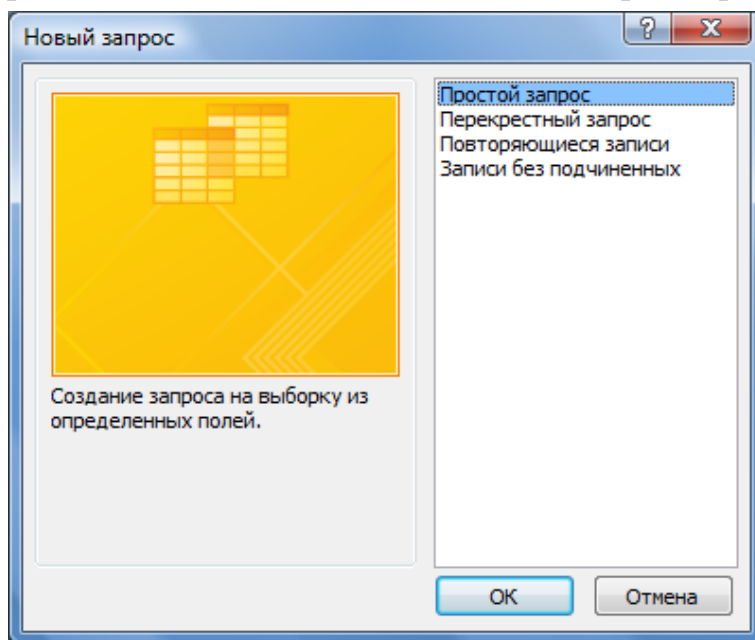


Рис. 2.3. Окно *Мастера запросов*

2. В появившемся окне выберите *Простой запрос* и нажмите кнопку *ОК*. На экране появится следующее окно *Мастера* (рис. 4). В раскрывающемся списке *Таблицы и запросы* выберите таблицу *Студенты*.

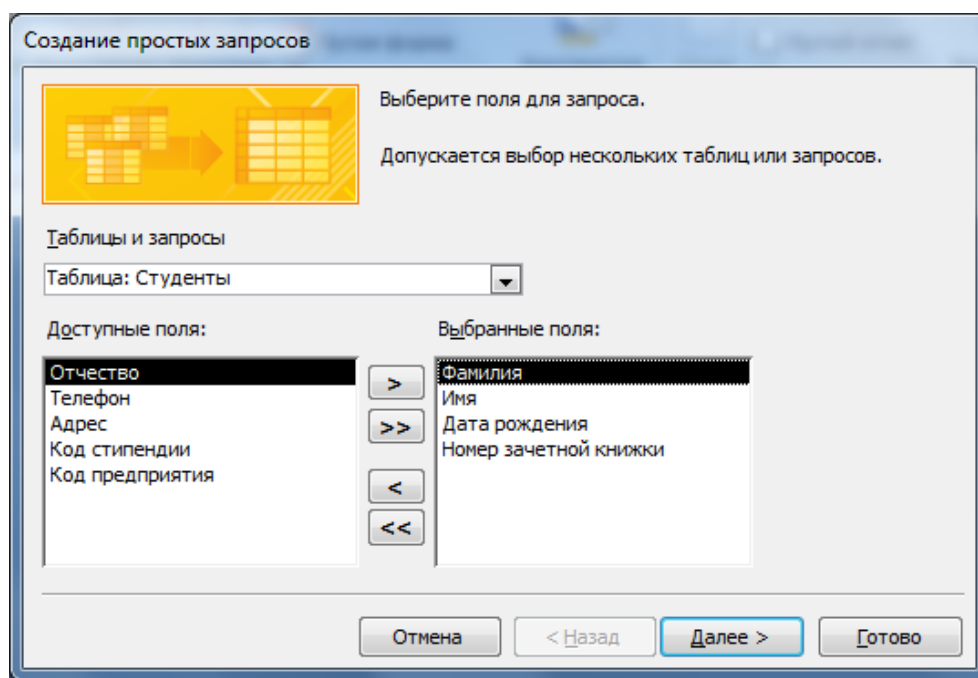


Рис. 2.4. Окно Мастера запросов

В секции *Доступные поля* выделите поля: *Фамилия, Имя, Дата рождения, Номер зачетной книжки*. Нажмите *Далее*. Появится окно, в котором в поле *Задайте имя запроса:* введите *Студенты*, установите переключатель *Открыть запрос для просмотра данных* и нажмите *Готово*.

Отобразится таблица с заданным именем, содержащая выбранные данные, а в области переходов появится название только что созданного запроса (рис. 5).

Все объекты Access		Студенты Запрос			
Таблицы		Фамилия	Имя	Дата рожде	Номер заче
Дисциплины		Авдеев	Игорь	05.08.1998	ПМ2-151.01
Практика		Варава	Анна	05.03.1998	ПМ2-151.02
Стипендия		Гаврилов	Семен	02.10.1999	ПМ2-151.03
Студенты		Светлова	Анна	02.03.1998	ПМ2-151.04
Учебный план		Иванов	Петр	18.05.1999	ПМ2-151.05
Запросы		*			
Студенты Запрос					

Рис. 2.5. Результат выполнения запроса

### Упражнение 2.

Создайте запрос, выводящий список студентов, получающих повышенную стипендию.

1. На вкладке *Создание Главного меню* выберите *Конструктор запросов*. Добавьте таблицы *Студенты* и *Стипендия*. Получится схема данных (рис. 6)

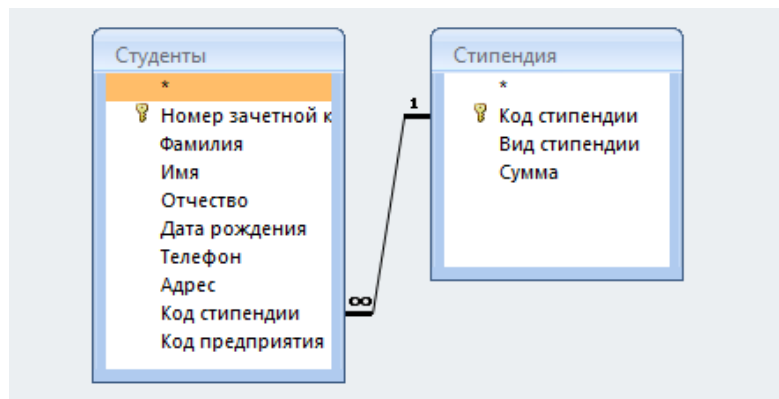


Рис. 2.6. Схема данных


В нижней части окна конструктора находится бланк запроса – специальная таблица, ячейки которой используются для определения структуры и свойств запроса. В бланке отображаются все столбцы базовых таблиц, включенные в результирующее множество запроса.

2. В бланке запроса выберите поля *Фамилия*, *Имя*, *Отчество* (таблица *Студенты*), *Вид стипендии* и *Сумма* (таблица *Стипендия*).

В столбце *Вид стипендии* в поле *Условие отбора* укажите «повышенная» (рис. 2.7).

Поле:	Фамилия	Имя	Отчество	Вид стипендии	Сумма
Имя таблицы:	Студенты	Студенты	Студенты	Стипендия	Стипендия
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:				"повышенная"	
или:					

Рис. 2.7. Конструктор запроса

3. Выполните запрос, нажав на кнопку *Выполнить* , и просмотрите результат. Сохраните запрос.

### Упражнение 3.

Создайте запрос, выводящий список дисциплин, у которых вид итогового контроля *Экзамен* и которые сдает студент Авдеев.

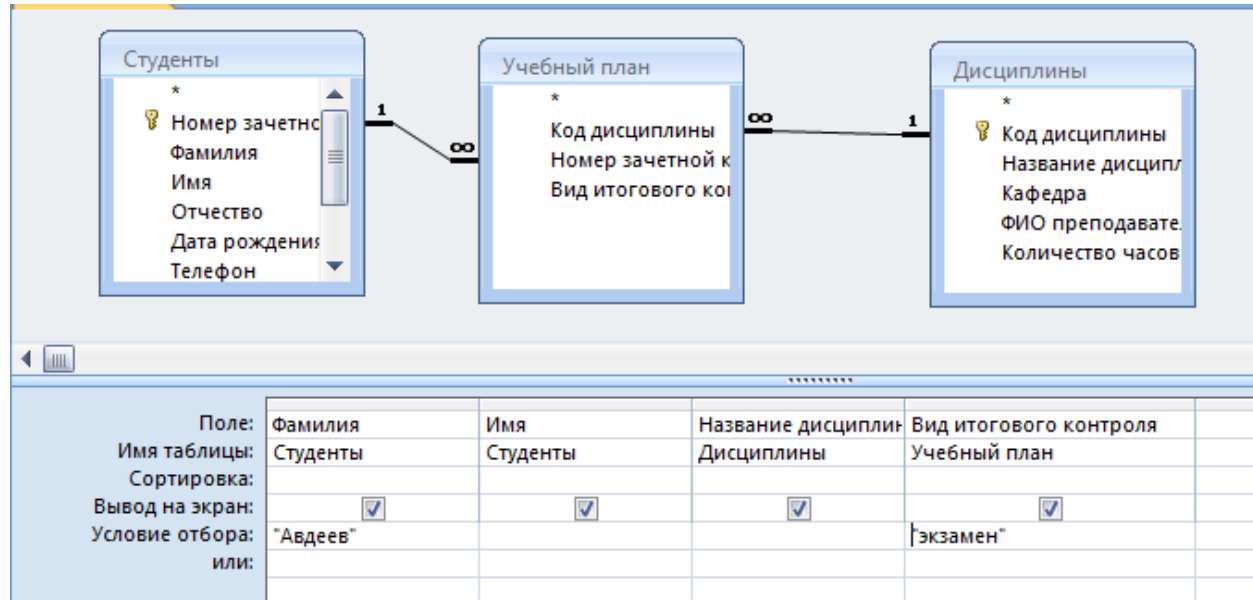


Рис. 2.8. Конструктор запроса

### Упражнение 4

Создайте запрос, отображающий информацию о студентах, год рождения которых 1998.

1. Создайте простой запрос, содержащий *Фамилию*, *Имя*, *Отчество* и *Дату рождения* студентов. Откройте его в режиме конструктора запросов.
2. В поле *Дата рождения* строку *Условие отбора* введите  $\geq \#01.01.1998\# \text{ And } \leq \#31.12.1998\#$
3. Запустите запрос, нажав кнопку *Выполнить*. Просмотрите результат.
4. Сохраните запрос.

### Упражнение 5

Создайте запрос, отображающий список дисциплин, которые преподает конкретный преподаватель.

1. Создайте простой запрос, который будет выводить поля *Название дисциплины*, *Кафедра*, *ФИО преподавателя*, *Количество часов*.
2. В окне конструктора запроса в строку *Условие отбора*: для поля *ФИО преподавателя* [Введите ФИО преподавателя] и нажмите кнопку *Выполнить*. Появится диалоговое окно ввода значения параметра (рис. 7), в

которое введите *Федоров К.А.* и нажмите *ОК*. Просмотрите результат, сохраните и закройте запрос.

### Упражнение 6

Создайте запрос, который по названию предприятия прохождения практики выводит Фамилию, Имя, Отчество студентов и номер зачетной книжки. *Название предприятия* – параметр запроса.

### Упражнение 7

Создайте простую форму для таблицы *Студенты*. Для этого:

1. В области переходов выберите таблицу *Студенты*.
2. На вкладке *Создание* в группе *Формы* выберите элемент *Форма*.

Приложение *Access* создаст форму для выбранной таблицы и отобразит ее в режиме макета (рис. 2.9).

Номер зачетной книжки:	PM2-151.01
Фамилия:	Авдеев
Имя:	Игорь
Отчество:	Игоревич
Дата рождения:	05.08.1998
Телефон:	(950)325-62-56
Адрес:	ул. Докучаева д.6 кв. 85
Код стипендии:	1
Код предприятия:	1

Код дисциплины	Вид итогов						
1	Экзамен						
2	Экзамен						
*							

Рис. 2.9. Форма *Студенты*

### Упражнение 8

На основе таблицы *Студенты* создайте форму таким образом, чтобы была возможность работать с данными как через форму, так и через таблицу. Для этого:

1. В области переходов выберите таблицу *Студенты*.

2. На вкладке *Создание* в группе *Формы* нажмите на кнопку *Разделенная форма*. Приложение *Access* создаст форму для выбранной таблицы и отобразит ее в режиме макета.

3. Откройте появившуюся форму в режиме *формы* и сохраните ее под именем *Студенты*.

4. Добавьте новые записи (из табл. 1) – одну через форму, другую в таблицу. Убедитесь, что записи появляются в обоих представлениях.

5. Закройте форму.

Таблица 1

Данные таблицы *Студенты*

Номер зачетной книжки	Фамилия	Имя	Отчество	Дата рождения	Телефон	Код стипендии	Код предприятия
					Адрес		
ПМ2-151.25	Иванов	Иван	Иванович	15.08.1999	(903)444-62-56 ул. Докучаева д.25 кв. 5	02	02
ПМ2-151.26	Петров	Петр	Петрович	05.03.1998	(920)131-13-10 ул. Пушкина д. 31 кв. 20	01	02


### Упражнение 9

На основе таблицы *Дисциплины* создайте простой отчет.

1. В области переходов выберите таблицу *Дисциплины*, которая используется в качестве источника данных для нового отчета.

2. На вкладке *Создание* в группе *Отчеты* нажмите на кнопку *Отчет*. Отобразится созданный отчет в режиме макета (рис. 2.10).



		<b>Дисциплины</b>			9 декабря 2015 г. 15:53:37
Код дисциплины	Название дисциплины	Кафедра	ФИО преподавателя	Количество часов	
1	Математика	Математики	Еремин П.П.	72	
2	ИТ в менеджменте	Информатики	Петрова М.И.	72	
3	Менеджмент	Экономики	Федоров К.А.	144	
4	Экономическая теория	Экономики	Исакова А.В.	72	
5	Английский	Иностранного языка	Рагозина Л.И.	54	
6	Психология	Гуманитарных наук	Коротких А.И.	36	
7	История	Гуманитарных наук	Власов А.Н.	54	
7					

Страница 1 из 1

Рис. 2.10. Простой отчет

3. Сохраните отчет под именем *Дисциплины*.

### **Упражнение 10**

Создайте наклейки для почтовых отправлений уведомлений о неуспеваемости студентов.

1. В области переходов выделите таблицу *Студенты* и на вкладке *Создание* в группе *Отчеты* нажмите *Наклейки*.

2. В появившемся окне *Мастера наклеек* (рис. 2.11) установите:

Система единиц – метрическая

Тип наклеек – на листах

Фильтр по изготовителю – Agira

Шаблон – Agira 119012

и нажмите *Далее*.

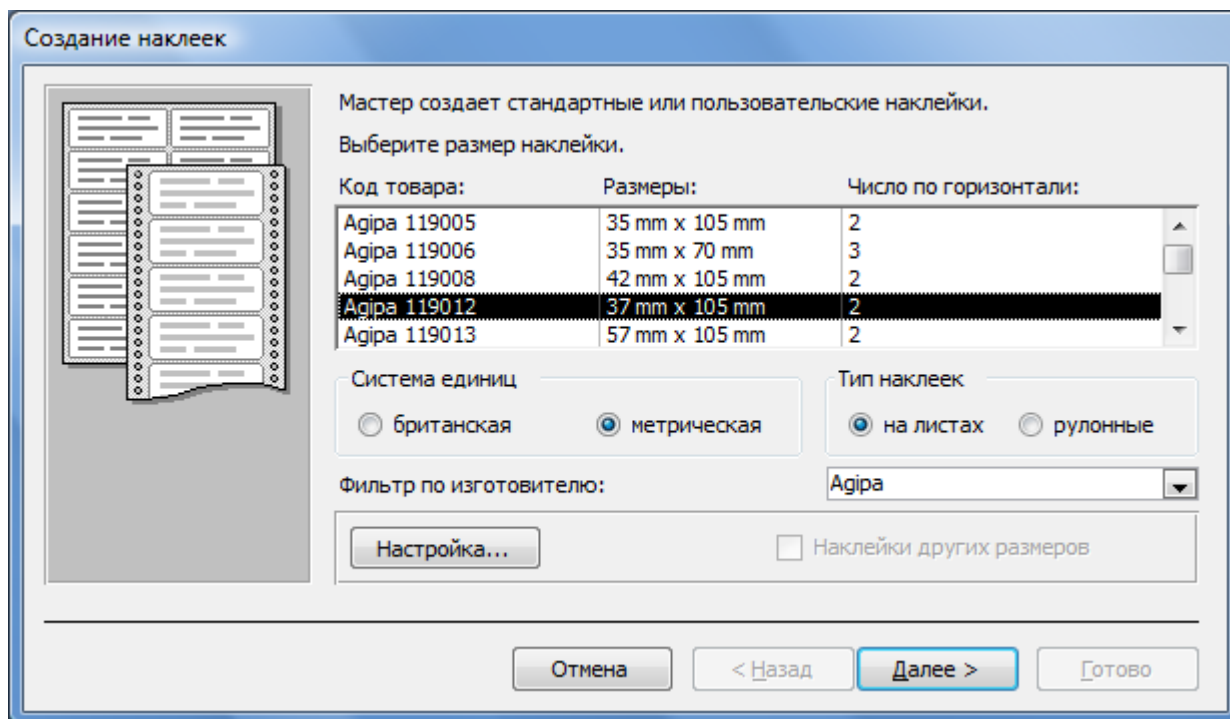


Рис. 2.11. Выбор размера наклейки

3. В следующем окне *Мастера* установите:

Шрифт – Arial

Размер – 12

Насыщенность – Средний

Начертание – Курсив

и нажмите *Далее*.

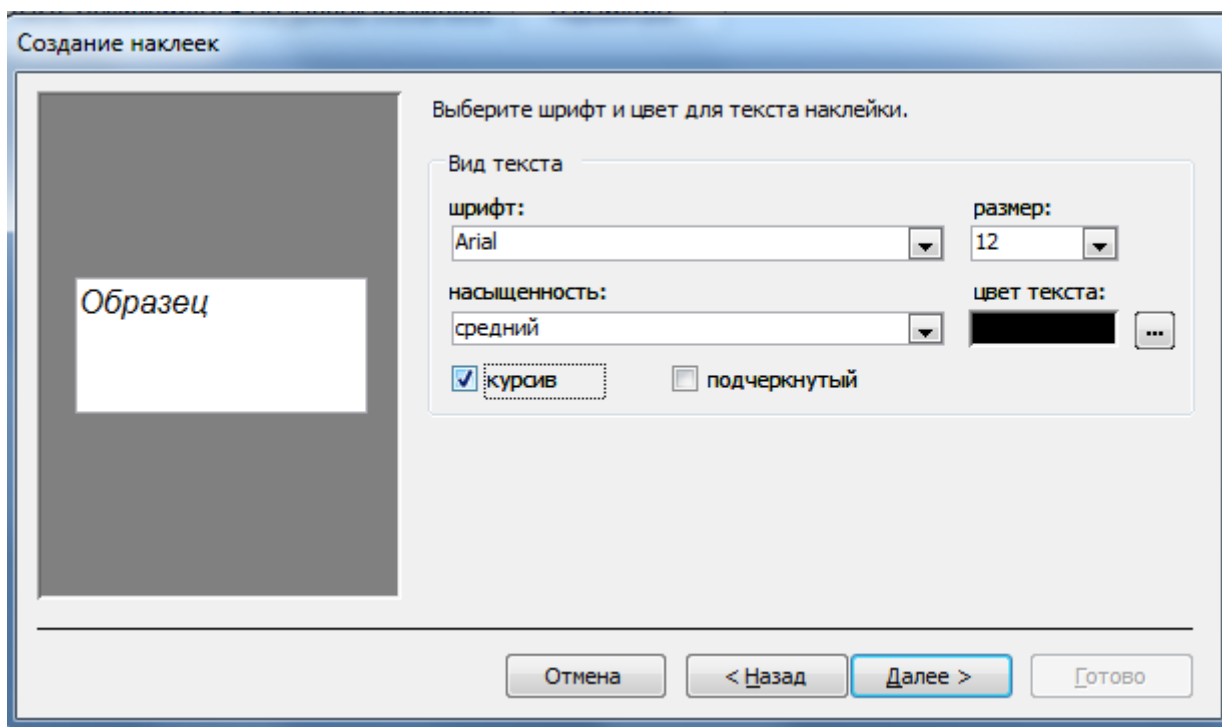


Рис. 2.12. Формат шрифта наклейки

4. В следующем окне (рис. 2.13) следует указать поля, которые требуется разместить на наклейке.

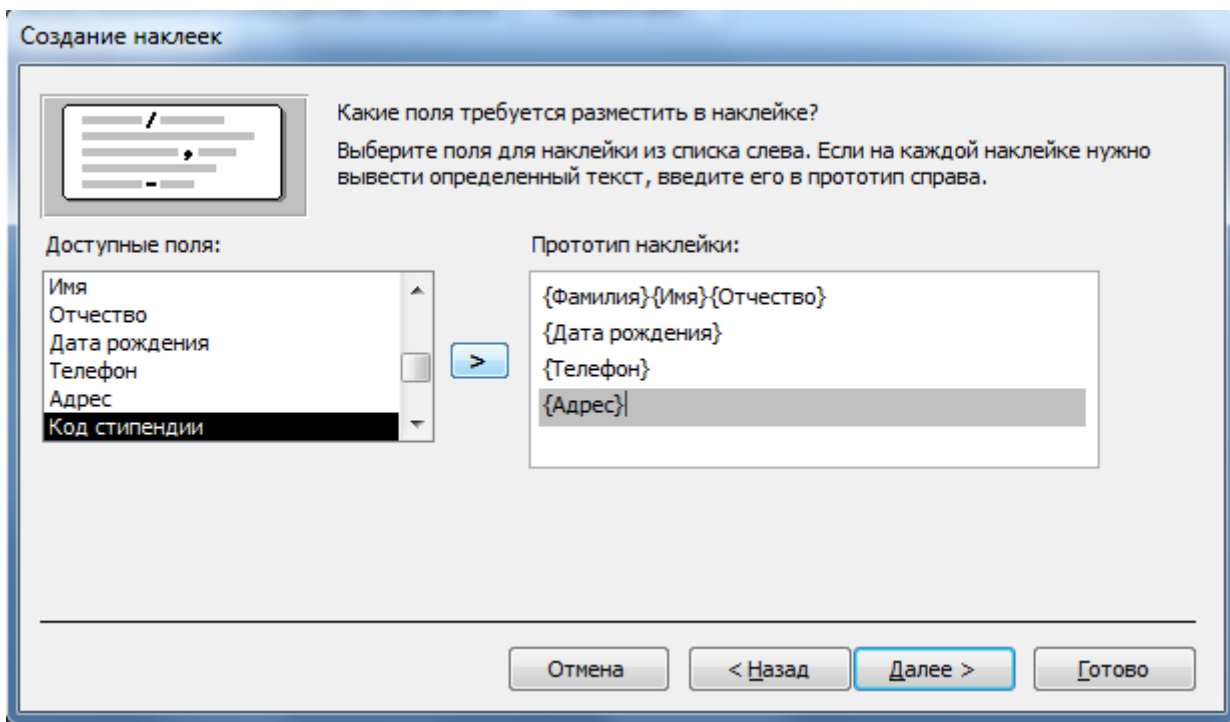


Рис. 2.13. Окно размещения полей на наклейке

5. В следующем окне *Мастера наклеек* задайте сортировку наклеек по полю *Фамилия* и нажмите *Далее*.

6. В последнем окне *Мастера* введите имя отчету *Почтовые наклейки* и нажмите *Готово*. Если заданные наклейки не помещаются на странице отчета, то перед просмотром наклеек отобразится диалоговое окно предупреждения о возможных проблемах вывода данных (рис. 14). Нажмите *ОК*, чтобы закрыть окно предупреждения и начать просмотр наклеек.

Откроется окно предварительного просмотра отчета, содержащего наклейки (рис. 2.15).

7. Закройте отчет.

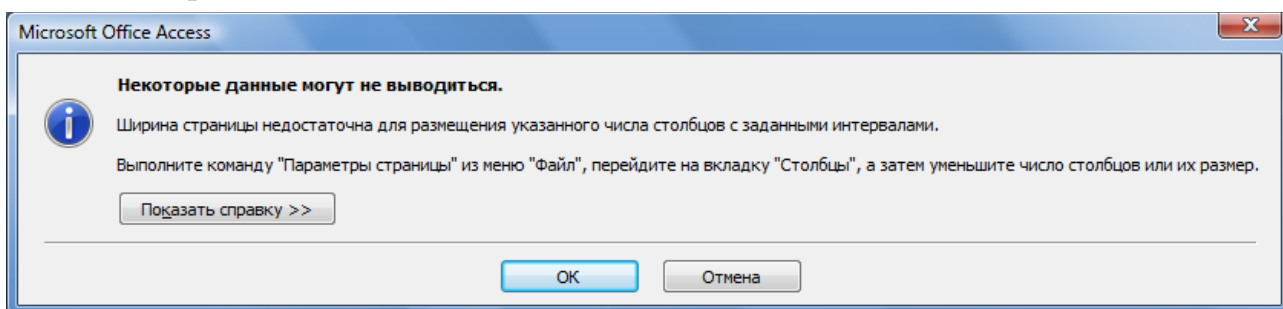


Рис. 2.14. Предупреждение об ошибках вывода

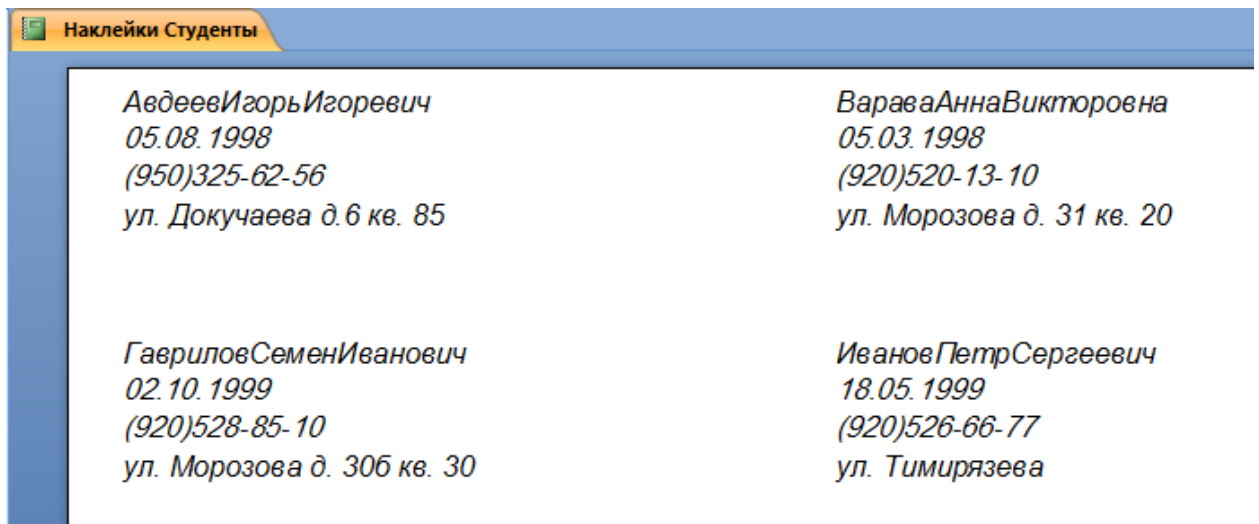


Рис. 2.15. Готовые наклейки

#### 4. Контрольные вопросы:

1. Как выполнить сортировку записей в таблице?
2. Что такое фильтр и как его создать?
3. Что такое запрос? Для чего он предназначен?
4. Какие способы создания запросов существуют?
5. Что такое форма?
6. Какие способы создания формы существуют?
7. Чем отличается разделенная форма от обычной?
8. Что такое отчет?
9. Какие существуют способы создания отчета в Access?

### ЛАБОРАТОРНАЯ РАБОТА № 3

#### Модели данных. Проектирование баз данных.

**1. Цель работы:** получить теоретические и практические навыки построения моделей данных при проектировании реляционных баз данных.

#### **2. Теоретический материал для домашнего изучения.**

**Предметная область** - часть реального мира, подлежащая изучению с целью организации управления и, в конечном счете, автоматизации. Предметная область представляется множеством *фрагментов*, например, предприятие - цехами, дирекцией, бухгалтерией и т.д. Каждый фрагмент предметной области характеризуется множеством *объектов* и *процессов*, использующих другие

объекты, а также множеством *пользователей*, характеризующихся различными взглядами на предметную область.

*Проектирование базы данных* – одна из наиболее сложных и ответственных задач, в результате решения которой должны быть определены содержание БД, эффективный для всех её будущих пользователей, способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

- корректность схемы БД;
- обеспечение ограничений;
- эффективность функционирования;
- защита данных (от аппаратных и программных сбоев и несанкционированного доступа);
- простота и удобство эксплуатации;
- гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Процесс проектирования БД включает в себя следующие этапы:

**1. Концептуальное проектирование** – сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия: обследование предметной области, изучение ее информационной структуры; выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами; моделирование и интеграция всех представлений. По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели "сущность-связь".

**2. Логическое проектирование** – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

**3. Физическое проектирование** – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

**Примечание:** следует различать проектирование систем складирования данных и проектирование так называемых OLTP-систем, ориентированных на оперативную обработку транзакций.

Сильно нормализованные модели данных хорошо подходят для так называемых **OLTP-приложений** (*On-Line Transaction Processing (OLTP)-оперативная обработка транзакций*). Типичными примерами OLTP-приложений являются системы складского учета, системы заказов билетов, банковские системы, выполняющие операции по переводу денег, и т.п.

Основная функция подобных систем заключается в выполнении большого количества коротких транзакций. Сами транзакции выглядят относительно просто, например, "снять сумму денег со счета А, добавить эту сумму на счет В".

Проблема заключается в том, что, во-первых, транзакций очень много, во-вторых, выполняются они одновременно (к системе может быть подключено несколько тысяч одновременно работающих пользователей), в-третьих, при возникновении ошибки, транзакция должна целиком откатиться и вернуть систему к состоянию, которое было до начала транзакции (не должно быть ситуации, когда деньги сняты со счета А, но не поступили на счет В).

Практически все запросы к базе данных в OLTP-приложениях состоят из команд вставки, обновления, удаления. Таким образом, критическим для OLTP-приложений является скорость и надежность выполнения коротких операций обновления данных. Чем выше уровень нормализации данных в OLTP-приложении, тем оно, как правило, быстрее и надежнее.

Другим типом приложений являются так называемые **OLAP-приложения** (*On-Line Analytical Processing (OLAP) - оперативная аналитическая обработка данных*). Это обобщенный термин, характеризующий принципы построения *систем поддержки принятия решений (Decision Support System - DSS)*, *хранилищ данных (Data Warehouse)*, *систем интеллектуального анализа данных (Data Mining)*. Такие системы предназначены для нахождения зависимостей между данными (например, можно попытаться определить, как связан объем продаж товаров с характеристиками потенциальных покупателей), для проведения анализа "что если...".

OLAP-приложения оперируют с большими массивами данных, уже накопленными в OLTP-приложениях, взятыми их электронных таблиц или из других источников данных. Такие системы характеризуются следующими признаками:

Добавление в систему новых данных происходит относительно редко крупными блоками (например, раз в квартал загружаются данные по итогам квартальных продаж из OLTP-приложения).

Данные, добавленные в систему, обычно никогда не удаляются.

Перед загрузкой данные проходят различные процедуры "очистки", связанные с тем, что в одну систему могут поступать данные из многих источников, имеющих различные форматы представления для одних и тех же понятий, данные могут быть некорректны, ошибочны.

Запросы к системе являются нерегламентированными и, как правило, достаточно сложными.

Скорость выполнения запросов важна, но не критична.

Данные OLAP-приложений обычно представлены в виде одного или нескольких гиперкубов, измерения которого представляют собой справочные данные, а в ячейках самого гиперкуба хранятся собственно данные. Например, можно построить гиперкуб, измерениями которого являются: время (в кварталах, годах), тип товара и отделения компании, а в ячейках хранятся объемы продаж. Такой гиперкуб будет содержать данных о продажах различных типов товаров по кварталам и подразделениям. Основываясь на этих данных, можно отвечать на вопросы вроде "у какого подразделения самые лучшие объемы продаж в текущем году?", или "каковы тенденции продаж отделений Юго-Западного региона в текущем году по сравнению с предыдущим годом?"

Возвращаясь к проблеме нормализации данных, можно сказать, что в системах OLAP, использующих реляционную модель данных (ROLAP), данные целесообразно хранить в виде слабо нормализованных отношений, содержащих заранее вычисленные основные итоговые данные. Большая избыточность и связанные с ней проблемы тут не страшны, т.к. обновление происходит только в момент загрузки новой порции данных. При этом происходит как добавление новых данных, так и пересчет итогов.

Выделяют следующие модели данных:

- концептуальные (инфологические),
- даталогические,
- физические.

Процесс проектирования БД начинается с создания инфологической модели. **Концептуальная (инфологическая) модель данных** – обобщенное

неформальное описание создаваемой базы данных, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающим над проектированием базы данных.

или по-другому

**Инфологическая модель данных** - обобщенное, непривязанное к каким-либо СУБД описание предметной области.

Существует множество подходов к построению таких моделей: *графовые модели, семантические сети, модель «сущность-связь» (ER-диаграмма)* и др.

**ER – модель** (модель «сущность-связь» (ER-диаграмма)) представляет собой обобщение реляционной модели данных путем разделения отношений, описывающих предметную область на две группы – сущностей и связей.

*Сущность* – некоторая абстракция реально-существующего объекта, процесса или явления, о котором необходимо хранить информацию в БД. Тип сущности определяет набор однородных объектов, а экземпляр – конкретный объект. Для идентификации конкретных экземпляров сущности используются специальные *атрибуты* – поименованные характеристики сущности, которые принимают значения из некоторого множества значений.

*Связи* – это средства представления отношений между сущностями, имеющими место в предметной области. Связи бывают бинарные, тернарные (между тремя сущностями), n-арные.

ER – модель является удобным средством описания предметной области перед тем, как перейти к ее представлению в реляционной модели данных. При проектировании БД на основе ER – моделей используют ER – диаграммы, на которых сущности представляют в виде четырехугольников, их атрибуты в виде овалов, связи между сущностями обозначают стрелками, а имя связи – в виде ромба.



Рис.3.1. Графическое изображение элементов ER – диаграммы

Инфологическая модель должна быть отображена в **дatalogическую модель**, «понятную» СУБД. Для ее реализации используют следующие модели: *иерархическую, сетевую, реляционную*.



**Иерархическая даталогическая модель** представляет собой совокупность связанных элементов, образующих иерархическую структуру. К основным понятиям иерархии относятся уровень, узел и связь. Узлом называется совокупность атрибутов данных, описывающих некоторый объект. Каждый узел связан с одним узлом более высокого уровня и с любым количеством узлов нижнего уровня. Исключением является узел самого высокого уровня, который не связан ни с одним узлом более высокого уровня.

В основе **сетевой модели данных** лежат те же понятия, что и в основе иерархической модели – узел, уровень и связь. Однако существенным различием является то, что в иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков. Сетевой подход к организации данных является расширением иерархического.

Сегодня наиболее распространена **реляционная модель**. В ее основе лежит идея о том, что любой набор данных можно представить в виде двумерной таблицы. Простейшая реляционная БД может состоять из единственной таблицы, в которой будут храниться все необходимые данные. На практике реляционная БД состоит из нескольких таблиц, связанных между собой по определенным критериям.

В основе реляционной модели данных лежит понятие отношения, представляющего собой подмножество декартова произведения доменов. Элементы отношения называют кортежами, элементы кортежа – атрибутами (полями). Длина кортежа (количество атрибутов) определяет **арность отношения**, количество кортежей – **мощность отношения**.

Обращение к таблице происходит по имени. Каждый атрибут также имеет имя, принадлежит к определенному типу данных и характеризуется размером памяти, выделяемой под его хранение. Перечень атрибутов отношения с их типами и размерами называется схемой отношения. Отношения, построенные по одинаковой схеме, называют односхемными, по разным схемам – разносхемными.

На атрибут (группу атрибутов) могут накладываться ограничения целостности, т.е. правила, которым должно соответствовать значение атрибута (или соотношение значений атрибутов).

Атрибут (группа атрибутов), значения которого являются уникальными в рамках отношения, идентифицируют кортеж и называются

потенциальными ключами. Если ключ состоит из нескольких атрибутов, он называется составным. Ключей может быть несколько, основным является **первичный ключ** (*primary key*). Таблица может иметь только один первичный ключ.

Отношение обладает двумя важными свойствами:

1. в отношении не должно быть одинаковых кортежей.
2. Порядок кортежей в отношении не существен.

Связи между отношениями реализуются с помощью внешнего ключа.

**Внешний ключ** (*foreign key*) – это атрибут подчиненного (дочернего) отношения, который является копией первичного (*primary key*) или уникального (*unique*) ключа родительского отношения.

В стандартах SQL используются другие термины: отношение принято называть **таблицей**, кортеж – **строкой**, а атрибут – **столбцами** таблицы.

На основе даталогической модели строится **физическая модель**.

Физическая организация данных оказывает основное влияние на эксплуатационные характеристики БД. Разработчики СУБД пытаются создать наиболее производительные физические модели данных, предлагая пользователям тот или иной инструментарий для поднастройки модели под конкретную БД.

Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

### ***Общие сведения о базах данных и системах управления базами данных***

**База данных** (БД) – это совокупность данных, организованная по определенным правилам, которые предусматривают общие принципы описания, хранения и манипулирования данными.

База данных включает данные, отражающие логическую модель взаимосвязанных информационных объектов, представляющих конкретную предметную область. Например, база данных может быть сформирована из списка сотрудников, заказчиков, расчетных ведомостей, сведений дорожных происшествий и т.д.

*Системы управления базами данных (СУБД)* являются инструментальными программными средствами, предназначенными для создания и ведения баз данных на внешних носителях, а также организации доступа к данным и их обработки.

База данных организуется в соответствии с моделью и структурами данных, которые поддерживаются системами управления базами данных.

Наиболее популярными являются *реляционные СУБД*, в которых все данные сгруппированы во взаимосвязанные таблицы. Таблицы используются для представления объектов и связей между ними. Каждая таблица имеет уникальное имя и состоит из строк и столбцов, где строки называются *записями*, а столбцы – *полями*. Каждая строка в таблице представляет некоторый объект реального мира или соотношения между объектами. Например, каждая строка (запись) таблицы *Книги* содержит сведения о какой-либо книге. Свойства объекта, его характеристики определяются значениями полей. Например, для книг полями будут автор, название, год издания и другие. Каждому имени поля ставится в соответствие множество допустимых значений (или определяется некоторый базовый тип данных, к которому относятся значения данного поля).

Любая таблица реляционной базы данных обладает следующими свойствами:

- каждое поле записи имеет единственное значение, а не состоит из группы значений;
- отсутствуют одинаковые записи;
- порядок следования полей и записей не имеет значения.

В таблице могут быть одно или несколько полей, которые однозначно идентифицируют запись таблицы, то есть определяют значения других полей, и называются *ключевыми полями* (или потенциальными ключами).

### **3. Лабораторные задания.**

#### ***Упражнение 1.***

Изучить и проанализировать модель «сущность-связь» для следующей предметной области – работа отдела сбыта издательства, занимающегося поставкой книг в магазины.

В данной предметной области можно выделить объекты (сущности) СОТРУДНИКИ, МАГАЗИНЫ, КНИГИ, ТРАНСПОРТНЫЕ КОМПАНИИ и ЗАКАЗЫ, связи и атрибуты (рис. 1):

– ДЕЛАЮТ – между объектами МАГАЗИНЫ и ЗАКАЗЫ. Магазин делает несколько заказов в издательство на поставку книг, а один заказ предназначен только для одного магазина, поэтому связь ДЕЛАЮТ относится к типу *один ко многим*;

– ОФОРМЛЯЮТ – между объектами СОТРУДНИКИ и ЗАКАЗЫ. Сотрудник отдела сбыта издательства оформляет несколько заказов, а один заказ может быть оформлен только одним сотрудником, поэтому связь ОФОРМЛЯЮТ относится к типу *один ко многим*;

– ВКЛЮЧАЮТ – между объектами ЗАКАЗЫ и КНИГИ. Заказ может включать несколько книг, а одна книга может входить в несколько различных заказов, поэтому связь ВКЛЮЧАЮТ относится к типу *многое ко многим*;

– ДОСТАВЛЯЮТ – между объектами ТРАНСПОРТНЫЕ КОМПАНИИ и ЗАКАЗЫ. Одна транспортная компания выполняет доставку нескольких заказов, а один заказ доставляет только одна транспортная компания, поэтому связь ДОСТАВЛЯЮТ относится к типу *один ко многим*.

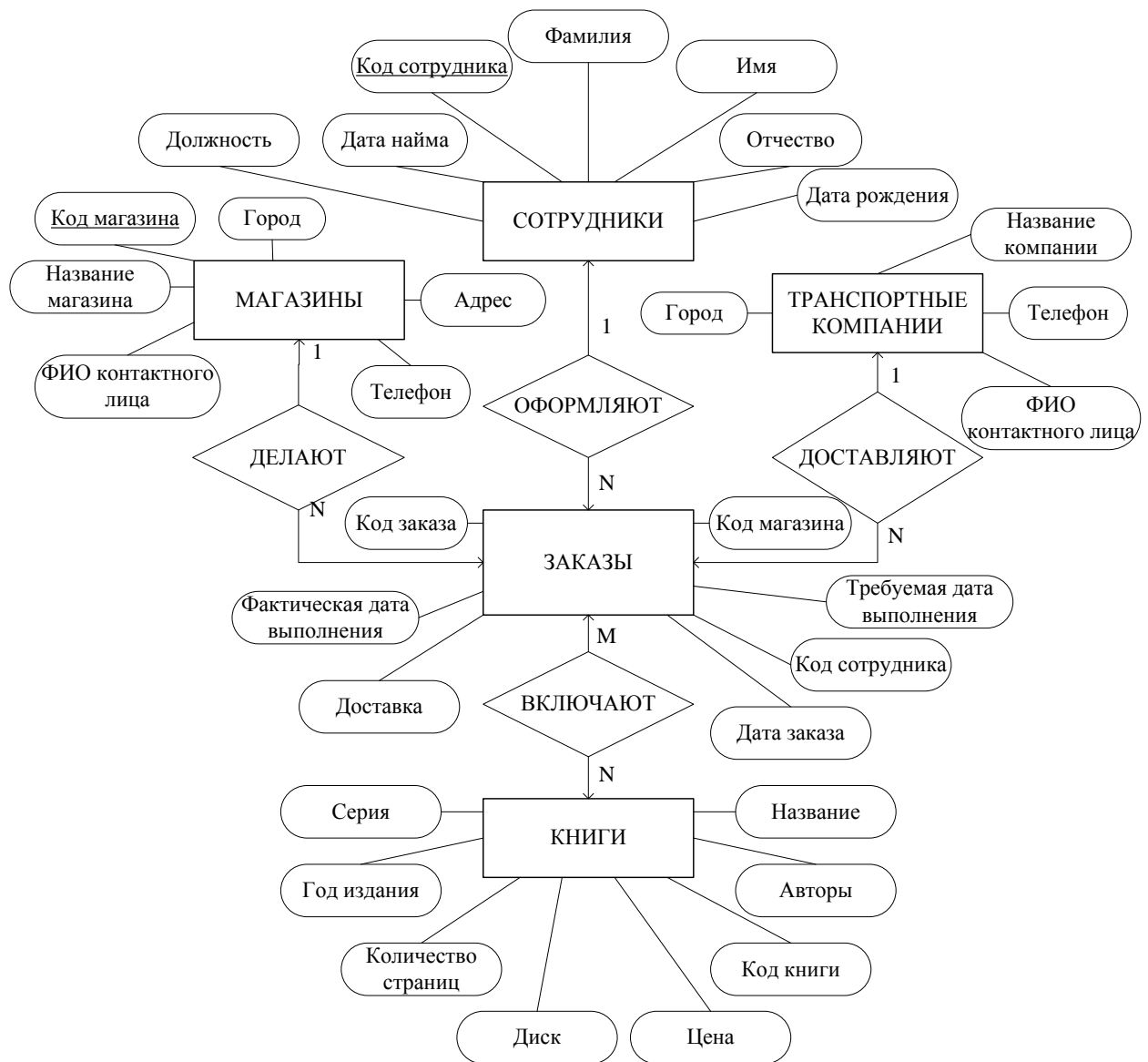


Рис. 3.1. *ER-диаграмма* базы данных «Поставки книг в магазины»

Для каждого объекта выделены свойства (атрибуты):

- СОТРУДНИКИ (Код сотрудника, Фамилия, Имя, Отчество, Должность, Телефон, Дата рождения, Дата найма);
- МАГАЗИНЫ (Код магазина, Название магазина, ФИО контактного лица, Город, Адрес, Телефон);
- КНИГИ (Код книги, Авторы, Название, Серия, Год издания, Количество страниц, Цена, Диск);
- ТРАНСПОРТНЫЕ КОМПАНИИ (Название компании, ФИО контактного лица, Город, Телефон);
- ЗАКАЗЫ (Код заказа, Код магазина, Код сотрудника, Дата заказа, Требуемая дата выполнения, Фактическая дата выполнения, Доставка).

### Упражнение 2.

Изучить и проанализировать реляционную структуру БД и реализацию связи многие ко многим в этой структуре данных.

Для исключения связи типа *многое ко многим* связь ВКЛЮЧАЮТ заменим объектом ОПИСАНИЕ ЗАКАЗОВ (Код заказа, Код книги, Количество, Скидка), который обеспечит взаимосвязь записей объектов КНИГИ и ЗАКАЗЫ. Таким образом, схема реляционной базы данных «Поставки книг в магазины» будет как на рис. 3.2.

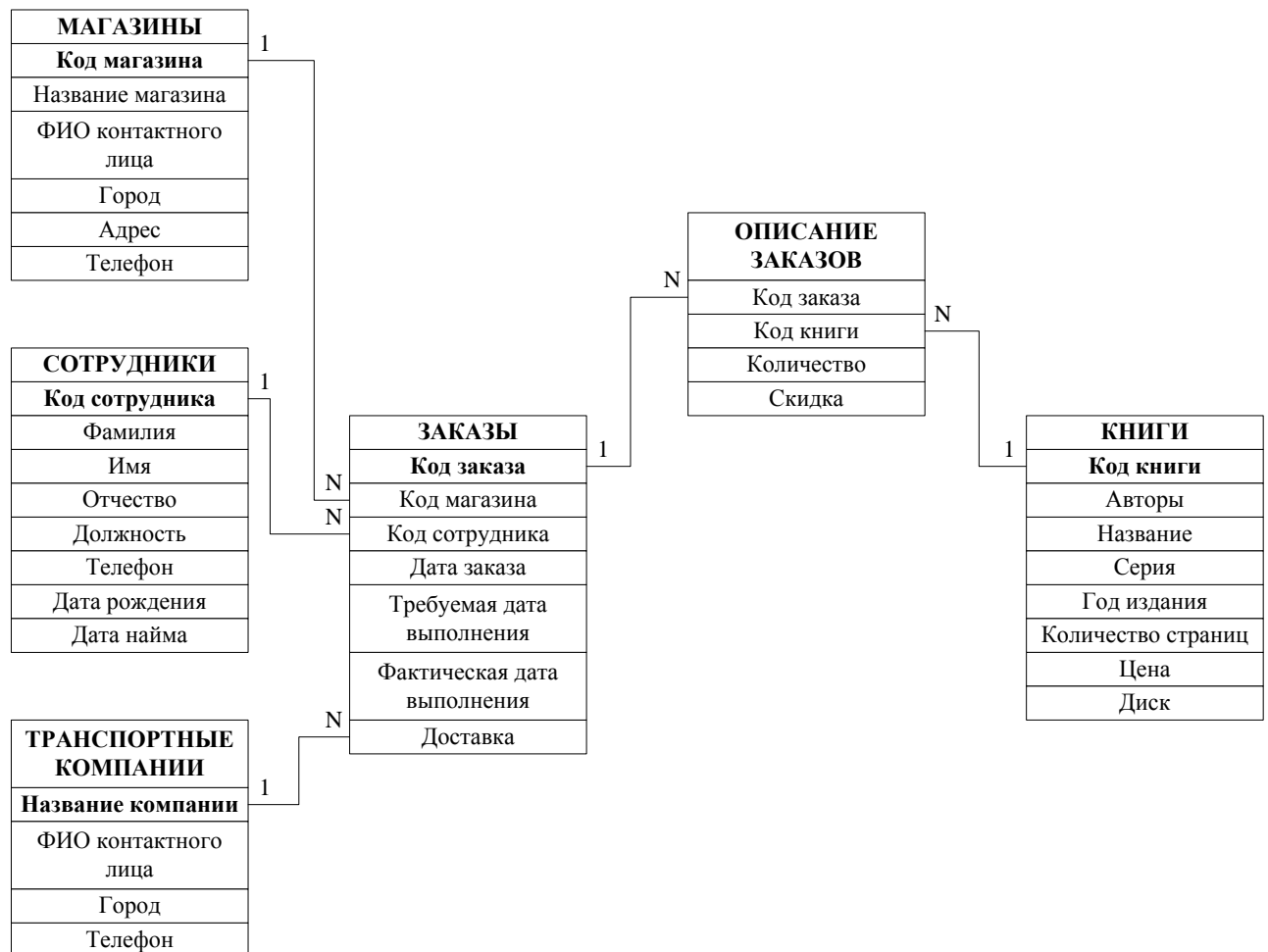


Рис. 3.2. Реляционная модель базы данных «Поставки книг в магазины»

Спроектировав структуру базы данных, можно использовать программное обеспечение СУБД для создания таблиц.

### Упражнение 3.

Разработать согласно индивидуальному заданию *ER-диаграмму* и реляционную модель базы данных.

#### 4. Контрольные вопросы

1. Дайте определение инфологической модели данных.
2. Подходы к построению инфологической модели данных.
3. Даталогическая модель данных.
4. Дайте определения терминам сущность, атрибут, связь и т.д., используемым при построении ER-диаграммы.
5. Дайте определения базе данных и системе управления базами данных.
6. Расскажите основные принципы построения реляционной базы данных.
7. Перечислите свойства таблицы в реляционной базе данных.
8. Какие существуют связи?

### ЛАБОРАТОРНАЯ РАБОТА № 4

#### Операции реляционной алгебры

**1. Цель работы:** получить теоретические и практические навыки применения операций реляционной алгебры.

#### 2. Теоретический материал для домашнего изучения.

Важной составляющей модели данных является описание набора операций над данными, которые поддерживает эта модель. Особенностью реляционной модели, коренным образом отличающей ее от предшествующих (дореляционных) моделей данных, является строгое математическое описание этой модели и лежащий в ее основе математический аппарат.

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных. Наиболее значительной из них была статья сотрудника фирмы IBM д-ра Эдварда Кодда (*Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970*), где впервые был применен термин «реляционная модель данных».

Будучи математиком по образованию, Э. Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение – *relation*.

Наименьшая единица данных реляционной модели – это отдельное атомарное (неразложимое) для данной модели значение данных.

Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой – как три различных значения.

**Доменом** называется множество атомарных значений одного и того же типа.

Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос «Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву»). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета?

Отношение на доменах  $D1, D2, \dots, Dn$  состоит из заголовка и тела. На рис. 4.1 приведен пример отношения для расписания движения самолетов, где  $A_i$  - атрибуты,  $V_i$  - значения атрибутов.

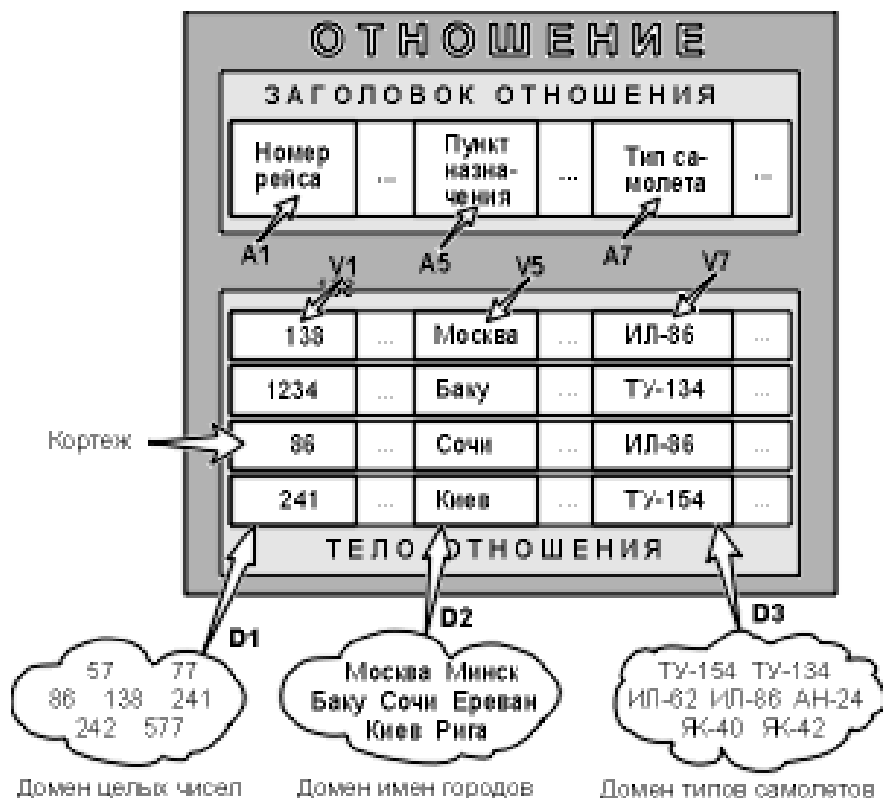


Рис. 4.1. Отношение с математической точки зрения



**Заголовок отношения** состоит из такого фиксированного множества атрибутов  $A_1, A_2, \dots, A_n$ , что существует взаимно однозначное соответствие между этими атрибутами  $A_i$  и определяющими их доменами  $D_i$  ( $i=1,2,\dots,n$ ).

**Тело отношения** состоит из меняющегося во времени множества **кортежей**, где каждый **кортеж** состоит в свою очередь из множества пар атрибут-значение  $(A_i:V_i)$ , ( $i=1,2,\dots,n$ ), по одной такой паре для каждого атрибута  $A_i$  в заголовке.

Для любой заданной пары атрибут-значение  $(A_i:V_i)$   $V_i$  является значением из единственного домена  $D_i$ , который связан с атрибутом  $A_i$ .

**Степень отношения** – это число его атрибутов.

Отношение степени один называют унарным, степени два – бинарным, степени три – тернарным, ..., а степени  $n$  –  $n$ -арным. Степень отношения «Рейс» (рис. 3.2) равна 8.

**Кардинальное число или мощность отношения** – это число его кортежей. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение – это множество, а множества по определению не содержат совпадающих элементов, то **никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно заданный момент времени.**

Пусть  $R$  – отношение с атрибутами  $A_1, A_2, \dots, A_n$ . Говорят, что множество атрибутов  $K=(A_i, A_j, \dots, A_k)$  отношения  $R$  является **возможным ключом**  $R$  тогда и только тогда, когда удовлетворяются два независимых от времени условия:

1. **Уникальность:** в произвольный заданный момент времени никакие два различных кортежа  $R$  не имеют одного и того же значения для  $A_i, A_j, \dots, A_k$ .
2. **Минимальность:** ни один из атрибутов  $A_i, A_j, \dots, A_k$  не может быть исключен из  $K$  без нарушения уникальности.

Каждое отношение обладает хотя бы одним возможным **ключом**, поскольку по меньшей мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный произвольным образом) принимается за его **первичный ключ**. Остальные возможные ключи, если они есть, называются **альтернативными ключами**.

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки

соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных.

**Замечание.** Также на практике широко используются неформальные эквиваленты этих понятий: *Отношение* – *Таблица*, *Кортеж* – *Строка таблицы* или *Запись*, *Атрибут* – *Столбец Таблицы* или *Поле*.

При этом принимается, что «запись» означает «экземпляр записи», а «поле» означает «имя и тип поля».

**Операции реляционной алгебры** Кодда делятся на 2 группы:

**Базовые теоретико-множественные** включают классические операции теории множеств: объединения, разности, пересечения и произведения.

**Специальные реляционные** включают операции проекции, селекции, деления и соединения.

Операции реляционной алгебры могут быть унарными – выполненными над одним отношением (к примеру, проекция), и бинарными – выполненными над двумя отношениями (к примеру, разность). Отношения, над которыми выполняется бинарная операция, должны быть совместимыми по структуре. Под совместимостью структур отношений понимается совместимость типов соответствующих доменов и имен атрибутов.

**Операции реляционной алгебры:**

**Объединение** (union).

Объединение – бинарная операция над односхемными отношениями. Объединением односхемных отношений  $R$  и  $S$  называется отношение  $T = R \cup S$ , которое включает в себя все кортежи обоих отношений без повторов.

Имена полей односхемных отношений могут быть разными, достаточно, чтобы совпадало количество полей, и типы данных соответствующих полей.

**Разность** (except).

Разность – бинарная операция над односхемными отношениями. Разностью отношений  $R$  и  $S$  называется множество кортежей  $R$ , не входящих в  $S$ .

**Пример.** Пусть имеются отношение  $R(A,B,C)$  и отношение  $S(A,B,C)$ , тогда:

<u>Отношение R</u>			<u>Отношение S</u>			<u>Разность R-S</u>		
A	B	C	A	B	C	A	B	C
a	b	c	g	h	a	c	a	d
c	a	d	a	b	c	c	h	c
c	h	c	h	d	d			

### **Пересечение** (intersect).

Пересечение – бинарная операция над односхемными отношениями. Пересечение односхемных отношений  $R$  и  $S$  есть подмножество кортежей, принадлежащих обоим отношениям. Это можно выразить через разность:  $R \cap S = R - (R - S)$ .

### **Декартово произведение** (Cartesian product).

Декартово произведение – это бинарная операция над разносхемными отношениями, соответствующая определению декартова произведения для реляционной модели данных. Кортежи результирующего отношения состоят из всех атрибутов исходных отношений.

**Пример.** Пусть имеются отношение  $R(A,B)$  и отношение  $S(C,D,E)$ . То декартово произведение  $R \times S$  будет:

<u>Отношение R</u>		<u>Отношение S</u>			<u>Декартово произведение R×S</u>				
A	B	C	D	E	A	B	C	D	E
1	4	g	h	a	1	4	a	b	c
2	5	a	b	c	2	5	g	h	a
3	6				2	5	a	b	c
					3	6	g	h	a
					3	6	a	b	c

### **Селекция** (select).

Селекция – это унарная операция, результатом которой является подмножество кортежей исходного отношения, соответствующих условиям, которые накладываются на значения одного или нескольких атрибутов. Арность отношения в результате селекции не меняется.

**Пример.** Для отношения  $R(A,B,C)$  селекция  $\sigma_{C=d}(R)$  (при условии «значение атрибута C равно d») будет:

<u>Отношение R</u>			<u>Селекция <math>\sigma_{C=d}(R)</math></u>		
A	B	C	A	B	C
a	b	c	c	a	d
c	a	d	c	b	d
c	b	d			

### **Проекция** (project).

Проекция – это унарная операция (выполняемая над одним отношением), служащая для выбора подмножества атрибутов из отношения

R. Она уменьшает арность отношения и может уменьшить мощность отношения за счет исключения одинаковых кортежей.

**Пример.** Пусть имеется отношение R(A,B,C). Тогда  $\pi_{A,C}(R)$  будет:

Отношение R			Проекция $\pi_{A,C}(R)$	
A	B	C	A	C
a	b	c	a	c
c	a	d	c	d
c	b	d		

**Соединение** (join).

Соединение – бинарная операция над разносхемными отношениями. Эта операция определяет подмножество декартова произведения двух разносхемных отношений. Кортеж декартова произведения входит в результирующее отношение, если выполняется условие соединения F, которое задает соотношение значений атрибутов разных таблиц.

Если условием является равенство значений двух атрибутов исходных отношений, такая операция называется эквисоединением. Естественным называется эквисоединение по одинаковым атрибутам исходных отношений.

**Пример.** Пусть имеются отношения R(A,B,C) и S(A,D,E). Тогда естественное соединение  $R \bowtie S$  будет таким:

Отношение R			Отношение S			Соединение $R \bowtie S$				
A	B	C	A	D	E	A	B	C	D	E
a	b	c	g	h	a	c	a	d	b	c
c	a	d	c	b	c	c	h	c	b	c
c	h	c	h	d	d	g	b	d	h	a
g	b	d								

**Деление** (division).

Пусть отношение R содержит атрибуты  $\{r_1, r_2, \dots, r_k, r_{k+1}, \dots, r_n\}$ , а отношение S – атрибуты  $\{r_{k+1}, \dots, r_n\}$ . Тогда результирующее отношение содержит атрибуты  $\{r_1, r_2, \dots, r_k\}$ . Кортеж отношения R включается в результирующее отношение, если его декартово произведение с отношением S входит в R. Деление может быть выражено так:

$$\frac{R}{S} = \pi_{r_1, \dots, r_k}(R) - \pi_{r_1, \dots, r_k}((\pi_{r_1, \dots, r_k}(R) \times S) - R).$$

**Пример.** Пусть имеются отношения R(A,B,C) и S(A,B). Тогда частное R/S будет:

<u>Отношение R</u>				<u>Отношение S</u>		<u>Частное R/S</u>	
A	B	C	D	C	D	A	B
a	b	c	b	c	b	a	b
a	b	g	h	g	h	c	f
c	f	g	h				
c	f	c	b				
a	v	c	b				
c	v	g	h				

### 3. Лабораторные задания.

#### Упражнение 1.

Выполнить операции реляционной алгебры согласно индивидуальному заданию.

### 4. Контрольные вопросы

1. Дать определение «домен».
2. Дать определение отношению с математической точки зрения.
3. Привести примеры доменов.
4. Что такое степень отношения?
5. Что такое мощность отношения?
6. Что такое первичный, возможный, альтернативный ключи?
7. Операции реляционной алгебры.

## ЛАБОРАТОРНАЯ РАБОТА № 5

### Системы управления базами данных. Создание базы данных

**1. Цель работы:** получить теоретические и практические навыки реализации моделей данных средствами конкретной СУБД (*Microsoft Access* и *SQL Server Management Studio*)

#### 2. Теоретический материал для домашнего изучения.

##### *Системы управления базами данных Microsoft Access*

СУБД *Microsoft Access* является мощным средством для создания баз данных реляционного типа и работы с ними. База данных в *Access* хранится в файле с расширением *mdb* и содержит следующие объекты:

- *таблицы* – используются для представления объектов и связей между ними;
- *запрос* – средство отбора данных из таблиц по определенным критериям;
- *форма* – это окно, которое используется для просмотра, ввода или изменения данных в таблицах или отображения результатов запросов;

- *отчет* – это способ представления данных в удобном формате в виде печатного документа (например, с номерами страниц и заголовками);

- *макросы* – представляют собой набор команд, соответствующих стандартным действиям пользователя. Макросы предназначены для автоматизации часто выполняемых задач, например, при нажатии пользователем кнопки запускается макрос, который распечатывает отчет;

- *модули* – это подпрограммы на языке *Visual Basic*, которые всегда связаны с другими объектами базы данных и не могут выполняться отдельно, они могут быть запущены из форм, отчетов или запросов. С помощью подпрограмм можно изменять свойства и параметры объектов, выполнять обработку данных и т.д.

### ***Начало работы в Microsoft Access***

Запуск программы *Microsoft Access* осуществляется двойным нажатием соответствующего ярлыка на рабочем столе *Windows*, или нажатием кнопки *Пуск* и выбора в главном меню *Windows* пунктов *Все программы*  $\Rightarrow$  *Microsoft Office*  $\Rightarrow$  *Microsoft Office Access 2007*.

Окно *Microsoft Access 2007* устроено так же, как *Word*, *Excel* и другие программы пакета *Microsoft Office 2007*: вверху располагается лента инструментов, над ней – панель быстрого доступа, в левом углу – кнопка *Microsoft Office*, при нажатии на которую появляется меню, содержащее команды по открытию, сохранению и печати файла (рис. 5.1).

Лента инструментов состоит из следующих вкладок:

- *Главная* – содержит инструменты, позволяющие выбрать режим представления базы данных (таблицы или конструктора), копировать и перемещать данные, задать настройки шрифта, произвести некоторые операции с записями в базе данных, а также фильтрацию и сортировку данных;

- *Создание* – включает команды создания различных объектов базы данных – таблиц, форм, отчетов, запросов и других;

- *Внешние данные* – используется для импорта/экспорта данных из различных источников;

- *Работа с базами данных* – содержит инструменты отображения схемы данных, отображения зависимостей между объектами, анализа данных и т.д.

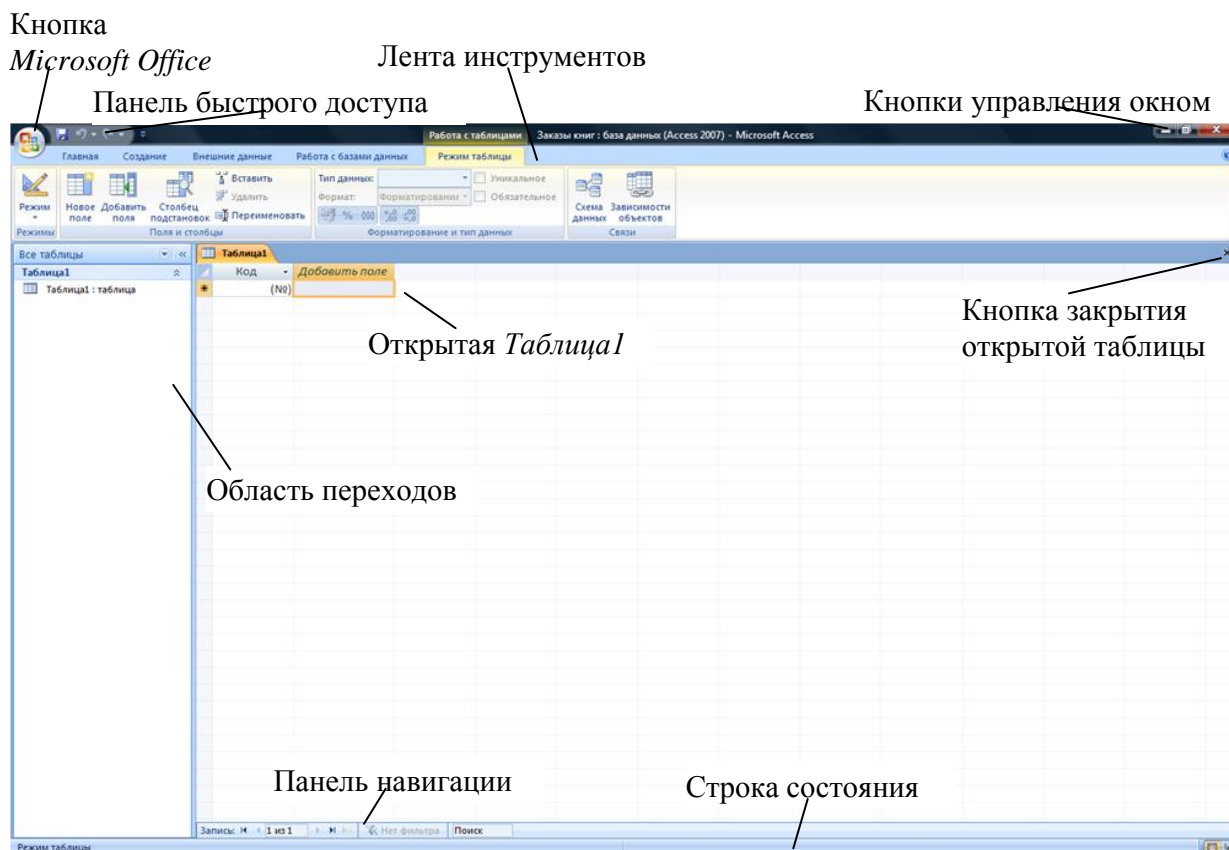


Рис. 5.1. Окно Access 2007

Центральную часть окна Access занимает окно открытой базы данных с ее элементами, внизу располагается строка состояния. В структуру окна текущей базы данных входят:

- *Область переходов* – размещена в левой части и отображает список существующих объектов, сгруппированных выбранным способом;
- *Вкладки открытых объектов* – занимают центральную рабочую область (на рис. 1 открыта *Таблица1*);
- *Панель навигации* – размещена на нижней границе окна объекта и используется для перемещения по записям открытого объекта базы данных.

### ***Создание таблиц***

В программе Access предусмотрены три способа создания таблиц:

- путем ввода данных;
- с использованием шаблона;
- с помощью конструктора таблиц, который применяется намного чаще других способов, потому что он предоставляет наибольшие возможности для создания новых таблиц.

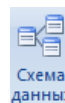
### ***Ввод данных в таблицы***

Заполнение таблицы базы данных *Access* требуемыми данными практически не отличается от ввода данных в электронную таблицу.

Для ввода данных в таблицу базы данных *Access* необходимо в области переходов дважды щелкнуть по ее названию, или щелкнуть правой кнопкой мыши на названии таблицы и в контекстном меню выбрать *Открыть*. Если таблица открыта в режиме конструктора, то для перехода на лист данных, нужно на вкладке *Конструктор* в группе *Режимы* нажать на кнопку *Режим - Режим таблицы*.

### ***Связи между таблицами***

Для установки связей между таблицами в *Access* существует окно



*Схема данных*, которое открывается нажатием кнопки *Схема данных* на вкладке *Работа с базами данных*. Как правило, связывается ключевое поле одной таблицы с соответствующим ему полем другой таблицы, которое называется *полем внешнего ключа*. Связанные поля могут иметь разные имена, однако у них должны быть одинаковые типы данных и одинаковые значения свойств.

При наличии связи между таблицами *Access* будет автоматически выбирать связанные данные из таблиц в запросах, отчетах и формах.

## ***Системы управления базами данных SQL Server Management Studio***

### **Создание и настройка базы данных**

В СУБД MS SQL Server существуют следующие варианты создания базы данных:

1. С использованием графического интерфейса *Management Studio*.
2. С использованием команд SQL.

*Создание базы данных* – это процесс указания имени файла, определения размеров и размещения файлов базы данных, а также определение параметров файла журнала транзакций.

*Создание БД с использованием графического интерфейса Management Studio:*

1. Запустить программу *SQL Server Management Studio*, в обозревателе объектов выбрать папку *Базы данных* и вызвать контекстное меню щелчком правой кнопки мыши.

2. В диалоговом окне *Создание базы данных* (рис. 5.2) ввести имя БД (в примере на рис. 5.2 – *Ivanov*) и нажать ОК.



3. В обозревателе объектов добавится новая база данных с именем Ivanov (рис. 5.3).

*Создание БД с использованием структурированного языка запросов SQL:*

1. Запустить программу **SQL Server Management Studio**, в меню выбрать создание запроса посредством T-SQL. В окне запроса ввести код:

CREATE DATABASE Ivanov \*// Ivanov – название создаваемой БД

on primary

(name=sqlcode,

filename='C:\SQLServerK\sql\_code.mdf', \*// прописывается путь физического расположения файла с кодом создаваемой БД

size=5,

maxsize=100,

filegrowth=0)

log on



(name=sqlcodelog,

filename='C:\SQLServerK\sql\_log.mdf', \*// прописывается путь расположения логического файла создаваемой БД

size=1,

maxsize=40,

filegrowth=0)

2. Нажмите в меню значок выполнить , а в меню Обозревателя объектов значок обновить . В Обозревателе объектов добавится новая база данных с именем Ivanov (рис. 5.3).

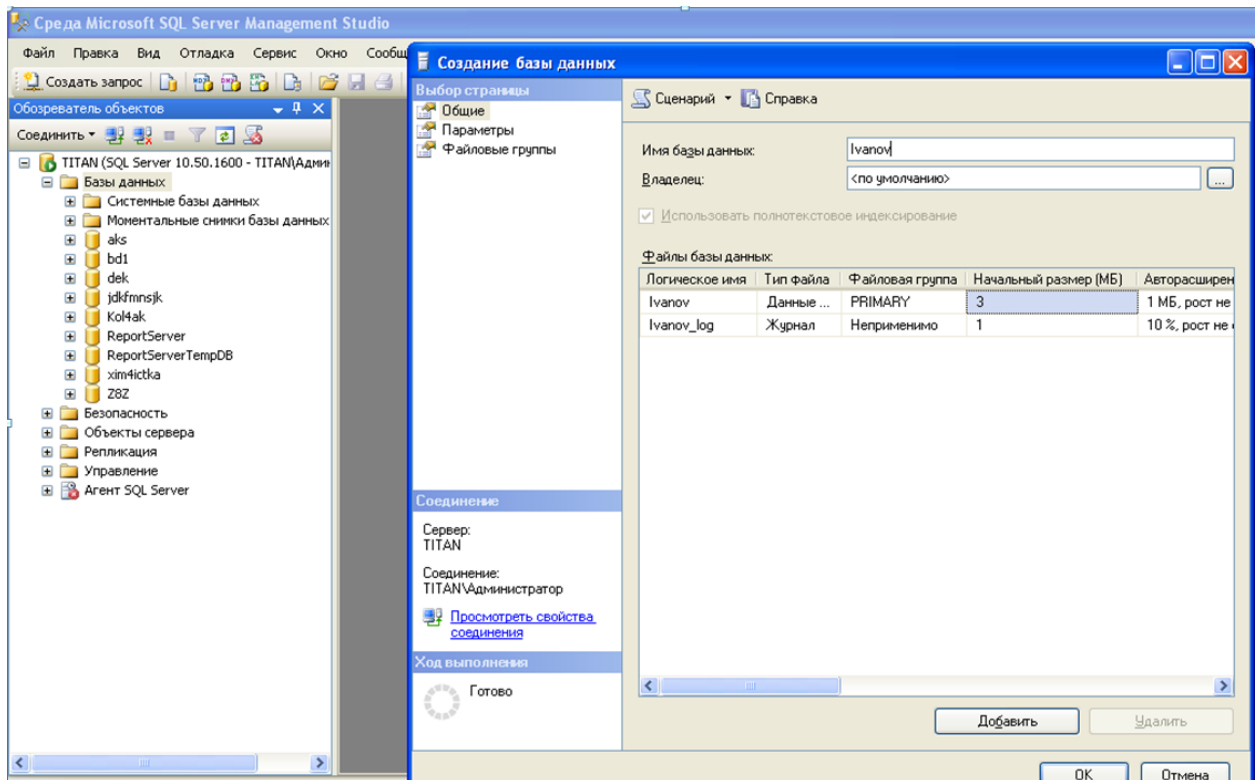


Рис. 5.2. Диалоговое окно Создание базы данных

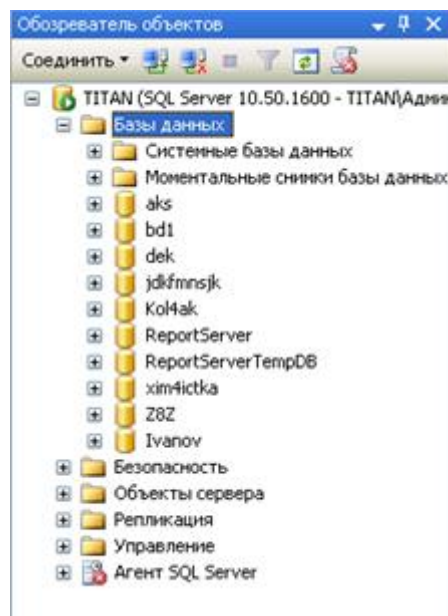


Рис. 5.3. Обозреватель объектов

**Примечание:** предварительно перед запросом на создание новой базы данных посредством языка SQL необходимо создать папку, которая будет прописываться в качестве физического и логического расположения новой БД.

Можно выделить три типа файлов в базах данных MS SQL Server:

1. *Первичные файлы данных.* Как правило, используется расширение MDF. В любой базе данных есть один первичный файл, который содержит данные и расположение всех остальных файлов БД.

2. *Вторичные файлы данных.* Как правило, используется расширение NDF. Вторичным является любой файл кроме первичного и файлов журналов. БД могут не содержать ни одного вторичного файла.

3. *Файлы журналов.* Как правило, используется расширение LDF. В каждой БД существует по меньшей мере один файл журнала. *Журнал транзакций* содержит сведения об изменениях, происходящих в БД, т.е. при совершении некоторой транзакции (операции) в этот журнал заносятся сведения. Со временем этот журнал становится все больше, поэтому требуется следить за его размером. Основное назначение журнала транзакций – это обеспечение целостности данных. Он позволяет отменять сделанные изменения в БД.

Для удобства администрирования и распределения нагрузки файлы можно объединять в файловые группы, которые делятся на два вида.

1. *Первичные файловые группы.* Сюда входят первичный файл и все файлы, которые явным образом не были помещены в другую группу.

2. *Пользовательские файловые группы* – это любая группа создаваемая пользователем в БД.

Файлы журналов не входят ни в одну файловую группу, они обрабатываются отдельно от обычных файлов.

Новая база данных представляет собой копию базы данных *model*, все параметры которой копируются в новую базу данных. По умолчанию базы данных имеют создавать только те пользователи, которым назначены роли *sysadmin* и *dbcreator*.

### ***Элементы языка SQL***

Фактически стандартным языком доступа к базам данных в настоящее время стал язык SQL (Structured Query Language).

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо "отношений" используются "таблицы", вместо "кортежей" - "строки", вместо "атрибутов" - "колонки" или "столбцы".

Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах отходит от нее.

Основу языка SQL составляют операторы, условно разбитые на несколько групп по выполняемым функциям:

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных.

Операторы DML (Data Manipulation Language) - операторы манипулирования данными.

Операторы защиты и управления данными, и др.

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных

CREATE SCHEMA - создать схему базы данных

DROP SCHEMA - удалить схему базы данных

CREATE TABLE - создать таблицу

ALTER TABLE - изменить таблицу

DROP TABLE - удалить таблицу

CREATE DOMAIN - создать домен

ALTER DOMAIN - изменить домен

DROP DOMAIN - удалить домен

CREATE COLLATION - создать последовательность

DROP COLLATION - удалить последовательность

CREATE VIEW - создать представление

DROP VIEW - удалить представление

Операторы DML (Data Manipulation Language) - операторы манипулирования данными

SELECT - отобразить строки из таблиц

INSERT - добавить строки в таблицу

UPDATE - изменить строки в таблице

DELETE - удалить строки в таблице

COMMIT - зафиксировать внесенные изменения

ROLLBACK - откатить внесенные изменения

Операторы защиты и управления данными

CREATE ASSERTION - создать ограничение

DROP ASSERTION - удалить ограничение

GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами

REVOKE - отменить привилегии пользователя или приложения

Кроме того, есть группы операторов установки параметров сеанса, получения информации о базе данных, операторы статического SQL, операторы динамического SQL.

Наиболее важными для пользователя являются операторы манипулирования данными (DML).

### ***Создание таблиц в MS SQL Server***

*Создание таблиц с использованием графического интерфейса Management Studio:*

1. Запустить программу ***SQL Server Management Studio***, в обозревателе объектов в дереве *Базы данных* выбрать папку с необходимой БД – вызвать контекстное меню щелчком правой кнопки мыши и создать таблицу.

2. В дереве выбранной *Базы данных* добавится новая таблица с заданным именем.

*Создание БД с использованием структурированного языка запросов SQL:*

1. Запустить программу ***SQL Server Management Studio***, в меню выбрать создание запроса посредством T-SQL. В окне запроса ввести код:

```
CREATE TABLE <имя таблицы>
```

```
(< имя поля1> <тип данных>[(<размер>)] [[NOT NULL] [DEFAULT  
<выражение>] [<ограничения целостности поля> ...]
```

```
[,<имя поля2> <тип данных> [(<размер>)] [[NOT NULL] [DEFAULT  
<выражение>] [<ограничения целостности поля>],...]
```

```
[,<ограничение целостности таблицы>]);
```

**Примечание.** Были приняты следующие обозначения для описания синтаксиса:

[] – содержимое этих скобок является необязательным;

<> – содержимое этих скобок заменяется соответствующими ключевыми словами, идентификаторами или выражениями;

... – все, что предшествует этим символам, может повторяться произвольное число раз;

..., – все, что предшествует этим символам, может повторяться произвольное число раз, каждое вхождение отделяется запятой;

Таблица 1

Описание элементов команды CREATE TABLE

Элемент	Описание
<имя таблицы>	Имя таблицы, идентификатор – должно быть уникальным в рамках БД.

<имя поля>	Имя поля (столбца) таблицы.
<тип данных>	Тип данных поля.
<размер>	Размер поля в символах (для текста и чисел).
<ограничения целостности>	PRIMARY KEY – первичный ключ (обязательный, уникальный и единственный на таблицу); UNIQUE – уникальное значение поля в пределах столбца таблицы; CHECK (<условие>) – условие, которому должно удовлетворять значение поля; REFERENCES <имя таблицы>[(<имя столбца>)] – внешний ключ.
<ограничения целостности таблицы>	CHECK (<условие>) – условие, которому должны удовлетворять значения одного или нескольких полей; FOREIGN KEY [(<список полей>)] REFERENCES<имя таблицы>[(<имя столбца>)] – внешний ключ; PRIMARY KEY (<список полей>) – первичный ключ; UNIQUE (<список полей>) – уникальное значение комбинации полей в пределах таблицы.

```

CREATE TABLE Изделие
(Шифр Integer primary key Not null,
Наименование Varchar(40) Not null,
"Номер проекта" Integer Not null,
"Количество осей шасси" Integer Not null);

CREATE TABLE Изготовление
("Шифр детали" Integer Not null,
"Шифр рабочего" Integer Not null,
"Шифр изделия" Integer Not null,
"Количество деталей" Integer Not null);

CREATE TABLE Покупатели
(Код_покупателя integer Primary key,
ФИО Varchar (50) Not null,
Телефон Varchar (50) not null,
Паспортные_данные Varchar (50) Not null,
Контактный_адрес Varchar (50) not null)

CREATE TABLE Продажи
(Код_маршрута Integer Not null,
Код_клиента Integer Not null,
Цена_путевки Money not null,
Количество_проданных Integer Not null,
Дата_продажи Date Not null);

CREATE TABLE Страны
(Код_страны Integer primary key Not null,
Название_страны Varchar(40) Not null,
Стоимость_визы Money not null);

```

Рис. 5.4 – Примеры создания таблиц в MS SQL Server

## Часто используемые типы данных

Тип данных	Размер (диапазон)	Описание
SMALLINT (M)	от -32768 до 32767 или от 0 до 65535	<p>Целое число. Может быть объявлено положительным с помощью ключевого слова UNSIGNED, тогда элементам столбца нельзя будет присвоить отрицательное значение. Необязательный параметр M - количество отводимых под число символов. Необязательный атрибут ZEROFILL позволяет свободные позиции по умолчанию заполнить нулями.</p> <p><b>Примеры:</b>  SMALLINT - хранит любое число в диапазоне от -32768 до 32767.  SMALLINT UNSIGNED - хранит любое число в диапазоне от 0 до 65535.  SMALLINT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить и пятизначные.  SMALLINT (4) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 0002.</p>
INT (M) или INTEGER (M)	от -2147683648 до 2147683648 или от 0 до 4294967295	<p>Аналогично предыдущему, но с большим диапазоном.</p> <p><b>Примеры:</b>  INT - хранит любое число в диапазоне от -2147683648 до 2147683648.  INT UNSIGNED - хранит любое число в диапазоне от 0 до 4294967295.  INT (4) - предполагается, что значения будут четырехзначные, но по факту будет хранить максимально возможные.  INT (5) ZEROFILL - свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 00002.</p>
DECIMAL (M,D) или DEC (M,D) или NUMERIC (M,D)	зависят от параметров M и D	<p>Используются для величин повышенной точности, например, для денежных данных. M - количество отводимых под число символов (максимальное значение - 64). D - количество знаков после запятой (максимальное значение - 30).</p> <p><b>Пример:</b>  DECIMAL (5,2) - будет хранить числа от -99,99 до 99,99.</p>
FLOAT (M,D)	мин. значение + (-) 1.175494351 * 10 <sup>-39</sup> макс. значение +(-) 3. 402823466 * 10 <sup>38</sup>	<p>Вещественное число (с плавающей точкой). Может иметь параметр UNSIGNED, запрещающий отрицательные числа, но диапазон значений от этого не изменится. M - количество отводимых под число символов. D - количество символов дробной части.</p> <p><b>Пример:</b>  FLOAT (5,2) - будет хранить числа из 5 символов, 2 из которых будут идти после запятой (например: 46,58).</p>
CHAR (M)	M символов	<p>Позволяет хранить строку фиксированной длины M. Значение M - от 0 до 65535.</p>

		<p><b>Примеры:</b>          CHAR (8) - хранит строки из 8 символов и занимает 8 байтов. Например, любое из следующих значений: ", 'Иван','Ирина', 'Сергей' будет занимать по 8 байтов памяти. А при попытке ввести значение 'Александра', оно будет усечено до 'Александ', т.е. до 8 символов.</p>
VARCHAR (M)	M символов	<p>Позволяет хранить переменные строки длиной L. Значение M - от 0 до 65535.</p> <p><b>Примеры:</b>          VARCHAR (3) - хранит строки максимум из 3 символов, но пустая строка " занимает 1 байт памяти, строка 'а' - 2 байта, строк 'аа' - 3 байта, строка 'ааа' - 4 байта. Значение более 3 символов будет усечено до 3.</p>
BLOB, TEXT	216-1 символов	<p>Позволяют хранить большие объемы текста. Причем тип TEXT используется для хранения именно текста, а BLOB - для хранения изображений, звука, электронных документов и т.д.</p>
DATE	от '1000-01-01' до '9999-12-31'	<p>Предназначен для хранения даты. В качестве первого значения указывается год в формате "YYYY", через дефис - месяц в формате "MM", а затем день в формате "DD". В качестве разделителя может выступать не только дефис, а любой символ отличный от цифры.</p>
TIME	от '-838:59:59' до '838:59:59'	<p>Предназначен для хранения времени суток. Значение вводится и хранится в привычном формате - hh:mm:ss, где hh - часы, mm - минуты, ss - секунды. В качестве разделителя может выступать любой символ отличный от цифры.</p>
DATETIME	от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'	<p>Предназначен для хранения и даты и времени суток. Значение вводится и хранится в формате - YYYY-MM-DD hh:mm:ss. В качестве разделителей могут выступать любые символы отличные от цифры.</p>
TIMESTAMP	от '1970-01-01 00:00:00' до '2037-12-31 23:59:59'	<p>Предназначен для хранения даты и времени суток в виде количества секунд, прошедших с полуночи 1 января 1970 года (начало эпохи UNIX).</p>
YEAR (M)	от 1970 до 2069 для M=2 и от 1901 до 2155 для M=4	<p>Предназначен для хранения года. M - задает формат года. <b>Например</b>, YEAR (2) - 70, а YEAR (4) - 1970. Если параметр M не указан, то по умолчанию считается, что он равен 4.</p>
MONEY	От -922 337 203 685 477,5808 до 922 337 203 685 477,5807	<p>Типы данных money имеют точность до одной десятитысячной денежной единицы, которую они представляют.</p>



### 3. Лабораторные задания.

#### Упражнение 1

Согласно индивидуальному заданию и разработанной в лабораторной работе №3 логической модели создать базу данных, содержащую таблицы, в СУБД *Microsoft Access*.

#### Упражнение 2

Согласно индивидуальному заданию и разработанной в лабораторной работе №3 логической модели создать с помощью программного кода базу данных в СУБД *SQL Server Management Studio*. В этой базе данных через программный код создать таблицы.

### 4. Контрольные вопросы

8. Что такое база данных?
9. Что такое первичный ключ?
10. Какие основные компоненты базы данных *Access*?
11. Какие основные компоненты базы данных *SQL Server*?
12. Какими способами в *Access* можно создать таблицу?
13. Какими способами в *SQL Server* можно создать таблицу?
14. Какими способами в *Access* можно вводить информацию в базу данных?
15. Какими способами в *SQL Server* можно вводить информацию в базу данных?

## ЛАБОРАТОРНАЯ РАБОТА № 6

### Создание связей между таблицами реляционной базы данных

**1. Цель работы:** получить представление о реляционной модели баз данных и навыки создания связей между таблицами реляционной базы данных.

### 2. Теоретический материал для домашнего изучения.

#### Внешние и родительские ключи

Реляционная база данных редко состоит из одной таблицы. При создании нескольких таблиц со связанной информацией можно выполнять более сложные и мощные операции над данными. Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчетов.

Например, в базе данных нужно хранить информацию о студентах нескольких учебных групп. Это можно сделать так:

1. Создать таблицу *Студенты*. Таблица *Студенты* должна содержать поле *Код\_гр*. В поле *Код\_гр* будем указывать код той группы, в которой числится данный студент. Поле *Код\_гр* таблицы *Студенты* ссылается на поле *Код* таблицы *Группы*. Если в таблице *Группы* нет записи с кодом 5, то в таблице *Студенты* не может присутствовать запись с таким значением поля *Код\_гр*. В противном случае будет нарушаться ссылочная целостность базы данных.

Когда поле таблицы ссылается на поле другой таблицы, оно называется **внешним ключом**. Поле, на которое ссылается внешний ключ, называется его **родительским ключом**.

Если внешний ключ не является первичным, то его значение может повторяться в нескольких строках. Например, в таблице *Студенты* есть несколько строк со значением поля *Код\_гр* равным 2. Значение родительского ключа, равное 2, соответствует группе *Группа2*. Образованная связь называется «**Один-ко-многим**», т.е. в одной группе могут числиться несколько студентов, а каждый студент может числиться только в одной группе (рис. 6.1).

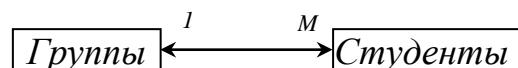


Рис. 6.1. Связь «один-ко-многим» между отношениями *Группы* и *Студенты*

Связь «**один-ко-многим**» - наиболее часто используемый тип связи между таблицами. В такой связи каждой записи некоторой таблицы *A* может соответствовать несколько записей в таблице *B* (поля с этими записями называются ключами), а запись в таблице *B* не может иметь более одной соответствующей ей записи в таблице *A*.

Более сложная связь, при которой каждой записи таблицы *A* может соответствовать несколько записей таблицы *B*, а каждой записи таблицы *B* может соответствовать несколько записей таблицы *A* называется «**многие-ко-многим**». В нашем примере такую связь нужно установить между таблицами *Группы* и *Дисциплины*, если в базе данных необходимо хранить информацию о том, какие дисциплины изучаются в различных группах. В каждой группе изучается несколько дисциплин и каждая дисциплина изучается в нескольких группах (рис. 6.2).

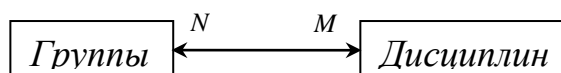


Рис. 6.2. Связь «многие-ко-многим» между таблицами *Группы* и *Дисциплины*

В реляционных базах данных такая схема реализуется только с помощью третьей (связующей) таблицы (назовем ее *Группы\_Дисциплины*), ключ которой состоит, по крайней мере, из двух полей, одно из которых является общим с таблицей *Группы*, а другое - общим с таблицей *Дисциплины*. Т.е. связь «многие-ко-многим» между таблицами *Группы* и *Дисциплины* сводится к двум связям «один-ко-многим» между таблицами *Группы*, *Группы\_Дисциплины* и *Дисциплины*, *Группы\_Дисциплины* (рис. 6.3).

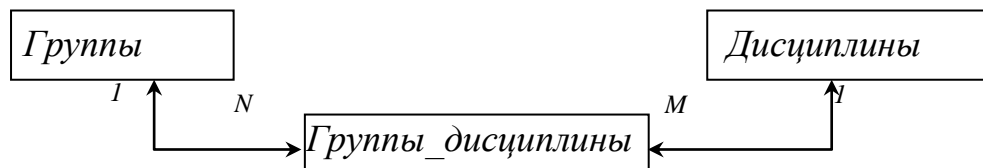


Рис. 6.3. Реализация связи «многие-ко-многим» между таблицами *Группы* и *Дисциплины*

При связи «один-к-одному» запись в таблице *A* может иметь не более одной связанной записи в таблице *B* и наоборот. Этот тип связи используется не очень часто, поскольку такие данные могут быть помещены в одну таблицу. Связь с отношением «один-к-одному» применяют для разделения очень широких таблиц, а также для отделения части таблицы в целях ее защиты.

#### **Создание схемы** базы данных в *SQL Server Management Studio*:

1. Запустить программу *SQL Server Management Studio*, в обозревателе объектов выбрать папку «**нужной**» Базы данных и нажав на пиктограмму «+» развернуть для просмотра.

2. Найти **диаграмму базы данных**, правой кнопкой мыши вызвать контекстное меню и выбрать *Создать диаграмму базы данных*. В открывшейся рабочей области добавить необходимые таблицы и связать их (добавление таблиц и связей производится аналогично работе со схемой базы данных в *Microsoft Access*).

#### **Заполнение таблиц** базы данных в *SQL Server Management Studio*.

INSERT – добавление записи в таблицу. Синтаксис команды:

INSERT INTO <имя таблицы>[(<имя поля>...)]

VALUES(<список выражений>) | <запрос>;

Предварительно необходимо указать БД – USE <имя БД>

### Пример.

```

insert into Сделки(Код_сделки,Код_покупателя,Код_объекта,Наименование,Дата_сделки)
values
('310','119','210','Съём','15.11.2016'),
('311','118','211','Съём','20.12.2016'),
('312','117','212','Покупка','26.12.2016'),
('313','116','213','Покупка','13.01.2017'),
('314','115','214','Покупка','04.02.2017'),
('315','114','215','Обмен','22.02.2017'),
('316','113','216','Обмен','13.03.2017'),
('317','112','217','Покупка','21.03.2017'),
('318','111','218','Покупка','28.03.2017'),
('319','110','219','Покупка','05.04.2016')

```

Рис. 6.4. Пример заполнения таблицы

Для просмотра результатов заполнения необходимо создать запрос со следующей командой:

**SELECT \* FROM Сделки** \*/ «Сделки» - таблица, результат заполнения которой будем просматривать

	Код_объекта	Код_категории	Площадь	Адрес	Стоимость
1	110	1	55 м?	Воронеж, р-н Железнодорожный, ул Калининградская...	1 550 000р
2	111	1	67 м?	Воронеж, р-н Левобережный, ул. Полевая, д. 44а	1 880 000р
3	112	1	33 м?	Воронеж, р-н Коминтерновский, ул. Хользунова, д. 99Б	1 195 000р
4	113	1	70 м?	Воронеж, р-н Коминтерновский, ул Генерала Лизюков...	2 700 000р
5	114	1	107 м?	Воронеж, р-н Левобережный, улица МОПРа, 8Б	6 900 000р
6	115	1	94 м?	Воронеж, р-н Центральный, ул Короленко, 5	7 000 000р
7	116	2	28.7 сот.	Воронеж, с. Староживотинное, ул. Фамильные усадьб...	3 000 000р
8	117	2	15 сот.	Воронеж, Воронежская область, Рамонский район, С...	600 000р
9	118	3	12 м?	Воронеж, р-н Ленинский, ул.20-летия Октября д.48 (ко...	4 000 р/мес.
10	119	3	30 м?	Воронеж, р-н Левобережный, Ленинский пр-кт, 12	6 500 р/мес.

Рис. 6.6. Таблица «Сделки»

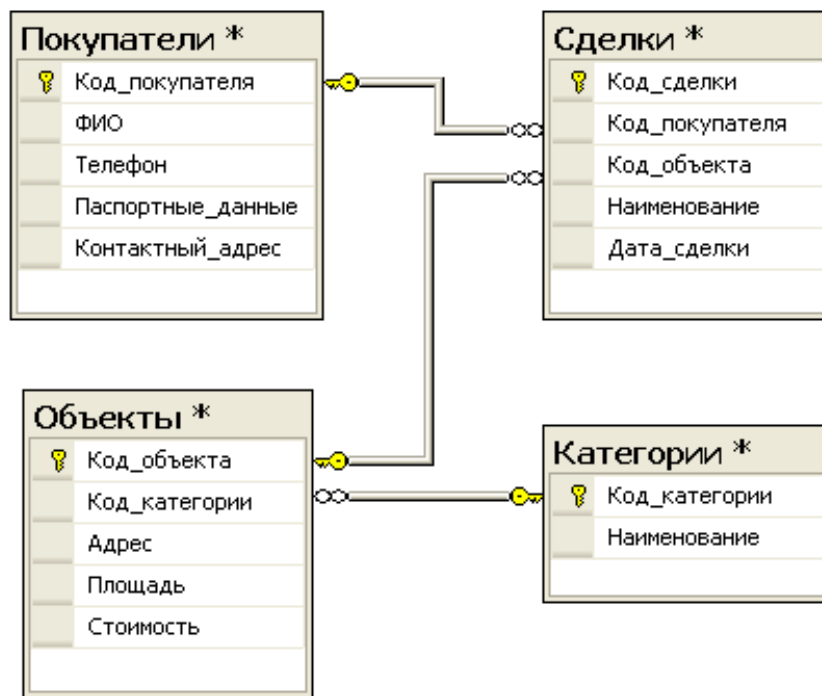


Рис. 6.5. Схема базы данных (на примере БД «Риэлтерская фирма»)

### 3. Лабораторные задания.

#### Упражнение 1

Для созданных в лабораторной работе №5 таблиц создать связи и заполнить таблицы данными.

#### 4. Контрольные вопросы

1. Для чего создаются связи между таблицами базы данных?
2. Может ли внешний ключ повторяться в нескольких строках таблицы?
3. Что такое ссылочная целостность базы данных? Как она обеспечивается в Access? Как она обеспечивается в SQL Server?
4. Какие типы связей между таблицами поддерживаются в реляционной модели.
5. Как обеспечивается ссылочная целостность в SQL Server?

**ЛАБОРАТОРНАЯ РАБОТА № 7****Создание простых запросов**

**1. Цель работы:** получить практические навыки формирования запросов на выборку.

**2. Теоретический материал для домашнего изучения.**

Запрос на выборку - это команда, которая формируется для СУБД и требует предоставить определенную указанную информацию. Эта информация обычно выводится непосредственно на экран монитора или используется в качестве исходных данных для другой команды или процесса.

Все запросы на выборку в SQL конструируются на базе одной команды. Эта команда называется SELECT.

Создавать и редактировать запросы Access можно двумя способами: непосредственно используя структурированный язык запросов SQL, а также используя специальное средство – конструктор запросов.

В простейшей форме команда SELECT дает инструкцию базе данных для поиска информации в таблице.

После ключевого слова SELECT нужно указать список столбцов таблицы, которые должны быть представлены в результате выполнения запроса. После ключевого слова FROM указывается имя таблицы, которая используется как источник информации.

**Пример.**

Получить список, содержащий фамилии и телефоны всех студентов:

```
SELECT Фамилия, телефон  
FROM Студенты
```

Если необходимо увидеть все столбцы таблицы, существует упрощенный вариант сделать это. Можно использовать символ "\*", который заменяет полный список столбцов.

**Пример.**

Вывести всю информацию из таблицы Студенты:

```
SELECT *  
FROM Студенты
```

***Вывод с сортировкой – предложение ORDER BY***

Предложение ORDER BY применяется для сортировки выходных данных. После этого предложения указывается имя поля, которое необходимо упорядочить. По умолчанию сортировка ведется по возрастанию. Если нужно сортировать по убыванию, то после имени поля, по которому ведется сортировка, нужно указать служебное слово DESC.

**Пример.**

Вывод списка предметов с сортировкой по названию

```
SELECT код, название
FROM Дисциплины
ORDER BY название
```

***Выборка из таблицы с условием - предложение WHERE***

Предложение WHERE позволяет определить условие, которое может быть либо истинным, либо ложным для каждой строки таблицы.

Команда SELECT извлекает только те строки из таблицы, для которых условие имеет значение “истина”. В этом предложении можно использовать реляционные операторы: =, <, >, <=, >=, < >. Для указания нескольких условий нужно использовать булевы операторы AND, OR, NOT.

**Пример.**

Получить информацию о студентах с фамилией Арбузов

```
SELECT *
FROM Студенты
WHERE фамилия = `Арбузов`
```

***Поиск подстрок – оператор LIKE***

Этот оператор позволяет просматривать поля для выяснения того, входит ли заданная подстрока в указанное поле.

**Пример.**

Получить список кафедр, расположенных на первом этаже (номер аудитории начинается на 1)

```
SELECT название
FROM кафедры
WHERE кабинет LIKE "1*"
```

### ***Запросы с соединением***

Таблицы соединяются по полям, имеющим одинаковую семантику, по одинаковому значению. Обычно соединяемые таблицы находятся в связи один-ко-многим. После того как таблицы соединены в одну таблицу, из нее производится выборка.

#### ***Простое соединение***

Условие соединения можно указать в предложении WHERE:

##### **Пример.**

Вывести фамилии преподавателей и названий дисциплин, которые они ведут.

```
SELECT Фамилия, Название
FROM Преподаватели, Дисциплины
WHERE Дисциплины.код =Преподаватели.код_дисц
```

Условие соединения можно также указать в предложении FROM:

```
SELECT Фамилия, Название
FROM Преподаватели INNER JOIN Дисциплины ON Дисциплины.код =
Преподаватели.код_дисц
```

#### ***Соединение с условием отбора***

##### **Пример .**

Показать преподавателей, преподающих информатику.

```
SELECT Преподаватели.Фамилия, дисциплины.название
FROM Преподаватели INNER JOIN Дисциплины ON Дисциплины.код
=Преподаватели.код_дисц
WHERE дисциплины.название='Информатика'
```

#### ***Соединение 3-х и более таблиц***

Это соединение обычно выполняется между таблицами со связью многие - ко - многим или когда в условии отбора данных из двух связанных таблиц присутствуют данные из 3-ей таблицы.

##### **Пример.**

Получить список всех студентов с отметками по физике



***Первый способ:***

```
SELECT студенты.фамилия, сессия.оценка
      FROM студенты, сессия, дисциплины
      WHERE студенты.код=сессия.код_студ
            AND дисциплины.код = сессия.код_дисц
            AND дисциплины.название ='физика'
```

***Второй способ:***

```
SELECT студенты.фамилия, студенты. ммя, студенты.отчество,
      сессия.оценка
FROM студенты INNER JOIN
      ( дисциплины INNER JOIN сессия
      ON дисциплины.код = сессия.код_дисц)
ON студенты.код=сессия.код_студ
WHERE дисциплины.название ='физика'
```

**3. Лабораторные задания.*****Упражнение 1.***

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию создать не менее 7 простых запросов для каждой БД, используя операторы, приведенные выше.

**4. Контрольные вопросы**

1. С использованием какой команды SQL конструируются запросы на выборку? Объясните формат записи этой команды.
2. Какими способами в Access можно конструировать запросы?
3. Какими способами в SQL Server можно конструировать запросы?
4. Какими способами можно записывать запросы с соединением?
5. Какой способ соединения таблиц используется при создании запроса с помощью конструктора?

**ЛАБОРАТОРНАЯ РАБОТА № 8****Функции агрегирования**

**1. Цель работы:** получить практические навыки использования агрегатных функций внутри запросов на выборку.

**2. Теоретический материал для домашнего изучения.**

Агрегатные функции применяются для обобщения значений одного поля таблицы. Они дают единственное значение для целой группы строк таблицы.

Ниже приводится список этих функций:

**COUNT** определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NULL-значениями.

**Пример.**

Сколько всего студентов в базе данных?

```
SELECT COUNT(*) AS [количество студентов]
FROM студенты;
```

**SUM** вычисляет арифметическую сумму всех выбранных значений данного поля.

**Пример.**

Посчитать количество часов учебной нагрузки у преподавателя с кодом 1.

```
SELECT Sum(группы_дисциплины.часов)
FROM группы_дисциплины
WHERE группы_дисциплины.код_преп = 1
```

**AVG** вычисляет среднее арифметическое всех выбранных значений данного поля.

**Пример.**

Вычислить средний балл для всех студентов по математике:

```
SELECT AVG(оценка) AS [средняя оценка]
FROM сессия INNER JOIN дисциплины
ON сессия.код_дисц = дисциплины.код
WHERE название = 'Математика'
```

**MAX, MIN** – возвращают максимальное и минимальное значения из всех выбранных значений данного поля.

**Пример.**

Поиск максимального кода студента:

```
SELECT MAX(код)
FROM студенты
```

Аналогично функции MAX работает функция *MIN*.

### ***Группировка данных***

Данные в одной таблице группируются по какому-то полю, значение которого не уникально, обычно являющегося вторичным ключом.

#### **Пример.**

Найти, сколько студентов в каждой группе

```
SELECT код_группы, COUNT(*)
FROM студенты
GROUP BY код_группы
```

Группировка ведется по полю код\_группы - все данные с одинаковым значением этого поля группируются и подсчитывается число записей в каждой такой группе.

### ***Отбор групп***

Выбирать можно не только записи, но и группы. Для этого используется предложение *HAVING*. Использовать агрегатные функции в предложении WHERE нельзя.

#### **Пример.**

Вывести список групп, в которых количество пятерок у студентов больше 6 на группу.

```
SELECT название, COUNT(*) AS пятерок
FROM группы, студенты, сессия
WHERE группы.код=студенты.код_группы
AND студенты.код=сессия.код_студ
AND сессия.оценка=5
GROUP BY группы.название
HAVING COUNT(*) > 6
```

В результате должны получить

название	пятерок
Группа1	7

Рис. 8.1. Список групп, в которых количество пятерок больше шести

Здесь сначала создается виртуальная таблица с количеством пятерок для каждой группы. После этого из этой таблицы выбираются только те записи которые удовлетворяют условию в предложении HAVING.

### 3. Лабораторные задания.

#### *Упражнение 1.*

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию создать не менее 7 запросов для каждой БД, используя операторы, приведенные выше.

### 4. Контрольные вопросы

1. Можно ли не указывать в предложении SELECT поле, по которому производится группировка?
2. Можно ли указывать условие отбора для групп в предложении WHERE?
3. Можно ли группировать данные по нескольким полям? Приведите пример.

## ЛАБОРАТОРНАЯ РАБОТА № 9

### Создание подзапросов

**1. Цель работы:** Получить практические навыки работы по формированию подзапросов.

#### **2. Теоретический материал для домашнего изучения.**

SQL позволяет вкладывать запросы друг в друга. Обычно внутренний запрос генерирует значения, которые тестируются на предмет истинности условия внешнего запроса. Подзапрос должен возвращать множество таких объектов, к которым можно применить условие внешнего запроса.

#### *Подзапросы с подмножеством*

#### **Пример.**

Получить ФИО и названия групп студентов – задолжников:

```

SELECT фамилия, имя, отчество, название
FROM студенты INNER JOIN группы ON студенты.код_группы =
группы.код
WHERE студенты.код IN ( SELECT код_студ FROM сессия
WHERE оценка = 2 )

```

Во внутреннем запросе (который и называется подзапросом), строится виртуальная таблица (подмножество), состоящая из одного столбца и включающая в себя коды всех задолжников. Во внешнем запросе поле код каждой записи таблицы Студенты анализируется на принадлежность к этому подмножеству. Если значение поля код из таблицы Студенты принадлежит подмножеству, возвращаемому внутренним запросом, то соответствующая запись из таблицы, полученной путем соединения таблиц Студенты и Группы вносится в результат.

### ***Использование агрегатных функций в подзапросах***

Запрос, использующий единственную агрегатную функцию без предложения GROUP BY, дает в результате единственное значение. Это значение можно использовать в основном предикате.

#### **Пример.**

Для заданной дисциплины узнать фамилии студентов получивших оценку, большую средней оценки по данной дисциплине:

```

SELECT фамилия
FROM дисциплины INNER JOIN(студенты INNER JOIN сессия ON
студенты.код = сессия.код_студ) ON дисциплины.код = сессия.код_дисц
WHERE название = [введите название дисциплины] and оценка >
(select AVG(оценка) FROM сессия)

```

В данном случае подзапрос выполняется только один раз. Возвращенное им значение используется во внешнем запросе.

#### **Пример.**

Показать дисциплину, по которой студенты получили максимальное количество двоек.

*Агрегатные функции нельзя вкладывать друг в друга, поэтому выражение  $\text{Max}(\text{Count}(\text{оценка}))$  будет неправильным. Для создания этого запроса нужно поступать следующим образом:*

1. Создать запрос, который будет возвращать количество двоек для каждой группы:

```
SELECT название, count(оценка) AS двоек
FROM сессия INNER JOIN дисциплины
      ON сессия.код_дисц = дисциплины.код
WHERE оценка=2
GROUP BY название
```

2. Сохраните запрос с названием, например, дисциплины\_двойки.

3. Используйте полученный результат в качестве входной таблицы в другом запросе, который будет выбирать название дисциплины, с максимальным количеством двоек.

```
SELECT название
FROM дисциплины_двойки
WHERE двоек = (SELECT MAX(двоек)
      FROM дисциплины_двойки);
```

### ***Подзапросы с EXISTS и NOT EXISTS***

EXISTS означает примерно «если подмножество не пусто», соответственно NOT EXISTS означает «если подмножество пусто». Используются в подзапросах.

#### **Пример.**

Получить ФИО и названия групп студентов – задолжников, используя оператор EXISTS:

```
SELECT фамилия, имя, отчество, название
FROM студенты INNER JOIN группы
      ON студенты.код_группы = группы.код
WHERE EXISTS
      ( SELECT *
      FROM сессия
      WHERE оценка = 2 AND студенты.код= сессия.код_студ)
```

Во внешнем запросе перебираются строки таблицы, полученной путем объединения таблиц студенты и группы, и для каждой записи этой таблицы выполняется подзапрос (так называемый связанный подзапрос) – ищутся строки в таблице сессия с кодом данного студента. Если внутренний запрос возвращает одну или более записей, то внешний запрос заносит строку с информацией о данном студенте в результат.

### **Пример.**

Найти всех студентов из группы Гр2, которые сдали сессию без двоек и троек.

```
SELECT *
FROM студенты
WHERE код_группы =
( SELECT код FROM Группы WHERE название = 'Гр2' )
AND NOT EXISTS
( SELECT *
FROM сессия
WHERE студенты.код = сессия.код_студ AND оценка <= 3 )
```

Во внешнем запросе для каждого студента проверяются два условия. В каждом условии присутствует подзапрос. Первый подзапрос возвращает таблицу, состоящую из одной ячейки, в которой записан код группы “Гр2”. Первое условие проверяет равен ли код группы данного студента коду группы “Гр2”.

Второй подзапрос возвращает записи таблицы сессия, в которых код студента равен коду проверяемого студента и оценка  $\leq 3$ . Второе условие истинно, если второй подзапрос вернет пустую таблицу.

Если оба условия для данного студента выполняются, то внешний запрос заносит информацию о студенте в результирующую таблицу:

## **3. Лабораторные задания.**

### **Упражнение 1.**

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию создать не менее 5 подзапросов для каждой БД, используя операторы, приведенные выше.

#### 4. Контрольные вопросы

1. Можно ли запрос из примера 5 записать следующим образом:  

```
SELECT *
FROM студенты
WHERE код_группы IN
( SELECT код FROM Группы WHERE название = 'Гр2' )
AND EXISTS
( SELECT *
FROM сессия
WHERE студенты.код = сессия.код_студ AND оценка >3 )
```
2. В каких случаях вложенный подзапрос выполняется один раз, а в каких несколько?
3. Можно ли вкладывать одну агрегатную функцию в другую?

### ЛАБОРАТОРНАЯ РАБОТА № 10

#### Запросы на изменение

**1. Цель работы:** Получить практические навыки использования запросов на изменение.

#### 2. Теоретический материал для домашнего изучения.

##### *Вставка записей в таблицу*

Все строки в SQL вводятся при помощи команды обновления INSERT. В простейшем случае команда INSERT имеет синтаксис:

```
INSERT INTO <имя таблицы> (<поле1>, <поле2>...)
Values (<значение>, <значение>... );
```

##### **Пример.**

Добавить нового студента:

```
INSERT INTO студенты (код, фамилия, имя, отчество, телефон,
код_группы)
VALUES ( 11, "Сидоров", "Олег", "Петрович", "34-56-78", 2)
```

##### *Вставка записей из другой таблицы*

Команду INSERT можно применить для того, чтобы извлечь значения из одной таблицы и разместить их в другой, воспользовавшись для этого



запросом. Для этого достаточно заменить предложение VALUES на соответствующий запрос.

**Пример.**

Физика читается у всех групп преподавателем Мироновым (его код = 2). Внести соответствующие записи в таблицу Группы\_Дисциплины.

```
INSERT INTO Группы_Дисциплины (код_дисц,код_группы,код_преп)
SELECT дисциплины.код,группы.код , 2
FROM Дисциплины,Группы
WHERE Дисциплины.название = 'Физика' AND Группы.N_курса=1
```

***Обновление таблиц***

По команде UPDATE можно изменять некоторые или все значения в существующей строке. Эта команда содержит предложение UPDATE, позволяющее указать имя таблицы, для которой выполняется операция, и SET предложение, определяющее изменение, которое необходимо выполнить для определенного столбца.

***Прямое обновление***

**Пример.**

Студента Иванова перевести из группы Гр1 в группу Гр2.

```
UPDATE студенты
SET студенты.код_группы = 2
WHERE студенты.фамилия = "Иванов" AND студенты.код_группы
=1
```

***Обновление с подзапросом***

Источником данных для команды UPDATE может быть результат, возвращаемый подзапросом.

**Пример.**

Студентам, сдавшим сессию без троек, назначить стипендию (предполагается, что нужно сдать 4 экзамена)

```
UPDATE студенты SET стипендия = YES
WHERE 4 = ( SELECT count(*) FROM сессия WHERE
сессия.код_студ = студенты.код) AND
NOT EXISTS
```

(SELECT \* FROM сессия WHERE студенты.код = сессия.код\_студ AND (сессия.оценка < 4 OR сессия.оценка IS NULL) );

В первом подзапросе проверяем, все ли экзамены сдавал студент. Во втором – не получил ли студент хотя одну оценку < 4 или по всем ли экзаменам проставлены оценки.

### ***Удаление данных***

Для удаления записей из таблицы используется команда DELETE.

Структура запросов на удаление аналогична структуре запросов на обновление.

### ***Простое удаление***

#### **Пример.**

Удалить все данные из таблицы Сессия:

```
DELETE *
FROM сессия
```

#### **Пример.**

Удалить кафедру истории.

```
DELETE
FROM кафедры
WHERE название ='информатики'
```

При выполнении этого запроса может появиться сообщение о возможном нарушении ссылочной целостности при удалении записи из таблицы. Такое сообщение возникает при возможном нарушении ссылочной целостности при удалении записи из таблицы. Поддерживать ссылочную целостность в таких ситуациях можно путем автоматического удаления связанных записей из присоединенных таблиц.

Для использования этой функции в свойствах связи между таблицами Кафедры и Преподаватели нужно установить каскадное удаление связанных записей. Теперь при удалении кафедры будут удаляться все преподаватели этой кафедры. Ссылочная целостность базы данных будет при этом сохранена.

### ***Удаление с подзапросом***

Удалить кафедру, если на которой не числится ни один преподаватель.

```
DELETE
FROM кафедры
WHERE NOT EXISTS
( SELECT код_кафедры
FROM преподаватели
WHERE кафедры.код=преподаватели.код_кафедры )
```

### **3. Лабораторные задания.**

#### ***Упражнение 1.***

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию создать не менее 5 запросов на изменение для каждой БД.

#### **4. Контрольные вопросы**

1. Будет ли работать следующий запрос?

```
DELETE
FROM преподаватели
WHERE NOT EXISTS ( SELECT код FROM кафедры
WHERE кафедры.код=преподаватели.код_кафедры )
```

2. В каком случае возникает нарушение ссылочной целостности?

3. Для чего служит команда UPDATE?

## **ЛАБОРАТОРНАЯ РАБОТА № 11**

### **Организация многопользовательского приложения архитектуры «файл-сервер»**

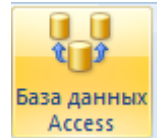
**1. Цель работы:** Получить навыки работы с основными средствами СУБД для организации многопользовательского приложения архитектуры файл-сервер

#### **2. Теоретический материал для домашнего изучения.**

По умолчанию Access хранит все объекты данных и интерфейса в одном общем файле MDB. В случае небольшого однопользовательского приложения это удобно, но в многопользовательской среде такая структура приложения может сильно снизить его производительность. Каждый раз, когда пользователю понадобится какой-нибудь объект, например форма, Jet

будет пересылать его по сети. В условиях реальной эксплуатации приложения, когда изменяются только данные, а все объекты интерфейса остаются неизменными, это ведет к неоправданному увеличению сетевого трафика. Чтобы этого избежать, базу данных Access можно разделить на две части: файл объектов данных, содержащий таблицы и файл объектов приложения, который содержит все остальные объекты – запросы, формы, отчеты, макросы, модули.

#### Отделение данных от приложения



Разделить базу данных можно с помощью пиктограммы на вкладке Работа с базами данных. После этого базу данных приложения (со всеми объектами, кроме таблиц) нужно связать с таблицами из серверной базы данных с помощью команды. При создании связи с таблицами имя базы данных удобно задать в формате UNC, например, \\имяСервера\ИмяОбщейПапки\Данные.MDB. Далее копии файла объектов приложения размещаются на всех рабочих станциях, а на файл-сервере оставляется файл данных, т.е. базу данных, состоящую из одних таблиц.

Полученное приложение будет иметь архитектуру файл-сервер. При такой архитектуре копия ядра базы данных работает на каждой рабочей станции и запрашивает файлы базы данных, расположенные на централизованном сервере.

У этого подхода имеются определенные преимущества:

- 1) более высокая производительность (особенно пользовательского интерфейса);
- 2) возможность создавать временные таблицы в базах данных, приложения на рабочих станциях;
- 3) разбиение базы данных упрощает установку новых версий приложения, поскольку данные находятся в отдельном файле и можно заменить файл приложения новым, не опасаясь их стереть или испортить;
- 4) каждый пользователь может настроить пользовательский интерфейс в соответствии со своими индивидуальными потребностями.

В случае перемещения базы данных связи с таблицами необходимо обновлять. Это можно делать интерактивно с помощью Диспетчера связанных таблиц на вкладке Работа с базами данных.

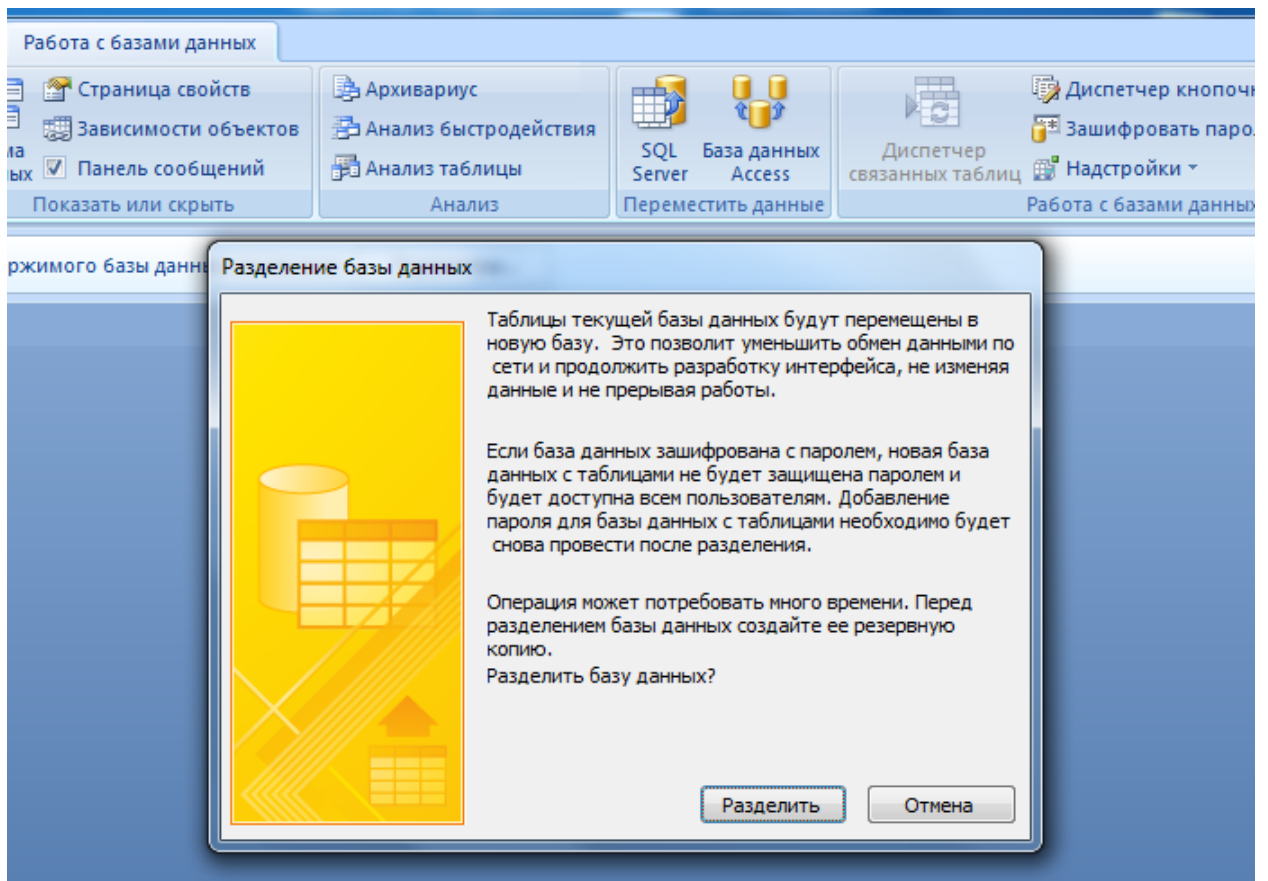


Рис. 11.1. Запуск разделения БД

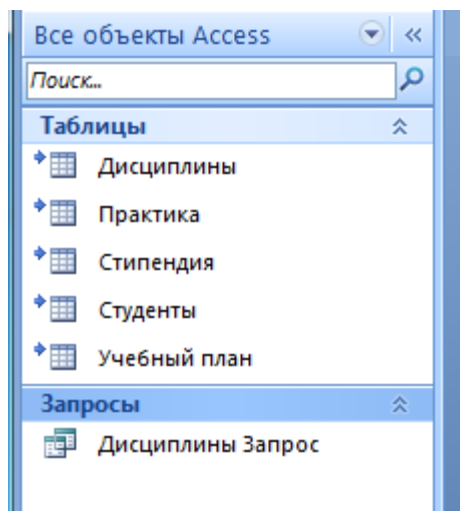


Рис. 11.2. Режимы открытия базы данных

Режимом открытия базы данных Access можно управлять тремя способами.

1) Можно включить в командную строку запуска Access, после имени базы данных, параметр `/Excl` или `/Ro`, чтобы база данных соответственно была открыта монопольно или только для чтения.

2) В диалоговом окне открытия базы данных (команда Открыть меню Файл) можно выбрать опцию монопольного открытия, открытия только для чтения или открытия монопольно и только для чтения.

3) Имеется возможность изменить режим открытия базы данных по умолчанию, выбрав из меню Сервис команду Параметры и изменив установку Режим открытия по умолчанию на странице Другие диалогового окна Параметры. Здесь можно задать либо общий доступ, либо монопольный.

### Период обновления

Период обновления по умолчанию каждые 60 секунд Access автоматически проверяет, изменились ли данные в открытых формах и таблицах. Для некоторых баз данных этот стандартный период может быть слишком большим. Однако, очень маленький интервал обновления повышает нагрузку на сеть. Оптимальный период обновления часто определяют экспериментальным путем. В общем случае, чем меньше сеть, тем меньший интервал обновления можно устанавливать. Обновить связанные таблицы можно вручную (рис.11.3).

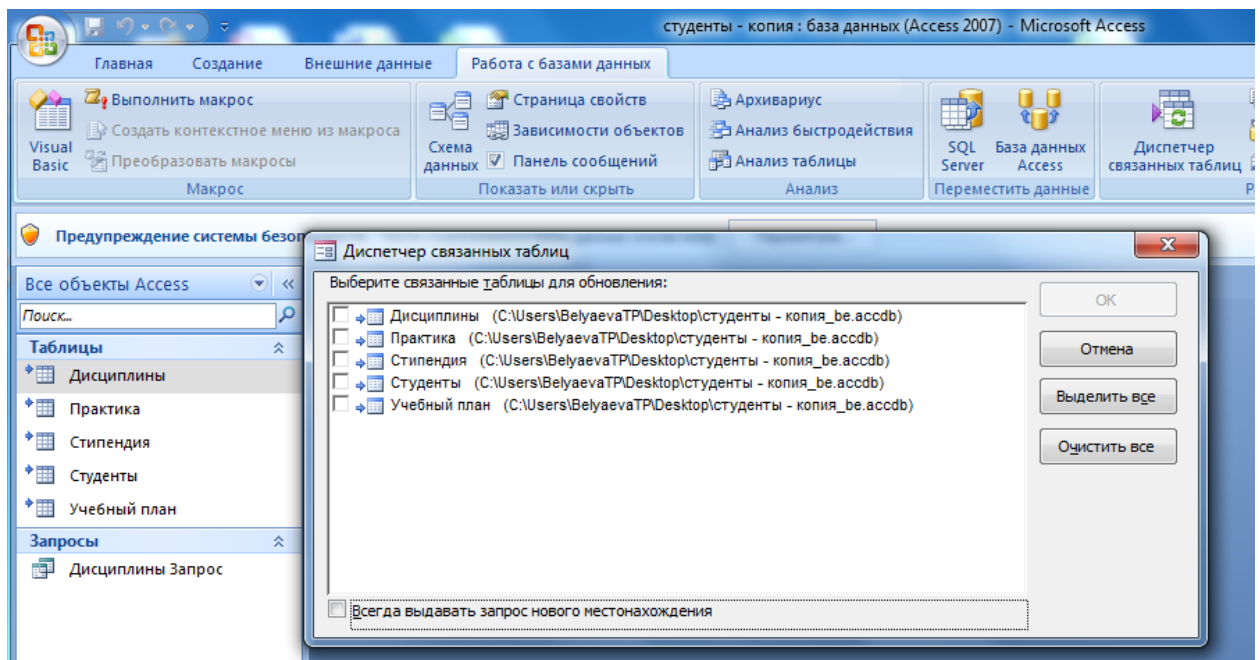


Рис. 11.2. Обновление связанных таблиц

### Блокировка

Для обеспечения параллельной работы с данными нескольких пользователей в Access применяется механизм блокировок. Во время редактирования пользователем некоторой записи, она становится заблокированной для других пользователей, пытающихся ее изменить. Блокировать можно записи по отдельности, а также целыми страницами.

Размер страницы составляет 2 Кбайт. Страница обычно включает в себя несколько соседних записей.

При одновременной работе с данными нескольких пользователей гораздо удобнее блокировать на уровне записей. Однако обновление большого количества записей быстрее выполняется при страничной блокировке.

Чтобы установить по умолчанию для текущей базы данных блокировку на уровне записей, установите флажок Блокировка записей при открытии БД (меню сервис / параметры, вкладка другие). Чтобы установить блокировку на уровне страниц снимите флажок.

Access поддерживает три режима блокировки:

1) Блокировка отсутствует. Этот режим часто называют оптимистической блокировкой. Он используется в Access по умолчанию. Запись (или содержащая ее страница в случае страничной блокировки) блокируется только на время ее сохранения, в процессе редактирования она остается незаблокированной.

2) Блокировка изменяемой записи. Как только пользователь начинает редактировать запись, она (или содержащая ее страница) блокируется до тех пор, пока изменение не будет сохранено. Этот режим известен как пессимистическая блокировка.

3) Блокировка всех записей. Блокируется весь набор записей. Полезна при обслуживании таблиц системным администратором.

#### Оптимистическая блокировка

Оптимистическая блокировка облегчает одновременный доступ к данным нескольких пользователей, снижая количество конфликтов блокировки, но повышает риск конфликтов записи. Конфликт записи возможен, когда:

1. пользователь начинает редактировать запись;
2. второй пользователь сохраняет изменения этой записи;
3. первый пользователь пытается сохранить свои изменения.

Получается, что первый пользователь редактирует совсем не ту запись, которую видит на экране.

#### Пессимистическая блокировка

При пессимистической блокировке два пользователя не могут одновременно редактировать одну и ту же запись. А если при этом установлен режим блокировки страниц, ограничение становится еще более

жестким: два пользователя не могут одновременно редактировать записи одной и той же страницы, даже если это разные записи.

Как правило, бывает нежелательным, чтобы пользователи одновременно редактировали одну и ту же запись. Поэтому, если используется блокировка на уровне записей, рекомендуется применять пессимистическую блокировку. Однако в некоторых ситуациях предпочтение следует отдавать оптимистической блокировке. Например, если некоторые пользователи подолгу блокируют записи или если используется блокировка на уровне страниц.

Режим блокировки можно настраивать отдельно для каждого объекта базы данных, манипулирующего наборами записей.

### Блокировка и формы

Главным недостатком оптимистической блокировки является возможность конфликтов записи. Когда такой конфликт происходит в связанной форме, Access выводит на экран диалоговое окно Конфликт записи. В этом окне представлены три опции.

1. Сохранить запись. Если пользователь выберет эту опцию, вносимые им изменения заменяют изменения, сделанные другим пользователем. Данную опцию применяют редко.

2. Копировать в буфер. Эта опция копирует производимые пользователем изменения в буфер обмена и обновляет текущую запись, чтобы в ней отразились изменения, сделанные другим пользователем.

3. Отменить изменения. При выборе этой опции все произведенные пользователем изменения будут отменены, а запись отразит изменения, сделанные другим пользователем.

В формах с пессимистической блокировкой конфликтов записи не бывает, поскольку два пользователя не могут одновременно редактировать одну и ту же запись. Пиктограмма с перечеркнутым кружком, которая заменяет селектор записи, сообщает другим пользователям, что запись заблокирована и редактировать ее пока нельзя.

### Транзакции

**Транзакция** – это последовательность операторов обработки данных, которая рассматривается как логически неделимая единица работы с базой данных.

Транзакция обладает следующими свойствами:



1. *Логическая неделимость* (атомарность) означает, что выполняются либо все операции, входящие в транзакцию, либо ни одной. (Логическая неделимость не подразумевает физической неделимости).

Система гарантирует невозможность фиксации части изменений, произведённых транзакцией. До тех пор, пока транзакция не зафиксирована, её можно "откатить", т.е. отменить все сделанные операторами из транзакции изменения в БД. Успешное выполнение транзакции означает, что все операторы транзакции проанализированы, интерпретированы как правильные и безошибочно исполнены.

2. *Согласованность*: транзакция начинается на согласованном множестве данных и после её завершения множество данных также согласовано.

3. *Изолированность*, т.е. отсутствие влияния транзакций друг на друга. (На самом деле это влияние существует и регламентируется стандартом: см. раздел 7.2. "Взаимовлияние транзакций").

4. *Продолжительность*: результаты зафиксированной транзакции не могут быть потеряны. Возврат БД в предыдущее состояние может быть достигнут только путём запуска компенсирующей транзакции.

Для управления транзакциями в системах, поддерживающих механизм транзакций и язык SQL, используются следующие операторы:

- фиксация транзакции: COMMIT [WORK];
- откат транзакции: ROLLBACK [WORK];
- точка сохранения: SAVEPOINT <имя\_точки\_сохранения>;

(Ключевое слово WORK необязательно). Предложение SAVEPOINT запоминает промежуточную "текущую копию" состояния базы данных для того, чтобы впоследствии, при необходимости, можно было вернуться к состоянию БД в точке сохранения: откатить работу от текущего момента до точки сохранения (rollback to <имя\_точки>) или зафиксировать работу от начала транзакции до точки сохранения (commit to <имя\_точки>).

**Начало транзакции** соответствует появлению первого исполняемого SQL-оператора. Транзакция завершается при наступлении одного из следующих событий:

- Поступила команда COMMIT или ROLLBACK (результаты транзакции соответственно зафиксируются или откатываются).

- Выдана и успешно проанализирована одна из команд языка описания данных (DDL, Data Definition Language), таких как CREATE, DROP или ALTER. При этом фиксируется предыдущая транзакция.
- Завершилась команда DDL. Таким образом, транзакция, содержащая оператор языка описания данных фиксируется автоматически.
- Пользователь завершил сеанс работы с системой (последняя транзакция фиксируется автоматически).
- Процесс пользователя аварийно завершен (последняя транзакция автоматически откатывается).

**Фиксация транзакции** заключается в следующем:

1. Изменения, внесённые транзакцией, делаются постоянными.
2. Уничтожаются все точки сохранения для данной транзакции.
3. Завершается транзакция (уничтожаются системные записи о транзакции в оперативной памяти).
4. Если выполнение транзакций осуществляется с помощью блокировок, то освобождаются объекты, заблокированные транзакцией.

Для организации отката СУБД во время выполнения транзакции производит запись в сегменты отката всех внесённых изменений. Все изменения выполняются в оперативной памяти (ОП), затем фиксируются в журнале транзакций и периодически (при выполнении контрольной точки) переписываются на диск. Процесс формирования контрольной точки заключается в синхронизации данных, находящихся на диске (т.е. во вторичной памяти) с теми данными, которые находятся в ОП: все модифицированные данные из ОП переписываются во вторичную память.

**Рассмотрим пример.** Пусть отношение Товары включает атрибут Общий объем, представляющий собой общий объем поставок для каждого товара. Значение Общий объем для любого определенного товара предполагается равным сумме всех значений атрибута объем отношения Поставки для всех поставок данного товара. В случае поставки необходимо выполнить две операции: добавить запись в таблицу Поставки и обновить поле Общий объем в таблице Товары.

Под транзакцией можно понимать преобразование одного согласованного состояния базы данных в другое, причем в промежуточных точках база данных находится в несогласованном состоянии. Не допустимо, чтобы одна из операций была выполнена, а другая нет, так как база данных остается в несогласованном состоянии (поставка зафиксирована, а общий

объем не увеличился). В идеальном случае должны быть выполнены обе операции. Однако между двумя операциями может возникнуть ошибка, например поле Общий объем окажется заблокированным другим пользователем. Если использовать транзакцию, то система гарантирует, что все операции будут отменены.

Для организации транзакций в Access можно использовать средства VBA, а именно библиотеку объектов доступа к данным ADO.

В ADO обработку транзакций обеспечивают три метода объекта Connection:

- BeginTrans - отмечает начало последовательности операций, которые должны быть объединены в одну транзакцию;
- CommitTrans – записывает результаты всех этих операций на диск;
- RollbackTrans – выполняет откат транзакции, т.е. отмену всех изменений, выполненных после последнего вызова BeginTrans.

Примерная базовая схема ADO-кода с обработкой транзакций приведена ниже:

## Function Fun1

## OnError GoTo FunErr

Dim cnn As ADODB.Connection

Dim cmd As ADODB.Command

## Dim fInTrnas As Boolean

Set `cnn = CurrentProject.Connection`

```
Set cmd = New ADODB.Command
```

## Dim a1 As Boolean

## Dim a2 As Boolean

```
a1= False
```

```
a2= False
```

```
cmd.ActiveConnection = cnn
```

```
cmd.CommandText = "Запрос1"
```

## cnn.BeginTrans

fInTrans = True

cmd.Execute

'последовательность изменений данных.

a1= True

cmd.CommandText = "Запрос2"

```

cmd.Execute
a2= True
cnn.CommitTrans
fInTrans = False
FunExit:
Set cmd = Nothing
cnn.Close
Set cnn = Nothing
Exit Function
FunErr :           ‘ при выполнении функции возникла ошибка
MsgBox “Error”
If fInTrans Then   ‘ если ошибка возникла внутри транзакции
Cnn.RollbackTrans ‘ производим откат транзакции
MsgBox “Rollback”
End If
If a1 Then MsgBox “a1= True” Else MsgBox “a1= False”
If a2 Then MsgBox “a2= True” Else MsgBox “a2= False”
Resume FunExit
End Function

```

Если в ходе выполнения транзакции ядро базы данных обнаруживает, что нужная запись заблокирована, происходит стандартная перехватываемая ошибка. В этом случае нужно либо дождаться освобождения записи, либо отменить транзакцию.

Применение транзакции, гарантирует целостность вносимых пользователем изменений данных, однако в ущерб производительности одновременной работы нескольких пользователей. Это связано с тем, что внутри транзакции ядро базы данных накапливает блокировки записей до тех пор, пока вся транзакция не будет сохранена. Поэтому, если внутри транзакции обрабатывается большой объем данных, каждым из пользователей подолгу будет блокироваться большое количество записей.

Выполняя запрос, не возвращающий данных, Access может обрабатывать его как одну транзакцию. Этим процессом можно управлять как с помощью объектов ADO, так и средствами пользовательского интерфейса.

В окне свойств сохраненного запроса на выполнение свойству UseTransaction можно присвоить значение No, чтобы Access не заключала запрос в транзакцию. Это повысит скорость обновления большого количества записей, но в случае ошибки отменить изменения будет невозможно. По умолчанию данное свойство имеет значение Yes, т.е. Access заключает любой запрос на выполнение в транзакцию.

### **3. Лабораторные задания.**

#### ***Упражнение 1.***

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию:

1. Разделите базу данных на данные и приложение. Создайте еще один файл объектов приложения и присоедините его к данным.
2. Откройте одно приложение базы данных в режиме монопольного доступа. Попробуйте открыть второе приложение базы данных.
3. Откройте обе базы данных приложения в режиме общего доступа.
4. Откройте одну и ту же форму в приложениях и протестируйте работу с ней в режимах оптимистической и пессимистической блокировки: в одном приложении для какой-либо записи установите режим редактирования, а в другом попытайтесь изменить ту же запись.
5. Для форм приложения по своему усмотрению выберите пессимистическую или оптимистическую блокировку записей.
6. Создайте два запроса на изменение, которые необходимо объединить в одну транзакцию.
7. Создайте программный модуль, а в нем - функцию, реализующую данную транзакцию.
8. Протестируйте работу транзакции: запустите два приложения; в одном приложении откройте форму или таблицы с записями, которые обрабатывает ваша транзакция, в другом приложении выполните функцию, реализующую транзакцию. Далее, в одном приложении - для записи, обрабатываемой транзакцией установите режим редактирования. В другом приложении запустите функцию с транзакцией в режиме отладки. Проверьте, возникает или нет ошибка доступа к записи, и обрабатывает ли ваша функция эту ошибку. Выполните эти действия для оптимистической и пессимистической блокировки в форме.

#### **4. Контрольные вопросы**

1. В чем преимущество разделения базы данных по сравнению с публикацией базы данных в общей папке?
2. В каких случаях следует отдавать предпочтение оптимистической блокировке?
3. Для чего предназначена созданная вами транзакция?

### **ЛАБОРАТОРНАЯ РАБОТА № 12**

#### **Защита приложений по управлению данными на уровне рабочих групп**

**1. Цель работы:** Получить практические навыки работы с технологией защиты приложений по управлению данными на уровне рабочих групп.

#### **2. Теоретический материал для домашнего изучения.**

Наиболее распространенным способом защиты базы данных является использование модели защиты на основе рабочих групп, называемой также защитой на уровне пользователей. В системе защиты Access каждый пользователь имеет собственное имя и пароль. Пароль подтверждает, что пользователь, который ввел имя, не выдает себя за другого. Пароль определяется самим пользователем, который может его изменить в любое время, - на других пользователях это никак не отразится.

Разрешения и пароль назначаются каждому пользователю, причем разрешения определяют права доступа к каждому из объектов базы данных. Для удобства администрирования пользователи могут объединяться в группы с общими разрешениями.

Система защиты Access состоит из двух частей.

- Учетные записи пользователей и групп и их пароли хранятся в файле рабочей группы. Этот файл в многопользовательской среде обычно содержится на файл-сервере и по умолчанию имеет имя SYSTEM.MDW.

- Разрешения на доступ к объектам хранятся в каждой базе данных.

Защита Access активна всегда, ее нельзя отключить. Причем система защиты остается невидимой до тех пор, пока вы не захотите ею воспользоваться. В системе всегда присутствует несколько стандартных учетных записей пользователей и групп. Каждый файл рабочей группы по умолчанию содержит две учетные записи групп (Admins и Users) и одну учетную запись пользователя (Admin).

Когда пользователь начинает сеанс работы с Access, предполагается, что он входит в систему с именем Admin и пустым паролем (строка нулевой длины). Если оказывается, что у входящего в систему пользователя Admin непустой пароль, запрашивается его имя и пароль. Таким образом, пока вы не измените пароль пользователя Admin, система защиты будет оставаться невидимой.

### Рабочие группы

Система защиты Access строится на основе рабочих групп. Рабочая группа определяется как группа пользователей, учетные записи которых хранятся в одном файле рабочей группы.

Файл рабочей группы – это особая зашифрованная база данных, по умолчанию имеющая имя SYSTEM.MDW, которую Access использует для хранения информации о пользователях:

- Имена пользователей и их личные идентификаторы.
- Данные групп и их личные идентификаторы.
- Данные о принадлежности пользователей к группам.

В многопользовательской среде файл рабочей группы обычно хранится на сервере, но можно и на каждой рабочей станции разместить его копии. Централизованное хранение данного файла, как правило, облегчает администрирование пользователей и групп. С другой стороны, если установки системы защиты достаточно статичны, размещение копий файла рабочей группы прямо на рабочих станциях снижает сетевой трафик.

### Создание новой рабочей группы

Microsoft включает в Access версия ниже 2007 утилиту под названием Администратор рабочих групп, которая предназначена для создания рабочей группы (файла рабочей группы), а также в выборе рабочей группы, используемой по умолчанию. Для запуска этой программы из Access выберите команду Сервис → Защита → Администратор рабочих групп. С помощью диалогового окна Сведения о владельце рабочей группы, которое открывается щелчком на кнопке Создать, вы можете создать новую рабочую группу.

В диалоговом окне Сведения о владельце рабочей группы нужно ввести значения в поля Имя, Организация, Код группы. На основе информации, введенной в это диалоговое окно генерируется зашифрованный двоичный идентификатор, называемый Workgroup SID. Он используется его для идентификации учетной записи группы Admins в рабочей группе. С

помощью утилиты Администратор рабочих групп можно изменить рабочую группу, используемую по умолчанию. Такой прием дает возможность пользователям быть членами нескольких рабочих групп, при том, что активной всегда будет только одна из них.

Чтобы использовать служебную программу администратора рабочих групп в Access 2007, воспользуйтесь одним из следующих методов.

*Способ 1:* использование кода Visual Basic.

С помощью кода Visual Basic, воспользуйтесь одним из следующих способов.

Запустите код Visual Basic в окне интерпретации:

1. В Access 2007 откройте надежную базу данных или включите макросы в существующей базе данных.

2. Нажмите клавиши CTRL + G, чтобы открыть окно **Интерпретация**.

3. Введите следующую строку кода и нажмите клавишу ВВОД.

DoCmd.RunCommand acCmdWorkgroupAdministrator

- Создайте модуль, содержащий код Visual Basic:

1. В Access 2007 откройте надежную базу данных или включите макросы в существующей базе данных.

2. На вкладке **Создать** в группе выберите **макросы** затем выберите **модуль**.

Создание подпрограммы и вставьте пример кода Visual Basic в подпрограмму. DoCmd.RunCommand acCmdWorkgroupAdministrator

3. Нажмите клавишу F5 для запуска кода.

*Способ 2:* Используйте макрокоманде ВыполнитьКоманду

1. В Access 2007 откройте надежную базу данных или включите макросы в существующей базе данных.

2. На вкладке **Создать** в группе **Другое** выберите **Макрос** и щелкните **Макрос**.

3. На вкладке **Конструктор** в группе **Показать или скрыть** нажмите кнопку **Показать все действия**.

4. На вкладке **Макрокоманда1** щелкните **RunCommand** в столбце **Действие** и выберите **WorkgroupAdminstrator** в списке **Команда**.

5. Нажмите кнопку **Сохранить**.

6. В группе «Сервис» выберите команду **выполнить**.

Учетные записи пользователей и групп



Access использует учетные записи пользователей и групп для управления разрешениями защиты. Эти два типа учетных записей разделяют общее пространство имен, и необходимо следить за тем, чтобы имена всех пользователей и групп были уникальными в файле рабочей группы.

Диалоговое окно для создания и настройки учетных записей пользователей и групп открывается в Access с помощью команды Сервис>Защита>Пользователи и группы. Добавлять, удалять и изменять учетные записи пользователей и групп могут только члены встроенной группы Admins, но любой другой пользователь может увидеть свою учетную запись и изменить свой пароль.

### Идентификаторы PID и SID, пароли

Создавая новую учетную запись пользователя или группы, вы должны ввести непустой личный идентификатор (Personal IDentifier, PID) длиной от 4 до 20 символов (с учетом регистра). Он объединяется его с именем учетной записи и создается для нового пользователя или группы идентификатор защиты (Security IDentifier, SID). По значению этого идентификатора система безопасности определяет права данного пользователя на доступ к объектам базы данных. Информация о правах пользователя хранится в базе данных. Увидеть личный идентификатор можно лишь однажды, при вводе.

Встроенные учетные записи пользователя Admin и группы Users имеют один и тот же SID для всех рабочих групп. Поэтому для обеспечения защиты вашей базы данных необходимо исключить пользователя Admin из группы Admins и лишить группу Users всех разрешений на доступ к объектам. Иначе, подключившись к вашей базе данных с использованием другого файла рабочей группы, можно получить неограниченные права на объекты вашей базы данных.

Ни одна из встроенных учетных записей не может быть удалена из рабочей группы.

Учетные записи должны создаваться только администратором базы данных. Он должен записывать и хранить вводимые личные идентификаторы. Это будет полезно в том случае, если кто-нибудь удалит учетную запись, которую затем потребуется воссоздать. Администратор также должен записывать и хранить имена пользователей, организаций и идентификаторы рабочих групп, введенные в диалоговом окне Сведения о владельце рабочей группы.

После создания новой учетной записи пользователя можно ввести на

вкладке Изменение пароля пароль длиной до 14 символов, учитывая при этом регистр символов. Пароли применяются для идентификации пользователей только при загрузке.

Изменить пароль может только его владелец, но члены группы Admins могут удалять пароли других пользователей. Пароли и личные идентификаторы (PID) хранятся в файле рабочей группы в зашифрованном виде.

### Группы

Учетная запись группы содержит не просто список пользователей. Во многих случаях учетные записи групп можно применять вместо учетных записей пользователей. Сравнительные характеристики этих двух типов учетных записей приведены в таблице 1.

Пользователь Admin по умолчанию включен и в группу Users и в группу Admins. Любой добавляемый в рабочую группу пользователь автоматически включается в группу Users. Эта группа представляет собой всех пользователей Access. Никакой пользователь не может быть удален из группы Users.

Группа Admins представляет собой группу администраторов всех баз данных, используемых рабочей группой. Этой группе автоматически предоставляются полные права на доступ к этим базам данных. В каждый момент времени группа Admins должна содержать по крайней мере, одну учетную запись пользователя.

Таблица 1

#### Характеристики пользователей и групп

Характеристика	Пользователь	Группа
Имеет набор разрешений на доступ к объектам	✓	✓
Имеет личный идентификатор	✓	✓
Может быть владельцем объектов	✓	✓
Может входить в систему	✓	
Имеет пароль	✓	
Может быть владельцем базы данных	✓	

#### Предоставление разрешений

Для предоставления разрешений на объекты базы данных посредством пользовательского интерфейса Access нужно выбрать команду Сервис->Защита->Разрешения, в результате чего будет открыто диалоговое окно.

В системе безопасности Access пользователи получают как явные, так и неявные разрешения. Явные разрешения – это те, которые вы явно предоставляете пользователю и которые непосредственно связаны с его учетной записью. Неявные разрешения пользователь получает автоматически, благодаря своему членству в группе. Итоговый набор имеющихся у пользователя разрешений на объекты является комбинацией его явных и неявных разрешений. Пользователь всегда получает максимально высокие разрешения из перечней своих явных разрешений и разрешений групп, в которые он входит.

#### Административные права доступа

Административные права предоставляют полный доступ к объектам и/или базе данных, а также разрешение на изменение прав доступа других пользователей и групп. Административными правами обладают:

- 1) все пользователи группы Admin;
- 2) владельцы объектов;
- 3) пользователи, которым явно предоставлены административные права доступа.

Если в определенный момент пользователь не может выполнить какую-либо операцию из-за отсутствия прав доступа, он может иметь разрешение назначить себе эти права. Такими пользователями являются пользователи группы Admins рабочей группы базы данных и владельцы объектов.

#### Право на владение объектами базы данных

Владельцем называется учетная запись пользователя, имеющего контроль над базой данных или ее объектом.

Владельцем базы данных является создавший ее пользователь. Этот пользователь всегда может открыть базу данных и имеет на нее особые и неотъемлемые права. Если ему не предоставлены определенные права доступа, он может их вернуть, изменив права доступа к объекту или базе данных для себя и других пользователей.

#### Удаление учетной записи

При удалении из рабочей группы учетной записи пользователя или группы все ее разрешения остаются в базе данных. Пользователь, восстановивший учетную запись путем ввода ее исходного PID получает

доступ к защищенной базе данных. Поэтому перед удалением учетной записи очень важно удалить все ее разрешения и передать все ее права владения объектами другому пользователю или группе.

Если кто-либо восстановит учетную запись с тем же именем, но с другим PID, система защиты будет рассматривать ее как совершенно иную запись. Эта новая запись не унаследует никаких разрешений исходной – идентификатором записи является SID, а не имя пользователя или группы, а эти идентификаторы у двух записей будет разными.

### Шифрование баз данных Jet

Не смотря на защиту, файл базы данных Jet может быть открыт с помощью низкоуровневого дискового редактора. Поэтому защита файлов баз данных предполагает еще их шифрование.

Зашифровать и дешифровать базу данных могут только ее владелец и члены группы Admins. Для шифрования и дешифрования базы данных предназначена команда Сервис → Защита → Шифровать/ Расшифровать...

У шифрования базы данных имеется два негативных побочных эффекта. Во-первых, снижается ее быстродействие на 10-15%. Во-вторых, при сжатии зашифрованной базы данных такими программами, как PKzip, LHA, Stacker и DriveSpace размер ее уменьшается незначительно.

## 3. Лабораторные задания.

### *Упражнение 1.*

Для созданных ранее баз данных (в СУБД Microsoft Access и СУБД SQL Server) согласно индивидуальному заданию выполнить:

1. Создайте новый файл рабочей группы.
2. Добавьте в файл рабочей группы учетные записи двух групп пользователей.
3. Для созданных групп установите различные права доступа на объекты вашей базы данных.
4. Добавьте в файл рабочей группы учетные записи нескольких пользователей и установите их членство в созданных группах.
5. Для некоторых пользователей установите дополнительные права доступа на объекты базы данных.
6. Удалите пользователя Admin из группы Admins.
7. Лишите группу Users всех разрешений на объекты базы данных.

8. Удалите учетную запись одного из пользователей, не удаляя ее разрешений.
9. Восстановите учетную запись.
10. Зашифруйте базу данных.

#### **4. Контрольные вопросы**

1. Как Access (SQL Server) определяет права входящего в систему пользователя?
2. Можно ли получить доступ к объектам базы данных Access (SQL Server), используя разные файлы рабочих групп?
3. Если у группы и входящего в нее пользователя разные разрешения на один и тот же объект, какие права получает пользователь?
4. Может ли пользователь, не являющийся администратором, создать или удалить объект базы данных?
5. Можно ли восстановить удаленную учетную запись пользователя?

### **ЛАБОРАТОРНАЯ РАБОТА № 13**

#### **Репликация баз данных**

**1. Цель работы:** получить практические навыки репликации баз данных.

**2. Теоретический материал для домашнего изучения.**

Репликация в Access – это технология создания и поддержки особого вида копий (реплик) обычных баз данных MDB, позволяющая быстро переносить изменения одной копии во все остальные.

Основным преимуществом репликации по сравнению с архитектурами клиент/сервер и файл/сервер является то, что связь между рабочими станциями, на которых установлены отдельные реплики, требуется только на время синхронизации.

Репликацию базы данных рекомендуется использовать если:

- данные обновляются нечасто;
- обновления обычно не отражаются на работе других пользователей;
- сеть перегружена;
- сеть часто не работает.

К недостаткам репликации относятся следующие:

- 1) реплицированная база данных более громоздка;
- 2) несколько пользователей могут одновременно модифицировать одну и ту же запись;
- 3) разные пользователи в одно и то же время могут видеть разное состояние одной и той же записи;
- 4) значения полей типа счетчик генерируются случайным образом.

В рамках данной технологии определяются три процесса:

- 1) репликация;
- 2) синхронизация;
- 3) устранение конфликтов.

### Репликация

В ходе конвертирования обычной базы данных в реплицируемую Jet вносит в ее структуру целый ряд изменений. Эти изменения заключаются в добавлении новых системных таблиц, модификации пользовательских таблиц данных, создании новых свойств самой базы данных и многих ее объектов. Конвертировав обычную базу данных в реплицируемую, получаем основную реплику нового набора реплик. Основную реплику сразу после создания реплицируют – создают как минимум еще одну. Последующие реплики могут создаваться путем репликации любой из реплик набора.

Данные базы данных можно модифицировать в любой реплике. Структуру базы данных можно изменять только в основной реплике.

Все запросы и таблицы, сохраненные в какой-либо реплике, не являющейся основной, становятся локальными. Единственный способ сделать такой объект реплицируемым заключается в том, чтобы импортировать его в основную реплику, удалить из той, где он был создан, а затем в основной реплике превратить объект в реплицируемый.

В рамках технологии репликации все объекты, кроме таблиц и запросов, считаются одним составным объектом. Можно реплицировать все (формы, отчеты, макросы, модули, страницы доступа к данным), либо не реплицировать ни один из них. Реплицированные вместе с базой данных объекты Access можно модифицировать только в основной реплике. Новые объекты также можно создавать лишь в ней.

По умолчанию Access реплицирует все таблицы и запросы базы данных. Эти объекты можно реплицировать выборочно. С помощью

пользовательского интерфейса – в окне Свойства объекта нужно снять или установить флажок Реплицируемый.

При использовании JRO (объектная модель, позволяющая создавать, модифицировать и синхронизировать реплики Jet) для установки статуса репликации таблицы или запроса используется метод `SetObjectReplicability` объекта `Replica`.

### Синхронизация

Когда обновляется одна из реплик, Jet фиксирует изменения в дополнительных таблицах, добавленных в ходе репликации базы данных. Однако без специального указания эти изменения не отсылаются другим членам набора реплик. Между репликами постоянного соединения нет. Они соединяются только на время синхронизационного обмена – процедуры, в которой всегда участвуют только две реплики.

Когда вы синхронизируете две реплики, Jet передает одной из них изменения, внесенные в другую реплику. Реплики обмениваются только изменениями. Этот процесс полностью управляем: вы сами определяете, когда должна происходить синхронизация и между какими репликами.

При использовании JRO можно выполнить однонаправленный обмен данными.

Для того чтобы инициировать синхронизацию программным путем с применением JRO нужно использовать метод `Synchronize` объекта `Replica`. Он имеет следующий синтаксис:

`Replica.Synchronize` путь [, тип]

`Replica` - это объектная переменная, указывающая на существующий объект `Replica`; путь – полный путь к реплике, с которой будет выполняться синхронизация; тип - константа, определяющая тип синхронизации. Возможные значения : `jrSyncTypeExport` (только отправить изменения), `jrSyncTypeImport` (только получить изменения), `jrSyncTypeImpExp` (отправить и получить изменения).

### Устранение конфликтов

Конфликтом называется ситуация, когда в одну и ту же строку синхронизируемых реплик вносятся противоречащие друг другу изменения. По завершении синхронизации Jet уведомляет вас обо всех обнаруженных конфликтах.

Конфликты могут устанавливаться как на уровне строк, так и на уровне столбцов.

Если задано устранение конфликтов на уровне столбцов, то конфликт не будет зарегистрирован, в случае изменения в репликах разных столбцов одной и той же строки таблицы. Если задано устранение конфликтов на уровне строк, то в такой ситуации конфликт будет зарегистрирован. Jet позволяет указать способ устранения конфликтов для каждой из таблиц.

Каждой реплике назначается определенный приоритет, в соответствии с которым производится выбор, изменения какой реплики в случае конфликта сохранить, а какой нет.

#### Видимость реплик

По уровню видимости, поддерживаемые Jet 4.0 реплики делятся на три группы: глобальные, локальные и анонимные.

Видимость реплики определяет, будет ли она видна другим репликам и смогут ли они с ней синхронизироваться.

Глобальная реплика видна всем остальным глобальным репликам, и все они могут с ней синхронизироваться. На основе глобальных реплик можно создавать реплики любых типов: глобальные, локальные, анонимные. По умолчанию реплика, созданная на основе глобальной, также является глобальной.

Локальная реплика может синхронизироваться только со своей родительской репликой (той, что ее создала). Локальную реплику видит только ее родительская реплика. Родительская реплика может инициировать синхронизационный обмен, и конфликты в любом случае устраняются в ее пользу. Приоритет локальной реплики всегда равен нулю. На основе локальной реплики можно создать только локальную реплику.

Анонимная реплика подобна локальной. Может синхронизироваться только со своей родительской репликой. Она также имеет нулевой приоритет. Информация об анонимной реплике, в отличие от локальной, хранится в таблице MSysReplica1 только временно и удаляется из нее после определенного периода бездействия. Анонимные реплики используются в тех случаях, когда общее количество реплик достаточно велико и большинство из них изредка участвует в синхронизационном обмене. После удаления информации из таблицы MSysReplicas анонимная реплика должна сама инициализировать следующую синхронизацию.

#### Частичные реплики

Частичной называется реплика, содержащая лишь некоторое подмножество данных из одной или нескольких реплицированных таблиц.



Если вы, к примеру, устанавливаете базу данных с информацией о продажах на портативных компьютерах представителей компании с ограниченным объемом дискового пространства, каждому из таких пользователей можно предоставить информацию только по его региону.

Выражение фильтра частичной реплики указывает записи в конкретной таблице, которые должны войти в частичную реплику. Мастер применит этот фильтр ко всем связанным таблицам в базе данных, чтобы обеспечить извлечение всех необходимых записей. Выражение для фильтра аналогично выражению SQL Where, с тем исключением, что оно не может содержать статистические функции, определяемые пользователем функции, а также подчиненные запросы.

#### **Пример:**

[КодСотрудника] = 1, [Фамилия]= 'Иванов' And [Имя]= 'Петр', [ДатаРазмещения] < #1/1/2000 Or [ДатаИсполнения] < #3/1/2000. Когда Access использует выражение фильтра при отборе записей для частичной реплики, выполняется запрос к реплицируемой базе данных. Для повышения быстродействия необходимо индексировать каждое поле реплики, используемое в выражении фильтра.

#### **Реплики с запрещенным удалением**

Создавая реплику с помощью команд меню Access, можно запретить удаление ее записей. Эта возможность полезна в тех случаях, когда нужно предоставить пользователю полную реплику, но вместе с тем и иметь гарантию того, что записи не будут удаляться.

#### **Управление репликацией посредством меню Access**

В меню Сервис → Репликация имеются команды, предоставляющие пользователю Access доступ к функциям репликации Jet.

Таблица 1.

Меню репликации Access

Пункт меню	Назначение
Синхронизация	Определяет реплику, с которой требуется выполнить синхронизацию; может также использоваться для передачи статуса основной реплики другой реплике набора
Создать реплику	Для нереплицированной базы данных создает основную и еще одну реплику; для реплицированной базы данных создает еще одну реплику
Мастер частичной репликации	Запускает мастер частичных реплик, который проведет вас через процесс создания и заполнения частичной реплики
Устранение конфликтов	Если в открытой реплике имеются конфликтующие записи, команда запускает утилиту, с помощью которой можно

	устранить все конфликты
Восстановить основную реплику	Если открытая реплика не является основной, можно превратить ее в таковую. Пользоваться следует, если основная реплика повреждена или разрушена

### Защита реплицированной базы данных

Access не обеспечивает возможность репликации файла рабочей группы (SYSTEM.MDW) в защищенной среде. Поэтому в локальной среде нужно либо подключить каждую реплику к файлу рабочей группы, либо скопировать файл рабочей группы на каждую рабочую станцию. В глобальной сети файл рабочей группы нужно скопировать таким образом, чтобы он стал достоянием для каждой удаленной реплики.

Репликация применяется не только при необходимости организовать многопользовательскую работу с данными, но и с целью резервного копирования. Реплицировав базу данных и регулярно синхронизируя ее с репликой, вы защищаете себя от потери данных на тот случай, если эта база данных будет разрушена. Также репликацию можно использовать с целью распространения обновлений приложения.

## 3. Лабораторные задания.

### Упражнение 1.

1. Для своей базы данных, созданной в предыдущих работах, создайте основную реплику.
2. Создайте набор из 3 реплик различных уровней видимости.
3. Создайте частичную реплику и реплику с защищенным удалением.
4. Протестируйте работу с репликами в многопользовательском режиме.
5. Защитите все реплики.

## 4. Контрольные вопросы

1. Можно ли локальную таблицу превратить в реплицируемую?
2. Как происходит обмен данными между репликами?
3. Каким образом учитывается приоритет реплик?
4. Чем отличается локальная реплика от анонимной?
5. В каких случаях используются частичные реплики?
6. Каким образом производится защита реплик?

**ЛАБОРАТОРНАЯ РАБОТА № 14****Публикация данных в корпоративной сети и Интернете**

**1. Цель работы:** Получить практические навыки публикации данных из баз данных.

**2. Теоретический материал для домашнего изучения.**

Страницы доступа к данным – это специальный тип Web-страниц, представляющий пользователям Web-интерфейс форм для доступа к данным из базы данных Access или SQL Server.

Основное преимущество такой организации совместного доступа к данным при сравнении с формами и отчетами состоит в том, что на рабочих станциях пользователей достаточно иметь только Internet Explorer версии 5 и выше.

Создавая в Access страницу доступа к данным, получаем два независимых элемента:

- 1) объект Access Страница доступа к данным, основная часть которого – адрес HTM-файла;
- 2) HTM-файл, содержащий весь исходный код страницы на языках HTML и XML

Создать страницу доступа к данным можно тремя способами:

- 1) с помощью Мастера;
- 2) преобразованием формы или отчета в страницу доступа к данным;
- 3) с помощью Конструктора.

Процесс создания страницы с помощью Мастера или Конструктора аналогичен работе с формами и отчетами в этом режиме. Чтобы выполнить преобразование формы или отчета в страницу доступа к данным необходимо выделить исходный объект, выполнить команду Сохранить как, выбрав в списке форматов Страница доступа к данным (Data Access Page).

Страницы доступа к данным создаются с целью решения одной из следующих задач:

- 1) просмотр данных;
- 2) ввод и редактирование данных;
- 3) анализ данных.

### Создание редактируемой страницы

Для начала создадим страницу доступа к данным, которая позволяла бы редактировать данные, отбираемые запросом `qryCustomersWithFullName` (запрос отбирает все поля таблицы Клиенты и добавляет на страницу вычисляемое поле Полное имя). Итак, вам нужно выполнить следующие действия.

1. В окне базы данных откройте в списке объектов страницу Страницы, откройте пустую страницу в режиме конструктора.
2. Щелкните на надписи Название страницы и введите заголовок Информация о клиентах.
3. Если на экране нет окна Список полей, откройте его либо с помощью кнопки панели инструментов, либо с помощью команды меню Вид → Список полей.
4. В окне Список полей щелкните на узле Запросы, чтобы открыть соответствующую ветвь, и выделите запрос ЗапросКлиентыФИО. Выполните щелчок на значке «+» рядом с именем запроса, чтобы открыть список полей.
5. Перетащите поля код клиента, ФИО и телефон в область сетки на странице и выровняйте их таким образом, чтобы все вместе выглядело примерно так, как на рис. 14.1.

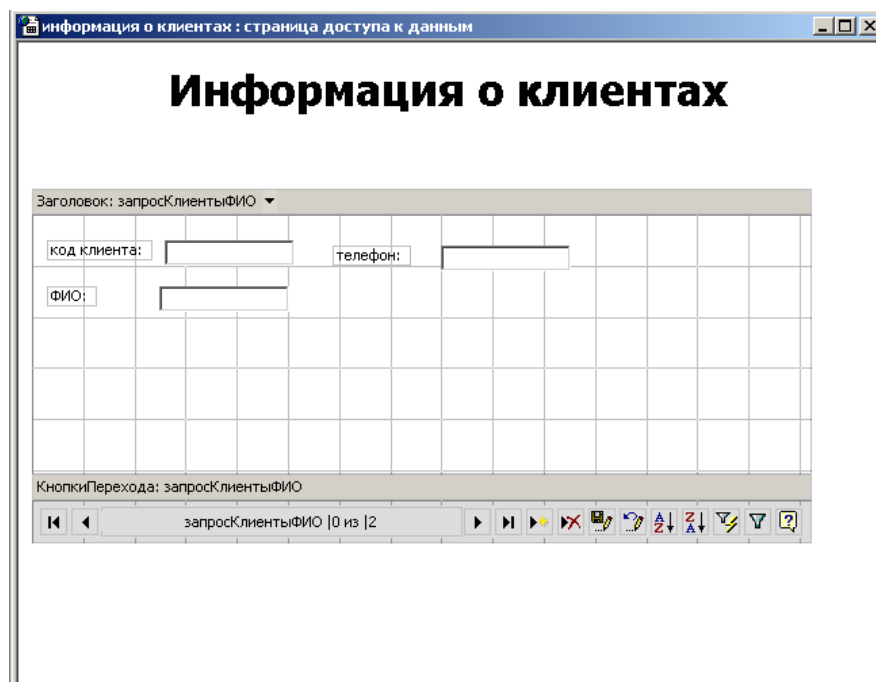


Рис. 14.1. Примерное размещение полей на странице

6. После этого имя раздела сетки изменится в соответствии с источником данных. Чтобы данные можно было редактировать, необходимо поместить на страницу поле Код клиента. Если на странице нет поля первичного ключа, данные нельзя редактировать. Кроме того, если вы поместите на страницу поле первичного ключа не первым, а после любого другого поля, то данные также невозможно будет редактировать. Чтобы исправить положение, нужно присвоить свойству UniqueTable раздела заголовка запросКлиентыФИО значение Клиенты. Естественно, поля Код клиента и ФИО доступны только для чтения; для редактирования поля Телефон необходимо, чтобы поле Код клиента находилось на странице доступа к данным.

7. Сохраните страницу в той же папке, где находится ваша база данных. (Если вы закрыли страницу, откройте ее снова в режиме конструктора.) Вы увидите сообщение, информирующее о том, что строка подключения для этой страницы представляет собой абсолютный путь. Чтобы закрыть окно сообщения, щелкните на кнопке ОК — вы всегда можете вернуться назад и внести изменения в информацию о подключении. Выберите команду Вид → Просмотр страницы и попробуйте поработать с полученной страницей. Вносить изменения в поле Телефон можно для любой записи. (Перемещаться по записям позволяют кнопки перехода, расположенные на навигационной панели.)

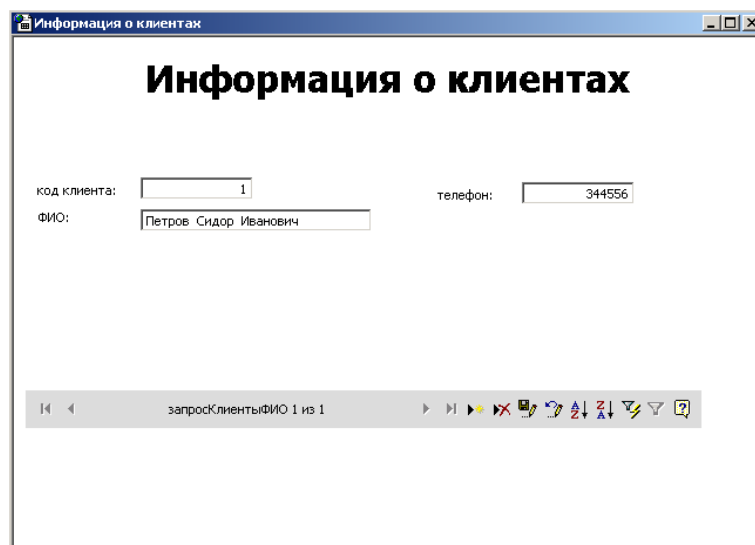


Рис. 14.2. Страница, на которой можно редактировать поля

Информация о клиентах - Microsoft Internet Explorer

Файл Правка Вид Избранное Сервис Справка

Назад Поиск Избранное Медиа

Адрес: C:\Documents and Settings\qwert\Мои документы\Информация о клиентах.htm Переход Ссылки

# Информация о клиентах

код клиента:

ФИО:

телефон:

запросКлиентыФИО 1 из 2

Готово Мой компьютер

9. Еще раз откройте страницу в режиме конструктора, а затем — окно свойств группировки для объекта Заголовок: запросКлиентыФИО (щелкнуть на стрелке в разделе заголовка группы (рис. 14.4) и выбрать в контекстном меню команду Свойства уровня группы). Найдите в открывшемся окне свойство `DataPageSize` и замените его значение 1 значением 5. В результате, если вы откроете страницу в режиме страницы, или в браузере, на ней будут отображаться одновременно пять строк.

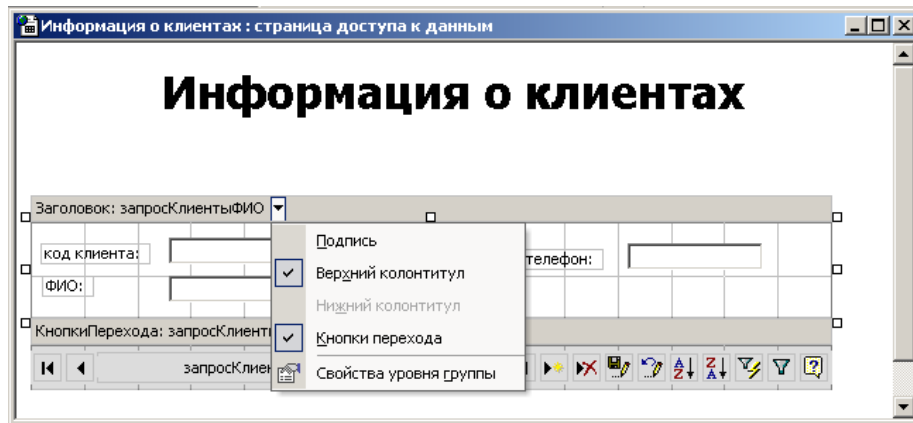


Рис. 14.4. Чтобы открыть окно свойств группирования, используйте контекстное меню заголовка

10. Перейдите в режим просмотра страницы и протестируйте любую из отображаемых пяти строк. Отредактируйте любой номер телефона, после чего переключитесь обратно в режим конструктора.

11. Уделите внимание панели кнопок перехода по записям. В частности, попробуйте скрыть кнопки, которые по вашему мнению не должны быть доступны пользователю (например, кнопки Создать и Удалить). Для этого в контекстном меню этой панели выберите команду Кнопки переходов. В появившемся подменю (рис. 11.5) можно снять или установить флажок кнопки, которую нужно скрыть или вывести на панель.

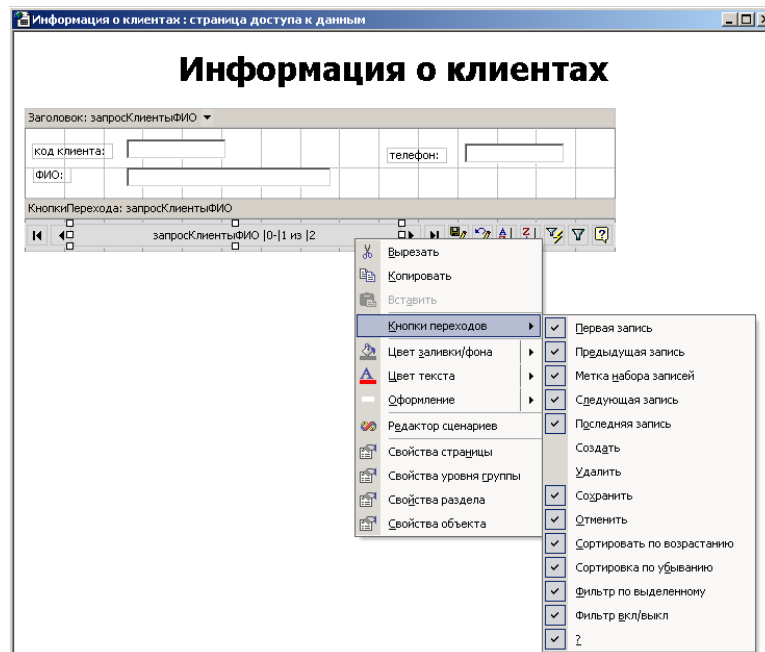


Рис. 14.5. Добавление и удаление кнопок переходов

12. Закончив работу, закройте и сохраните страницу.

С помощью страниц доступа к данным предоставляется возможность интерактивной работы с многоуровневыми данными.

При создании таких страниц используются связанные элементы управления и применяется группировка данных. Пользователь может разворачивать интересующие его группы, просматривать и редактировать входящие в них записи, перемещаясь по уровням иерархии и разыскивая нужную информацию.

### ***Анализ данных***

Страницы доступа к данным могут включать сводные списки, позволяющие реорганизовать данные для их анализа различными способами. Страница может содержать диаграммы для анализа тенденций, закономерностей и выполнения сравнений между данными в базе данных, а также электронные таблицы.

Страница доступа к данным непосредственно связана с базой данных. При просмотре в Internet Explorer отображается копия страницы. Поэтому любой отбор данных, сортировка и другие изменения способа отображения данных влияют только на копию страницы. Изменения в самих данных – изменение значений, добавление или удаление данных – сохраняются в базе данных и впоследствии доступны всем, кто просматривает страницу.

Страницы доступа представляют собой удобную технологию создания Web-страниц для публикации отчетов и ввода информации в корпоративных сетях. Страницы доступа к данным можно расположить также на Web-сервере, но при этом придется ограничиться только данными SQL Server. Это связано с тем, что все операции по обработке данных из MDB файлов должны выполняться на клиентской машине. В таком случае Jet должна быть установлена в локальной системе пользователя. Однако концепции Web-приложений больше соответствует архитектура на базе SQL Server, когда вся обработка выполняется на сервере.

### ***Публикация страниц в папках Web или на web-сервере.***

Опубликовать Web-страницу можно при помощи Проводника Windows. Настройка Web-папки позволяет работать с папками и файлами на Web-серверах так же, как с папками и файлами на локальном диске.

Для публикации страницы доступа к данным, необходимо поместить нужный файл HTML и каталог с сопутствующими файлами в Web-папку или в виртуальный каталог на web-сервере.

Для использования этой возможности необходимо:



1. запустить «Проводник» Windows;
2. выделить элемент Web-папки. Дважды щелкнуть по элементу Добавить Web-папку;
3. в появившемся диалоговом окне введите URL папки на Web-сервере, в которой нужно сохранить Web-страницу, например [http://srvGK415/ Page1](http://srvGK415/Page1), и нажмите кнопку Готово. В списке Web-папки появится новый элемент.
4. Скопируйте нужные файлы и папки в Web-папку.

Если Web-сервер расположен на вашем локальном компьютере, опубликовать на нем данные можно, скопировав необходимые файлы в корневую папку Web-сервера или одну из его вложенных папок. Корневая папка Web-сервера может иметь имя \Inetpub\wwwroot.

### ***Публикация данных через Интернет с помощью виртуальных частных сетей***

Обеспечивая защищенный обмен данными, технология виртуальных частных сетей (Virtual Private Networking, VPN) предоставляет работающим дома пользователям, филиалам компании, удаленным пользователям и другим компаниям возможность подключения к корпоративной сети через Интернет. Пользователи могут использовать проверку подлинности Windows, как если бы они находились в локальной сети. Репликации Microsoft SQL Server всех типов могут реплицировать данные через виртуальную частную сеть, но для репликации слиянием необходимо предусмотреть веб-синхронизацию, так как в этом случае применение виртуальной частной сети не требуется.

Виртуальная частная сеть включает клиентское программное обеспечение, так что компьютеры подключаются к программному обеспечению выделенного компьютера или сервера через Интернет (или, в особых случаях, даже через корпоративную сеть). При желании можно использовать как шифрование с обеих сторон, так и пользовательские методы проверки подлинности. Соединение с виртуальной частной сетью через Интернет логически функционирует как соединение глобальной сети (WAN) между сайтами.

Виртуальная частная сеть связывает компоненты сети с помощью другой сети. Чтобы подключиться, пользователь осуществляет связь через Интернет или иную общедоступную сеть, используя протоколы, такие как Microsoft PPTP (Point-to-Point Tunneling Protocol) или L2TP (Layer Two

Tunneling Protocol). Этот процесс предоставляет такую же безопасность и возможности, которые были доступны ранее только в частных сетях. Протокол PPTP доступен в Microsoft Windows NT версии 4.0 и Microsoft Windows 2000 (и более поздних) операционных систем. Протокол L2TP в Windows 2000 и более поздних версий.

Промежуточная инфраструктура маршрутизации Интернета остается невидимой для пользователя и ему кажется, будто данные посылаются через выделенную частную линию. С точки зрения пользователей виртуальная частная сеть представляет собой двухточечное соединение между компьютером пользователя и корпоративным сервером.

После того как у удаленного клиента выполнена настройка для подключения с использованием виртуальной частной сети и клиент, имеющий доступ в Интернет, зарегистрировался в корпоративной сети, можно настроить репликацию так, будто удаленный пользователь напрямую подключен к локальной сети. По соображениям безопасности можно предоставить доступ к разным сетевым ресурсам для пользователей, подключающихся через виртуальную частную сеть и для тех, кто подключается непосредственно к локальной сети.

### **3. Лабораторные задания.**

#### ***Упражнение 1.***

1. В своем приложении добавьте редактируемую страницу доступа к данным.
2. Протестируйте страницы в Access, а также в браузере.
3. Разместите страницу в папке Web.
4. Протестируйте страницу.

### **4. Контрольные вопросы**

1. В чем отличие страниц доступа к данным от форм и отчетов?
2. Какие изменения, сделанные пользователем на странице доступа к данным доступны другим пользователям, а какие нет?
3. В чем особенность публикации страницы доступа к данным на Web-сервере?

## ЛАБОРАТОРНАЯ РАБОТА № 15

### Приложения архитектуры «клиент-сервер»

**1. Цель работы:** изучить основополагающие принципы построения клиент-серверных приложений.

**2. Теоретический материал для домашнего изучения.**

Access является средством разработки клиент-серверных приложений баз данных. В таких приложениях вся работа по хранению данных и выполнению над ними всевозможных операций возложена на специальное программное обеспечение (сервер данных), обычно работающее на мощном сетевом сервере. Access выступает в роли клиента, обеспечивая интерфейс к серверным данным.

В Access реализована технология построения клиентских приложений – проекты Access (Access Data Project, ADP). В ней используется файл нового формата (ADP), который заменил базу данных Jet в части хранения объектов приложения (форм, отчетов, макросов и модулей). Хранение и обработка данных возложены на SQL Server, к которому подключается приложение.

Преимущества архитектуры клиент-сервер

Цели перехода на клиент-серверную архитектуру:

- 1) увеличение производительности;
- 2) повышение масштабируемости – возможности поддержки большого количества пользователей;
- 3) обеспечение большей надежности и защищенности.

Надежность и защита данных

Когда вы используете Access для работы с удаленным файлом базы данных, каждый экземпляр Jet управляет собственным кешем физических страниц из этого файла, страницы обновляются на каждой клиентской машине и отправляются обратно на сервер по сети. Каждая копия Jet выполняет обновление индексов, операции с системными таблицами и другие функции управления базой данных, необходимые для работы вашего приложения. Сбой любого клиентского компьютера может привести к разрушению всей базы данных.

В противоположность этому, когда Access используется в качестве интерфейса к данным, управляемым сервером базы данных, сервер сам выполняет все операции над данными. Только он один обновляет физический файл и видит физические страницы базы данных. Каждая копия Access

отправляет серверу базы данных запросы и получает от него данные или указатели на данные. Поскольку все операции по обработке данных выполняются ее сервером, сбои сети или клиентских компьютеров лишь в редких случаях могут повлиять на саму базу. Именно метод доступа к физическим данным и выполнения операций над ними в базе данных делает архитектуру клиент-сервер более надежной и защищенной по сравнению с архитектурой файл/сервер.

#### Производительность

Сравнительную производительность приложений файл/сервер и клиент/сервер часто оценивают некорректно. В общем случае, если у пользователей чаще всего возникает потребность извлекать небольшое количество данных из огромного количества записей, клиент-серверная система будет делать это быстрее. С другой стороны, если пользователь анализирует значительные объемы данных или формирует множество отчетов, несущественно отличающихся друг от друга, то в такой работе быстрее будет справляться система с файл-серверной архитектурой.

#### Создание проектов Access

Одним из способов создания проекта Access является конвертирование существующей базы данных Access в новый проект Access и базу данных SQL Server.

Откройте имеющуюся в вашем распоряжении базу данных Access и выберите из меню Сервис->Служебные программы команду мастер преобразования в формат SQL. Перед вами появится окно мастера. Здесь необходимо указать, следует мастеру создавать новую базу данных SQL Server или использовать уже существующую. Выберите первую. Так надежнее- вы получаете гарантию того, что мастер не повредит уже имеющиеся данные.

Второе окно мастера позволяет выбрать сервер, указать имя новой базы данных и ввести информацию для подключения.

Третье окно мастера служит для выбора таблиц, которые вы хотите переместить из существующей базы данных Access в новую базу данных SQL Server. Чтобы избежать потенциальных проблем с переносом в новое приложение других объектов, например запросов или форм, связанных с конкретными таблицами, имеет смысл переносить все таблицы.

В следующем окне мастера выбираются атрибуты таблиц, которые должны быть перенесены в новую базу данных. Переносить можно индексы, условия на значения, значения по умолчанию, межтабличные связи.

В следующем окне определите изменения, которые мастеру следует внести в ваше приложение.

Здесь следует выбрать Create a New Access Client/Server Application. Мастер создаст базу данных SQL Server и свяжет ее таблицы с новым проектом Access. Существующие объекты приложения Access будут перенесены из базы данных в новый проект. В поле ADP File Name введите имя нового проекта.

Сделав нужные установки, щелкните на кнопке Финиш. В результате мастер сделает следующее:

1. Создаст новую базу данных SQL Server.
2. Перенесет в эту базу данных таблицы с выбранными атрибутами
3. создаст новый проект Access и свяжет его с только что сформированной базой данных SQL Server.
4. Создаст в базе данных SQL Server представления и хранимые процедуры, соответствующие запросам, которые имелись в базе данных Access, и преобразует Jet SQL в Transact-SQL.
5. Скопирует формы и отчеты из базы данных в проект Access.
6. Создаст новые страницы доступа к данным, которые будут использовать данные из базы данных SQL Server.
7. Скопирует все макросы и модули из базы данных Access в проект, ничего в них не меняя.

Закончив работу, мастер выведет отчет, в котором перечисляются все выполненные мастером действия и указывается, какие объекты ему не удалось перенести в новое приложение.

После закрытия отчета вместо текущей базы данных Access откроет новый проект.

В окне проекта Access имеется девять типов объектов. Таблицы, представления, схемы данных и хранимые процедуры содержатся в базе данных SQL Server, а формы, отчеты, страницы доступа к данным, макросы и модули – в проекте Access.

Для создания или изменения объектов в проекте Access существует набор конструкторов. По своему назначению и внешнему виду конструкторы для проектов Access сходны с конструкторами для баз данных.

Схемы баз данных SQL Server очень похожи на схемы баз данных Access, однако представляют собой более гибкий и мощный инструмент. В частности, из окна схемы баз данных SQL Server можно удалить таблицу из базы данных, создать новую таблицу, управлять представлением таблиц в схеме данных. Эти операции можно выполнить с помощью контекстного меню схемы данных.

Представления – в проекте Access играют ту же роль, что и запросы на выборку в базе данных Access. Представления SQL Server можно рассматривать просто как оболочку для SQL-инструкций Select.

Хранимые процедуры – это наборы операторов SQL и операторов управления потоком, которые компилируются и хранятся в базе данных на сервере.

Триггеры – особый вид хранимой процедуры, которая автоматически выполняется при вставке, обновлении или удалении записи. Можно считать, что это просто обработчик событий таблицы.

#### Создание хранимых процедур

Если нужно сделать выбор между выполнением оператора SQL, хранящегося в клиентской подсистеме (задаваемого непосредственно или хранящегося в MDB-файле запроса Access) и выполнением того же кода, хранящегося на сервере в виде хранимой процедуры, всегда лучше выбрать хранимую процедуру. На это есть несколько причин.

- Хранимые процедуры содержатся на сервере, причем в откомпилированном виде, поэтому выполняются быстрее SQL-кода, который хранится на клиенте, должен быть переслан серверу, интерпретирован, откомпилирован и выполнен.
- Использование хранимых процедур позволяет централизовать логику приложения, сократить количество избыточного кода, тем самым упростить сопровождение проекта.
- В хранимые процедуры можно включать сложные операторы управления потоком и транзакциями, благодаря чему программный код клиентской части проекта упрощается.
- Возможность организации доступа к данным таблиц посредством хранимых процедур избавляет от необходимости предоставлять пользователям разрешение на доступ к таблицам, чем упрощает защиту базы данных, уменьшая вероятность внесения в нее нежелательных изменений.

Хранимую процедуру можно создать с помощью конструктора запросов, если хранимая процедура будет содержать единственный оператор SQL. Если запрос не может быть создан или отредактирован с помощью конструктора, используется текстовый редактор запросов.

#### Синтаксис хранимых процедур

При использовании текстового редактора хранимая процедура SQL Server создается с помощью оператора SQL CREATE PROCEDURE.

```
CREATE PROCEDURE имя_процедуры
```

```
AS
```

```
Операторы SQL
```

#### Пример:

С помощью оператора UPDATE стоимость блюд в меню увеличивается на 10%

```
CREATE PROCEDURE procMenuPricesIncrease
```

```
AS
```

```
UPDATE tblMenu
```

```
SET Price = Price*1.1
```

#### Параметры и переменные

Хранимые процедуры SQL Server могут иметь параметры и переменные, а также возвращать значения.

```
CREATE PROCEDURE имя_процедуры
```

```
@параметр1 тип_данных [=значение по умолчанию] [OUTPUT]
```

```
@параметр2 тип_данных [=значение по умолчанию] [OUTPUT]
```

```
...
```

```
AS
```

```
Операторы SQL
```

Имена всех параметров хранимой процедуры обязательно должны начинаться с символа @, и все они являются локальными для процедуры. По умолчанию все параметры считаются входными. Если параметр выходной – после его объявления помещается ключевое слово OUTPUT.

**Пример** использования входных параметров:

Из таблицы Заказчики выбрать строку, соответствующую заказчику с заданным номером:

```
CREATE PROCEDURE procGetCustomer
@custid INT
As
SELECT * FROM      Заказчики
WHERE код_заказчика = @custid
```

При выполнении хранимой процедуры из окна базы данных Access система сама запрашивает значение каждого параметра, подобно тому, как она это делает при выполнении обычного запроса с параметрами.

Бывает необходимо, чтобы хранимая процедура возвратила информацию другой процедуре - вызывающей процедуре. Для этой цели используются выходные параметры.

Для присваивания значения выходному параметру используется оператор SET:

```
SET @параметр = значение
```

**Пример** использования выходных параметров:

Создать процедуру, добавляющую в таблицу Заказы новый заказ и возвращающую номер заказа.

```
CREATE Procedure procInsertOrder
    @orderdate datetime,
    @customerid int,
    @orderid int OUTPUT
As
    set nocount on
    INSERT INTO Заказы
    (Дата_заказа, код_заказчика)
    VALUES
    (@orderdate, @customerid)
    -- Возвращаем вызывающей программе
    -- новое значение Orderid в выходном параметре
```



```
SET @orderid = @@IDENTITY
```

В этой процедуре параметру @orderid присваивается значение специальной переменной T-SQL @@IDENTITY, которая возвращает значение последнего идентификатора записи.

В дополнение к параметрам хранимая процедура может иметь и локальные переменные. Для их объявления предназначен оператор DECLARE:

```
DECLARE @переменная тип_данных
```

В качестве вызывающей процедуры можно воспользоваться такой процедурой:

```
CREATE Procedure procInsertOrderTest
```

```
As
```

```
    DECLARE @intOrderId INT
```

```
    EXECUTE procInsertOrder
```

```
    '9/1/2004', 1, @intOrderId OUTPUT
```

```
    -- возвращаем код нового заказа
```

```
    SELECT @intOrderId AS Код_нового_заказа
```

В данном примере процедура procInsertOrderTest выполняет процедуру procInsertOrder с помощью оператора T-SQL Execute.

После этого она возвращает вызывающей программе единственную запись, содержащую последнее присвоенное значение поля идентификатора записи.

### Выполнение хранимых процедур

Для выполнения хранимой процедуры из окна проекта Access

достаточно выполнить на ней двойной щелчок. Если у процедуры имеются входные параметры, Access запросит их значения. Возвращенные хранимой процедурой записи система выведет в виде таблицы.

Если хранимая процедура не возвращает записей, Access выводит диалоговое окно с сообщением о том, что процедура выполнена.

Непосредственный вывод выходных параметров хранимых процедур в Access не предусмотрен.

### Создание триггеров

Триггер – это хранимая процедура, которая автоматически выполняется при вставке, обновлении или удалении записи. Можно считать, что это просто обработчик событий таблицы.

Триггеры полезны для принудительного соблюдения определенных деловых условий или требований. В отличие от проверяемых ограничений, триггеры могут выполнять запросы к другим таблицам и могут содержать сложные инструкции SQL.

Для создания триггера в Access нужно выбрать в окне базы данных серверную таблицу и в ее контекстном меню выбрать команду Триггеры... В результате этого появится диалоговое окно Триггеры для таблицы. Здесь можно выбрать команду создания, удаления или редактирования триггера.

### Синтаксис триггеров

Триггеры создаются с помощью оператора CREATE TRIGGER:

CREATE TRIGGER имя триггера

ON имя таблицы

FOR [ Insert ], [ Update ], [ Delete ]

AS

Операторы SQL

Имена триггеров должны быть уникальными в пределах базы данных. Каждый триггер может быть связан только с одной таблицей, но для каждой таблицы можно создать любое количество триггеров.

Предложение FOR определяет, когда вызывается данный триггер. Он может вызываться для одной или нескольких таких операций, как вставка ( Insert ), удаление ( Delete ) и обновление ( Update ) записей. За ключевым словом AS следует тело триггера. Его составляют операторы, которые будут выполняться, когда SQL-Server вызовет ваш триггер.

При модификации существующего триггера, конструктор заменяет оператор CREATE TRIGGER на ALTER TRIGGER (команда замены существующего триггера).

Код триггера выполняется перед тем, как данные будут полностью сохранены, но триггер видит таблицу уже в измененном состоянии.

Пример:

Для таблицы tblMenu определен триггер, обновляющий итоговые значения в таблице tblMenuTotals каждый раз, когда обновляется, добавляется или удаляется запись в таблице tblMenu.

```
Create Trigger trgtblMenuUpdateTotalsUID
```

```
On dbo.tblMenu
```

```
For Insert, Update, Delete
```

```
As
```

```
    set nocount on
```

```
    DECLARE @curAvgPrice MONEY,
```

```
    @curMinPrice MONEY,
```

```
    @curMaxPrice MONEY
```

```
    SELECT @curAvgPrice = Avg(Price),
```

```
    @curMinPrice = Min(Price),
```

```
    @curMaxPrice = Max(Price)
```

```
    FROM tblMenu
```

```
    UPDATE tblMenuTotals
```

```
    SET AvgPrice = @curAvgPrice,
```

```
    MinPrice = @curMinPrice,
```

```
    MaxPrice = @curMaxPrice
```

```
    WHERE Id = 1
```

### Таблицы Inserted и deleted

SQL Server поддерживает две специальные таблицы, с именами inserted и deleted. В них он помещает копии вставляемых, изменяемых и удаляемых строк.

В ходе выполнения команд обновления, удаления или вставки данных SQL Server выполняет над таблицей триггера и таблицами inserted и deleted следующие операции.

- Перед вызовом Insert-триггера SQL Server добавляет новые строки в таблицу триггера и еще одну их копию в таблицу Inserted.
- Перед вызовом Delete-триггера SQL Server удаляет заданные строки из таблицы триггера и копирует их в таблицу deleted.
- Перед вызовом Update-триггера SQL Server помещает заменяемые строки в таблицу deleted. Обновленные строки добавляются в таблицу триггера и в таблицу inserted.

### Транзакции и оператор Raiseerror

Триггер всегда выполняется внутри неявной транзакции. Поэтому незачем явно помещать тело триггера между операторами BEGIN TRANSACTION и COMMIT TRANSACTION.

Единственный управляющий оператор транзакции, который не выполняется автоматически, это оператор ROLLBACK TRANSACTION. Отменить операцию удаления, обновления или добавления данных можно с помощью явного оператора ROLLBACK TRANSACTION.

Пример: Цена (Price) должна быть больше нуля. Для определения значения цены в обновляемой или добавляемой записи в триггере используется таблица inserted.

```
ALTER Trigger trgtblMenuPriceUI
On dbo.tblMenu
For Insert, Update
As
    set nocount on
    If UPDATE(Price)
        IF(SELECT Count(*)
           FROM inserted WHERE Price <= 0) > 0
        BEGIN
            RAISERROR 50001 'Цена должна быть больше 0'
            ROLLBACK TRANSACTION
        END
```

Если оказывается, что значение поля Price меньше или равно 0, транзакция отменяется с помощью оператора ROLLBACK TRANSACTION.

В данном примере с помощью оператора RAISEERROR генерируется ошибка SQL Server. Эта ошибка передается клиенту также, как если бы ее сгенерировал сам SQL Server. Одна из форм записи этого оператора:

RAISERROR код\_ошибки сообщение\_об\_ошибке

Код ошибки должен быть целым числом, а сообщение об ошибке – строкой. Чтобы коды ошибок не конфликтовали со встроенными ошибками SQL Server, назначайте им значения выше 50 000.

### Защита проектов Access (ADP)

Если вы работаете не с файлом базы данных Access (MDB), а с файлом проекта (ADP), система защиты Jet в вашем приложении не будет действовать.

Для защиты таблиц, хранимых процедур и других объектов SQL Server должна использоваться система защиты SQL Server.

Для защиты форм, отчетов, модулей можно удалить из проекта исходный код, конвертировав проект в MDE-файл. Сохранение базы данных в виде MDE-файла защищает формы и отчеты без требования регистрации пользователей и необходимости для разработчика создавать и поддерживать учетные записи пользователей и разрешения, требуемые для защиты на уровне пользователей.

Для защиты страниц доступа к данным можно опубликовать их на защищенном Web-сервере.

## 3. Лабораторные задания.

### *Упражнение 1.*

1. Создайте проект Access
2. Создайте хранимую процедуру с входными параметрами
3. Создайте две хранимые процедуры, первая из которых возвращает параметры, а вторая принимает и обрабатывает эти параметры.
4. Создайте для одной из таблиц триггер.
5. Протестируйте работу хранимых процедур и триггера.
6. Одну из страниц доступа, разместите на Web-сервере.
7. Протестируйте страницу в браузере.

### *Упражнение 2*

Провести аналогичную упражнению 1 работу для баз данных согласно индивидуальному заданию.

#### **4. Контрольные вопросы**

1. В каких случаях для приложения целесообразно использовать архитектуру клиент-сервер?
2. Чем отличается хранимая процедура от запроса, хранящегося в MDB файле?
3. Для чего предназначены триггеры?
4. Как организуется защита проекта Access?
5. Как организуется защита проекта SQL Server?

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Агальцов В. П. Базы данных [Электронный ресурс]. В 2-х кн. Кн.
2. Распределенные и удаленные базы данных / доп.УМО вузов по политехническому университетскому классическому образованию в качестве учебника / В.П. Агальцов. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 272 с,- ЭБС «Знаниум».
2. Кузин А. В. Базы данных [Текст] : доп. УМО вузов по унив. политехи, образованию в качестве учеб, пособия для студентов вузов, обучающихся по направлению подготовки дипломир. специалистов 654600 "Информатика и выч. техника" / А. В. Кузин, С. В. Левонисова. - 2-е изд., стер. - М. : Академия, 2008. - 320 с.
3. Кузовкин А. В. Управление данными [Текст] : доп. УМО вузов по унив. политехи, образованию в качестве учеб, для студентов высш. учеб, заведений / А. В. Кузовкин, А. А. Цыганов, Б. А. Щукин. - М. : Академия, 2010. - 256 с.
4. Нестеров С.А. Базы данных [Электронный ресурс]: учебное пособие. - СПб.: Изд-во Политехи, ун-та, 2013. - 250 с. - ЭБС "Единое окно".
5. Советов Б. Я. Базы данных [Текст] : теория и практика : рек. УМО по унив. политехи, образованию в качестве учеб, для студентов вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовский. - Изд. 2-е, стер. - М. : Высш. тик., 2007. - 463 с.
6. Хомоненко А. Д. Базы данных [Текст] : учеб, для высш. учеб, заведений : рек. УМО по образованию / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; под ред. А. Д. Хомоненко. - 6-е изд. - М. : Бином-Пресс; СПб. : КОРОНА-век, 2007. - 736 с.
7. Шаптала В.В. Управление данными: лабораторный практикум. Белгород: Изд-во БГТУ им. В.Г. Шухова, 2005. – 101 с.
8. Новикова Т.П., Евдокимова С.А. Информационные системы управления: лабораторный практикум. Воронеж, 2016.
9. Новикова Т.П., Евдокимова С.А. Информационные технологии в менеджменте: лабораторный практикум. Воронеж, 2016.
10. Новикова Т.П. Система управления проектами дизайн-центра микроэлектроники : монография / Т.П. Новикова. Воронеж, 2014.
11. Беляева Т.П. Система управления формированием и реализацией проектов дизайн центра микроэлектроники: диссертация на соискание

ученой степени кандидата технических наук / Воронежская государственная лесотехническая академия. Воронеж, 2012

12. index\_option (Transact-SQL) [Электронный ресурс]. Режим доступа: [https://msdn.microsoft.com/ru-ru/library/ms186869\(v=sql.90\).aspx](https://msdn.microsoft.com/ru-ru/library/ms186869(v=sql.90).aspx)

13. Изучение основ языка SQL: методические указания к лабораторным работам по курсу «Базы данных» / Московский институт электроники и математики НИУ ВШЭ; Сост.: И.П. Карпова. – М., 2012 – 39 с.

14. Карпова, И.П. Введение в базы данных / Издательский дом «Питер». – 240 с.

15. Пушкинов, А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие / Изд-е Башкирского ун-та. - Уфа, 1999. - 108 с.

16. Пушкинов А.Ю. Введение в системы управления базами данных. Часть 2. Нормальные формы отношений и транзакции: Учебное пособие / Изд-е Башкирского ун-та. - Уфа, 1999. - 138 с.

17. Реляционная алгебра. Операции реляционной алгебры: МИГКУ ИТ-51вс [Электронный ресурс]. Режим доступа: <http://migku.wikidot.com/gos-db-16>

18. Пособие. Реляционная модель. Операции над данными: ВГУ. [Электронный ресурс]. Режим доступа: <https://edu.vsu.ru/mod/book/view.php?id=22326&chapterid=1696>

19. SQL - Урок 2. Типы данных [Электронный ресурс]. Режим доступа: <https://site-do.ru/db/sql2.php>

20. Проектирование реляционной базы данных: Метод. указания к курсовому проектированию по курсу "Базы данных" / Московский государственный институт электроники и математики; Сост.: Карпова И.П. – М., 2003. – 28 с.

21. Изучение языка SQL: Метод. указания к лабораторным работам по курсу "Базы данных" / Московский государственный институт электроники и математики; Сост.: И. П. Карпова. М., 2003. – 32 с.

22. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация, сопровождение. Теория и практика. – 2-е изд.: Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2000. – 1120 с.

23. Меерсон, В.Э. Методические указания к выполнению лабораторных работ по дисциплине «Информационные технологии» для студентов сокращенной формы обучения по направлению подготовки



080100.62 Экономика, профиль Бухгалтерский учет, анализ и аудит [Текст]: методические указания / В.Э. Меерсон, Т.П. Беляева. – Воронеж: ВГЛТА, 2013. – 54 с.

24. Achkasov V.N. Controlling means of development electronic component basis [Текст]: monograph / V.N. Achkasov, V.K. Zolnicov, T.P. Belyaeva . – Science Book Publishing House, Lorman, MS, USA, 2012. – 130 p.

25. Новикова, Т.П. Управление данными [Электронный ресурс] : лабораторный практикум / Т.П. Новикова, К.В. Зольников; М-во образования и науки РФ, ФГБОУ ВО «ВГЛТУ». – Воронеж, 2016. – 101 с.

26. Крёнке, Д. Теория и практика построения баз данных. 9-е изд. – СПб.: Питер, 2005 – 859 с.

## Варианты индивидуальных заданий:

## Вариант 1. Кухня.

Повар (Номер, ФИО, Место работы, Разряд)

Продукты (Шифр, Наименование, Цена, Количество)

Блюдо (Шифр, Наименование, Номер рецепта)

Изготовление (Номер повара, Шифр продукта, Шифр блюда, Расход продукта)

Изготовление показывает, какой повар, какой продукт в каком количестве использует для изготовления какого блюда.

## Вариант 2. Поликлиника.

Врач (Номер, ФИО, Категория, Специализация)

Больной (Шифр, ФИО, Дата рождения, Вес, Рост)

Лекарство (Шифр, Название, Изготовитель)

Назначение (Номер врача, Шифр больного, Шифр лекарства, Дозировка)

Назначение показывает, какому больному, какой врач какое лекарство в какой дозировке выписал.

## Вариант 3. Библиотека.

Книга (Номер, Название, Издательство, Год издания, Количество страниц)

Автор (Номер, ФИО, Город проживания)

Полка (Шифр, Название, Место нахождения)

Расстановка (Номер книги, Номер автора, Шифр полки, Количество)

Расстановка показывает, на какой полке, какого автора, какая книга стоит в каком количестве экземпляров.

## Вариант 4. Парикмахерская.

Клиент (Шифр, ФИО, Возраст, Тип волос)

Мастер (Шифр, ФИО, Квалификация)

Прически (Шифр, Название, Цена, Время)

Заказы (Шифр клиента, Шифр мастера, Шифр прически, дата, время)

Заказы показывают, какая прическа, какому клиенту, кем из мастеров была или будет сделана.

#### Вариант 5. Производство.

Рабочий (Шифр, ФИО, Разряд, Место работы)

Детали (Шифр, Наименование, Цена)

Изделие (Шифр, Наименование, Номер проекта)

Изготовление (Шифр рабочего, Шифр детали, Шифр изделия, Количество деталей)

#### Вариант 6. Станция технического обслуживания автомашин.

Мастер (Шифр, ФИО, квалификация)

Машины (Шифр, Марка, Цвет, Пробег)

Услуги (Шифр, Наименование, Цена, Время выполнения)

Заказы (Шифр мастера, Шифр машины, Шифр услуги, Дата)

#### Вариант 7. Магазин.

Товары (Шифр, Наименование, Стоимость единицы измерения, Вид единицы измерения)

Отделы (Шифр, Наименование, Этаж)

Продавцы (Шифр, ФИО, Возраст, Пол, Шифр отдела {где работает})

Продажи (Шифр товара, Шифр продавца, Шифр отдела, Количество, Дата)

Продажи показывают, какой товар, в каком количестве, когда и кем из продавцов был продан.

#### Вариант 8. Агентство по обмену жилья.

Квартиры (Шифр, Количество комнат, Площадь, Район, Этаж, Признак спрос/предложение)

Абоненты (Шифр, ФИО, Шифр квартиры)

Агенты (Шифр, ФИО, Номер кабинета)

Обмены (Шифр квартиры, Шифр абонента, Шифр агента, Дата обмена, Сумма, уплаченная первой квартирой за вторую (может быть не положительной))

Обмен показывает когда, кем и какая квартира, на какую сумму была обменена

Учебное издание

Татьяна Петровна **Новикова**  
Константин Владимирович **Зольников**

УПРАВЛЕНИЕ ДАННЫМИ

Лабораторный практикум

Редактор

Подписано в печать

Формат 60×90 /16.

Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ

ФГБОУ ВО «Воронежский государственный лесотехнический университет  
имени Г.Ф. Морозова»

РИО ФГБОУ ВО «ВГЛТУ». 394087, г. Воронеж, ул. Тимирязева, 8

Отпечатано в УОП ФГБОУ ВО «ВГЛТУ»

394087, г. Воронеж, ул. Докучаева, 10