МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

А. М. ЧЕРНОУСОВА

# СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ БАЗ ДАННЫХ

Рекомендовано Ученым советом государственного
образовательного учреждения высшего профессионального образования
«Оренбургский государственный университет» в качестве учебного пособия
для студентов, обучающихся по программам высшего профессионального
образования по специальностям «Автоматизация технологических процессов и
производств» и «Системы автоматизированного проектирования»

Оренбург 2009

УДК 004.65(075.8)
ББК 32.973.26-018.2я73
Ч49

Рецензент
заведующий кафедрой «Программное обеспечение и управление в технических системах» Поволжской государственной академии телекоммуникаций и информатики, профессор, доктор технических наук В. Н. Тарасов

**Черноусова, А. М.**
Ч49 **Создание и использование баз данных: учебное пособие/ А. М. Черноусова. - Оренбург: ГОУ ОГУ, 2009. - 244 с.**
**ISBN**

Учебное пособие предназначено для выполнения лабораторных и самостоятельных работ при изучении компьютерных технологий создания баз данных в Microsoft Office Access 2007 и Delphi 7, а также их использования.

Представлен методический материал по 14 работам. Каждая работа включает теоретическое изложение материала, задания на выполнение лабораторных работ, содержание отчёта. Для самоподготовки к каждой работе приводятся тесты и контрольные вопросы.

Учебное пособие предназначено для студентов всех форм обучения, изучающих дисциплину «Базы данных». Приведенные примеры помогут освоить полученные знания и применить их на практике. Учебное пособие может быть использовано аспирантами и инженерно-техническими работниками при работе с базами данных в Microsoft Office Access 2007 и Delphi.

Ч $\frac{2404010000}{}$

ББК 32.973.26-018.2я73

ISBN

# Содержание

# Введение

Использование вычислительной техники для автоматизации управления и проектирования невозможно без рациональной организации данных и обеспечения эффективного доступа к ним пользователей. Широкое распространение персональных компьютеров приводит к непрерывному возрастанию важности организации информации в виде баз данных (БД).

*База данных* – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ [1]. Для создания, наполнения, обновления и удаления баз данных появилось множество различных систем управления базами данных (СУБД).

Большинство современных СУБД, используемых на персональных компьютерах, ориентированы на реляционные модели данных, представляющие собой совокупность двумерных таблиц [2 - 12]. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Записи представляют собой строки таблицы, которые можно добавлять, удалять или изменять. Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию. Однотипные поля разных записей образуют столбец таблицы. Записи одной таблицы могут содержать ссылки на данные другой таблицы, такая взаимозависимость таблиц называется связью. Другие модули базы данных предназначены для обработки информации, хранящейся в таблицах. С помощью запросов производится выборка данных, отвечающих определенным условиям. Формы служат для форматированного ввода и вывода информации. Отчеты обеспечивают вывод оформленного списка записей с заголовками, пунктами и подпунктами.

Одной из наиболее популярных систем управления базами данных является Microsoft Office Access 2007 (далее – Access), входящая в состав последней версии популярного пакета Microsoft Office 2007 [13 - 17]. Достоинствами программного средства являются удобный интерфейс, понятные инструменты и приходящие на помощь мастера таблиц, форм и отчетов. На создание таблиц данных, их редактирование, разработку запросов, для поиска нужных данных затрачивается минимум усилий.

В учебное пособие включены семь лабораторных работ с базой данных в Access: «Создание таблиц, ввод и редактирование данных в Microsoft Office Access 2007», «Поиск данных в Microsoft Office Access 2007», «Создание и использование форм для ввода и редактирования данных в Microsoft Office Access 2007», «Создание отчетов в Microsoft Office Access 2007», «Реляционные базы данных в Microsoft Office Access 2007», «Создание запросов в Access с помощью SQL», «Разработка приложений в Microsoft Office Access 2007». Выполнив лабораторные работы, пользователь Access сможет самостоятельно создавать приложения по работе с базой данных, имеющей достаточно сложные структуры данных с несколькими таблицами, расширенные возможности поиска, сортировки и форматированного представления информации.

Для создания и использования баз данных, а также разработки разнообразных приложений применяется система визуального программирования Borland Delphi [18 - 26]. Система обладает практически всеми возможностями современных СУБД, таких как, например, Access или Visual Fox Pro. Она имеет развитые возможности по созданию пользовательского интерфейса с помощью широкого набора инструментальных программных средств, позволяет визуально подготавливать запросы к базам данных, а также непосредственно писать запросы на языке SQL. В системе имеются развитые средства отладки, облегчающие разработку приложений. Простота и естественность языка, ориентация системы на разработку приложений баз данных, наконец, эффективность (большая производительность и относительно небольшие размеры) создаваемых с ее помощью программ сделали Delphi незаменимым средством разработки клиентских мест

В учебное пособие включены методические указания по созданию и использованию баз данных в Delphi 7 для семи лабораторных работ: «Формирование базы данных в системе Delphi», «Разработка простейшего приложения базы данных в Delphi», «Управление и фильтрация в базе данных в Delphi», «Создание диаграмм и отчетов при работе с базой данных в Delphi», «Реляционный способ доступа к данным при работе в Delphi», «Работа с таблицами данных в Delphi», «Создание базы многомерных данных в Delphi».

Для каждой лабораторной работы приводятся общие теоретические положения, задания на выполнение работы, содержание отчета. Для самоподготовки предлагаются тесты и контрольные вопросы. Методические указания имеют контекстные иллюстрации, позволяющие лучше усваивать материал.

В основе учебного пособия лежат методические указания к лабораторным практикумам, разработанные автором [27 - 29].

С целью более глубокого изучения технологии разработки приложений баз данных в программных средствах Microsoft Office Access 2007 и Delphi читатель может обратиться к литературе, указанной в пособии.

Автор с благодарностью примет замечания и предложения по содержанию учебного пособия по адресу chern@mail.osu.ru.

# 1 Создание таблиц, ввод и редактирование данных в Microsoft Office Access 2007

Целью работы является создание реляционной базы данных в Microsoft Office Access 2007 в режиме таблицы, окне конструктора и с помощью шаблонов, приобретение навыков работы с созданными таблицами.

## 1.1 Общие положения

### 1.1.1 Назначение Access

*База данных* (БД) - совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ [1].

Наиболее распространенным типом модели представления данных является реляционный (relational) тип [2 - 12]. Название «реляционный» объясняется тем, что каждая запись в базе данных содержит информацию, относящуюся (related) к одному конкретному объекту. Кроме того, с информацией, принадлежащей разным объектам, можно работать как с единым целым, основанным на значениях связанных между собой (related) данных. В реляционных СУБД все обрабатываемые данные представляются в виде таблиц.

Одной из самых популярных сегодня настольных систем управления базами данных, включенных в богатое окружение продуктов семейства Microsoft Office, является Microsoft Office Access 2007 (далее - Access). Она предназначена, как на начинающего, так и на квалифицированного пользователя. База данных Access представляет набор данных и объектов (таких как таблицы, запросы и формы), относящихся к определенной задаче [13 - 17].

Основными функциями Access являются:

- определение данных (Data definition), то есть определение структуры и типа данных, а также указание, как эти данные связаны между собой;
- обработка данных (Data manipulation), включающая поиск, фильтрацию, сортировку, вычисление; обработка предусматривает также объединение данных с другой связанной с ними информацией;
- управление данными (Data control), то есть указание, кому разрешено пользоваться данными и актуализировать базу данных, а также определение правил коллективного пользования данными.

Access предоставляет максимальную свободу в задании типа данных - текст, числовые данные, даты, время, денежные значения, рисунки, звук, документы, электронные таблицы. Имеется возможность задавать форматы хранения (длина строки, точность представления чисел и даты/времени) и представления этих данных при выводе на экран или печать.

Access является современным приложением Windows и позволяет использовать все возможности DDE (Dynamic Data Exchange) – динамический обмен данными и OLE (Object Linking and Embedding) – связь и внедрение объектов. DDE обеспечивает обмен данными между MS Access и любым другим

приложением Windows. OLE устанавливает связь с объектами другого приложения или внедряет какой-либо объект в базу данных Access; в качестве объектов могут выступать рисунки, диаграммы, электронные таблицы или документы из других приложений Windows. Access может работать с большим числом разнообразных форматов данных, позволяя осуществлять импорт и экспорт данных из файлов текстовых редакторов и электронных таблиц. Access способна непосредственно обрабатывать файлы Paradox, dBase III, dBase IV, FoxPro и другие.

СУБД Access для работы с данными использует процессор баз данных Microsoft Jet, объекты доступа к данным и средство быстрого построения интерфейса - конструктор форм. Для получения распечаток используется конструктор отчётов. Автоматизация рутинных операций может выполняться с помощью макрокоманд. Несмотря на свою ориентированность на конечного пользователя, в Access присутствует язык программирования Visual Basic for Application, который позволяет создавать массивы, свои типы данных, контролировать работу приложений.

Access имеет три основных режима работы:

- режим запуска, позволяющий осуществлять сжатие и восстановление базы данных без ее открытия;
- режим конструктора, в котором можно создавать и модифицировать структуру таблиц и запросов, разрабатывать формы для отображения и изменения данных, а также производить формирование отчётов перед печатью;
- режим выполнения, при котором в главном окне выводятся окна объектов баз данных.

В состав любой базы данных Access входят следующие элементы:

- *таблицы*, которые состоят из записей, содержащих данные о конкретном предмете;
- *формы*, используемые для ввода и просмотра таблиц в окне формы и позволяющие ограничить объем информации, отображаемой на экране в требуемом виде;
- *отчёты*, используемые для отображения информации, содержащейся в базе данных;
- *запросы*, являющиеся средством извлечения информации из базы данных;
- *модули*, содержащие VBA-код, используемый для написания процедур обработки событий.

### 1.1.2 Создание таблицы базы данных

Запуск Access осуществляется командой **Пуск => Программы => Microsoft Office => Microsoft Office Access 2007.** После запуска Access отображает диалоговое окно, которое позволяет создавать новую базу данных или открыть уже существующую (рисунок 1.1).

В Access поддерживаются два способа создания базы данных:

- создать пустую базу данных, а затем добавить в неё таблицы, формы,

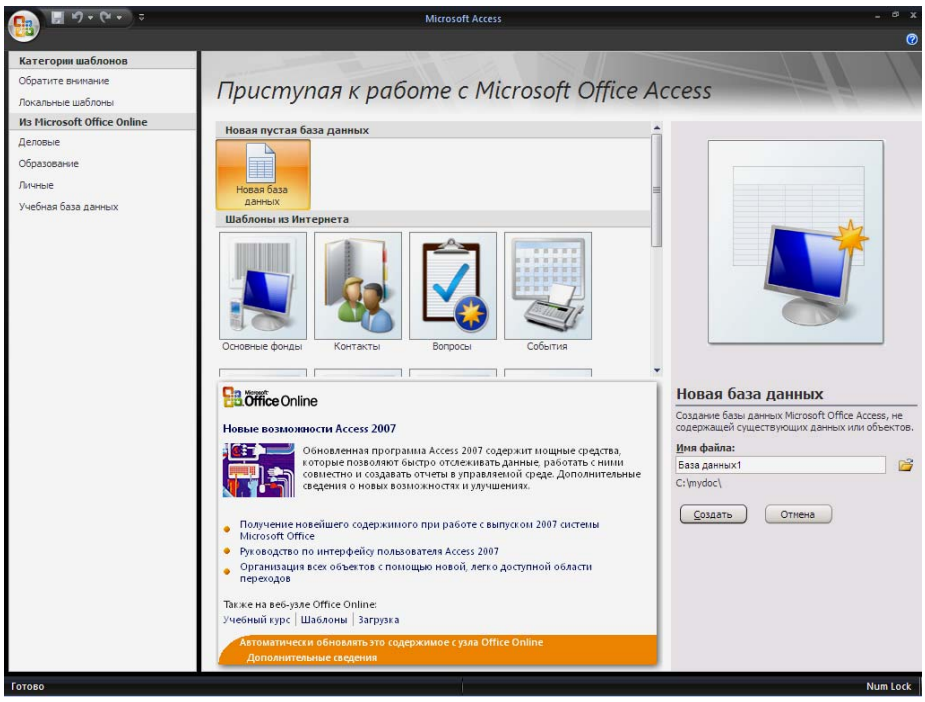



Рисунок 1.1 - Окно Microsoft Access 2007

отчёты и другие объекты, которые будут нужны;

- использовать один из имеющихся шаблонов баз данных, которые поставляются вместе с Access.

Первый способ является наиболее гибким, но требует отдельного определения каждого элемента базы данных. Второй способ во многих случаях может оказаться предпочтительным, поскольку Access содержит большой набор подготовленных баз данных. Но в обоих случаях имеется возможность изменить и расширить созданную базу данных.

Для создания новой базы данных необходимо выполнить следующие действия:

- на странице «Приступая к работе с Microsoft Office Access» в разделе **Новая пустая база данных** выбрать команду **Новая база данных**;

- в области **Новая база данных** (на рисунке 1.1 справа) в поле **Имя файла** ввести имя файла; если имя файла указано без расширения, расширение будет добавлено автоматически; чтобы сохранить файл в другой папке, отличной от используемой по умолчанию, нажать кнопку **Открыть**  (рядом с полем **Имя файла**), перейти к нужной папке и нажать кнопку **ОК**;

- нажать кнопку **Создать**.

В базе данных Microsoft Office Access 2007 объекты и данные сохраняются в ACCDB-файле, а в более ранних версиях Access используется формат MDB.

Access создаст базу данных с пустой таблицей с именем «Таблица1» и откроет эту таблицу в режиме таблицы. *Таблица* - объект базы данных, в котором данные хранятся в виде записей (строк) и полей (столбцов). *Поля* (столбцы) таблицы служат для хранения различных характеристик субъектов, а каждая

*запись* содержит сведения о конкретном субъекте. Для каждой таблицы можно определить первичный ключ (одно или несколько полей, имеющих уникальные значения в каждой записи). Данные в отдельной таблице обычно относятся к определенной категории (например, сведения о сотрудниках или заказах). Режим таблицы - режим, в котором данные из таблицы, формы, запроса, представления или хранимой процедуры выводятся в формате строк и столбцов. В режиме таблицы можно изменять поля, добавлять или удалять данные и выполнять их поиск. В Microsoft Office Access 2007 в этом режиме также можно изменять и добавлять поля таблицы.

В верхней части главного окна находится **Лента**, на которой находятся кнопки управления, представленные виде пиктограмм выполняемых ими действий. Под **Лентой** находится строка с **вкладками документов**. Слева находится **Область переходов**, внизу находится **Строка состояния**. В левом верхнем углу располагается **Кнопка «Office»**  .

**Лента** (рисунок 1.2) заменяет меню и панели инструментов, а также служит основным командным интерфейсом в Microsoft Office Access 2007. Главное преимущество ленты состоит в том, что в ней сосредоточены все средства выполнения задач, которые раньше находились в меню, панелях инструментов, областях задач и других компонентах интерфейса пользователя. Благодаря этому нужную команду не приходится искать в нескольких разных местах.

**Лента** содержит ряд вкладок с командами. В Access основные вкладки команд - **Главная**, **Создание**, **Внешние данные** и **Работа с базами данных**. Каждая вкладка содержит группу связанных команд, которые могут открывать другие новые элементы интерфейса, например, коллекцию, являющуюся элементом управления, позволяющим выбирать варианты по внешнему виду. Команды на ленте соответствуют текущему активному объекту.

При создании новой базы данных курсор находится в первой пустой ячейке столбца **Добавить поле**. Структура таблицы создается при вводе данных - при каждом добавлении нового столбца в таблицу определяется новое поле. Access автоматически задает тип данных для каждого поля на основе введенных данных. Например, при вводе в столбец даты «01.01.2009» Access распознает, что введена именно дата, и задает для этого поля тип данных «Дата / Время». Если на основании введенных данных Access не может точно определить тип данных, задается тип данных «Текстовый».


Рисунок 1.2 – Вид **Ленты** в режиме таблицы

Чтобы явно задать тип данных и формат для поля, переопределив тип, назначенный Access, необходимо использовать команды в группе «Форматирование и тип данных» на вкладке **Таблица** (рисунок 1.3). Выбор типа данных осуществляется в раскрывающемся списке рядом с полем **Тип данных**, а затем выбирается тип данных.

Явное задание формата осуществляется выбором на вкладке **Таблица** в группе «Форматирование и тип данных», щелкнув стрелку в раскрывающемся списке рядом с полем **Формат**, а затем выбирается формат.

Сохранение таблицы осуществляется командой **Сохранить**. При нажатии на кнопку «Закрыть» открытой таблицы появится запрос на сохранение изменений в таблице. При нажатии кнопки «Да» сохраняются изменения, кнопки «Нет» - отменяются изменения, кнопки «Отменить» - чтобы оставить таблицу открытой.

Для переименования столбца (поля) необходимо дважды щелкнуть заголовок столбца и ввести новое имя. Также для переименования поля можно щелкнуть правой кнопкой мыши его заголовок и выбрать пункт **Переименовать столбец** в контекстном меню.

Если поле добавляется путем ввода данных в ячейке под заголовком **Добавить поле**, Access автоматически присваивает полю имя. Имена идут последовательно: «Поле1» для первого поля, «Поле2» для второго поля и так далее. Лучше использовать какие-то более описательные имена полей. Рекомендуется присваивать полям значимые имена, чтобы при просмотре области **Списка полей** было понятно, что содержится в каждом поле. Чтобы добавить новое поле, необходимо щелкнуть на вкладке **Режим таблицы** в группе «Поля и столбцы» на пиктограмму команды **Новое поле** . Если в таблице уже содержится большое количество полей, может понадобиться прокрутить текст вправо, чтобы увидеть столбец с заголовком **Добавить поле.**

В Access отображается область **Шаблоны полей** (рисунок 1.4) со списком обычно используемых типов полей. *Шаблон поля* – это готовый набор свойств, которые в совокупности используются в качестве основы для создания нового поля. Если дважды щелкнуть по имени поля или перетащить одно из этих полей в таблицу, будет добавлено поле с этим именем и заданы соответствующие типу этого поля значения свойств. При необходимости эти свойства можно затем изменить. Необходимо перетащить это поле в область таблицы, содержащую данные, в результате появится вертикальная линия вставки, указывающее место вставки поля.

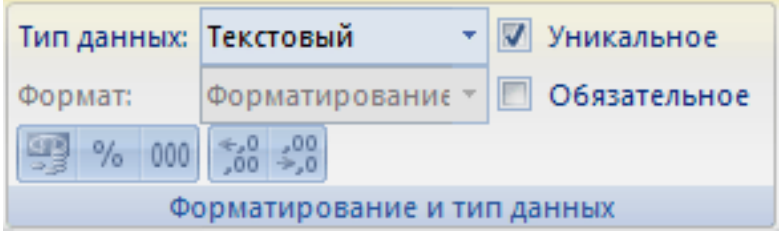



Рисунок 1.3 – Группа «Форматирование и тип данных»

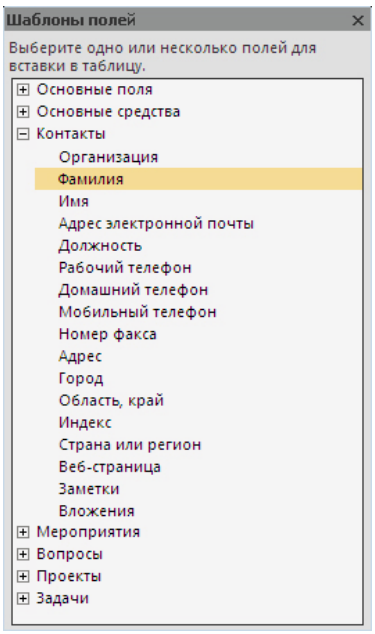



Рисунок 1.4 - Шаблоны полей

Данные можно добавить или путем непосредственного ввода в таблицу, или путем вставки из другого источника.

Для удаления столбца необходимо щелкнуть заголовок столбца правой кнопкой мыши, а затем выбрать команду **Удалить столбец** или на вкладке **Таблица** в группе «Поля и столбцы» выбрать команду **Удалить**. В режиме таблицы нельзя удалить поле, являющееся частью первичного ключа. Чтобы удалить поле первичного ключа, следует применить режим конструктора.

Ввод данных в режиме таблицы очень похож на работу в электронной таблице Microsoft Office Excel 2007.

Если первое поле является ключевым или оно определено типом **Счётчик**, то не нужно вводить информацию в это поле, Access автоматически будет увеличивать содержимое на единицу, обеспечивая уникальные значения столбца. Вставка новых данных в поле осуществляется следующим образом:

- выделить поле (или перейти к полю с помощью клавиши **Tab** или клавиш со стрелками, после чего нажать клавишу F2);
- поместить курсор в том месте, с которого требуется вводить данные;
- ввести нужный текст; если при вводе текста допущена ошибка, нажать клавишу Backspace;
- продолжать вводить информацию в строке, нажимая **Tab** или **Enter** для перехода в следующее поле и вводя в него новые данные (если пропущено поле, то можно вернуться назад, нажав комбинацию клавиш **Shift+Tab**);
- после того, как введены данные в последнее поле записи, опять нажимается **Tab**, тем самым осуществляется переход к новой записи.

При перемещении на другую запись или при закрытии формы или таблицы Access сохраняет все изменения.

В процессе ввода данных меняются пиктограммы, появляющиеся в области выбора записи (слева от первого поля) таблицы. Как только начинается

ввод или редактирование, этот символ изменяется на карандаш (  ). Это означает, что сделанные изменения ещё не сохранены. Символ звёздочка (*) появляется в новой пустой записи в конце таблицы.

Для быстрого перемещения по таблице можно использовать клавишные команды, приведённые в таблице 1.1. Операция перемещения по базе данных называется *навигацией*.

Таблица 1.1 – Клавишные команды для перемещения по таблице

| Клавишная команда | Перемещает на … |
|---|---|
| Tab или Enter | следующее поле |
| Shift+Tab | предыдущее поле |
| ↓ | следующую запись |
| ↑ | предыдущую запись |
| Home | первое поле текущей записи |
| End | последнее поле текущей записи |
| Ctrl+Home | первое поле первой записи |
| Ctrl+End | последнее поле последней записи |
| Page Down | следующую страницу |
| Page Up | предыдущую страницу |

Для замены значения в поле выделить содержимое поля, которое необходимо заменить. В режиме таблицы можно выделить поле, щелкнув рядом с левой границей поля, когда указатель мыши приобретает форму знака плюс (+). В режиме формы, чтобы выделить поле, следует щелкнуть подпись поля. Ввести или выбрать новое значение поля. Вводимые знаки или выбранное значение заменит значение в поле. При переходе к другой записи или закрытии формы или таблицы изменения будут сохранены.

Чтобы вставить в поле текущую дату, нажать одновременно клавиши **Ctrl+;** (точка с запятой). Чтобы вставить текущее время, нажать одновременно клавиши **Ctrl**+**Shift+:** (двоеточие).

Если необходимо удалить только некоторые данные, но не всю запись, то выделить данные, которые необходимо удалить, и нажать клавишу **Delete**.

Для удаления записи щелкнуть область выделения записей, расположенную рядом с записью. Чтобы расширить или уменьшить выделение, перетащить область выделения записей (если она доступна) либо нажать клавиши **Shift**+«Стрелка вниз» или **Shift+**«Стрелка вверх». Затем нажать клавишу **Delete** или на вкладке **Начальная страница** в группе «Записи» щелкнуть **Удалить**.

При удалении записи данные безвозвратно удаляются из таблицы.

Для перемещения столбца сначала щелкнуть его заголовок, чтобы выделить столбец, а затем перетащить столбец в нужное место.

Кроме того, можно выбрать сразу несколько смежных столбцов, а затем одновременно перетащить их в новое место. Чтобы выбрать несколько последовательно расположенных столбцов, щелкнуть заголовок первого столбца, а затем, удерживая нажатой клавишу **Shift**, щелкнуть заголовок последнего столбца.

Добавление таблиц к существующей базе данных осуществляется командами группы «Таблицы» на вкладке **Создание** (рисунок 1.5).

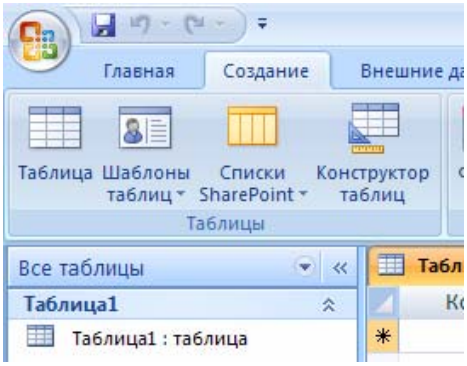



Рисунок 1.5 - Вкладка **Создание** на **Ленте**

### 1.1.3 Создание таблицы в режиме конструктора

*Конструктор* предназначен для задания и изменения структуры таблицы. Переход в режим конструктора осуществляется, если на вкладке **Создание** в группе «Таблицы» щелкнуть **Конструктор таблиц**  . Окно режима конструктора показано на рисунке 1.6.

Для каждого поля в таблице вводится имя в столбце **Имя поля**, а затем в списке **Тип данных** выбирается тип данных.

В Access 2007 предусмотрено 10 разных типов данных (в предыдущих версиях Access их было девять), и каждый тип имеет свое назначение. В приведенной ниже таблице (таблица 1.2) перечислены типы данных, показано, какие данные хранит поле каждого типа, а также описаны ограничения, налагаемые каждым типом.

При желании можно ввести описание для каждого поля в столбце **Описание**. Это описание будет отображаться в строке состояния, когда в режиме таблицы курсор будет находиться в данном поле. Это описание также отображается в строке состояния для любых элементов управления в форме или отчете, которые создаются путем перетаскивания этого поля из области **списка полей**, и любых элементов управления, которые создаются для этого поля при использовании мастера отчетов или мастера форм.

Когда все необходимые поля будут добавлены, необходимо сохранить таблицу: щелкнуть значок **Кнопка «Office»**  , а затем выбрать команду **Сохранить**.

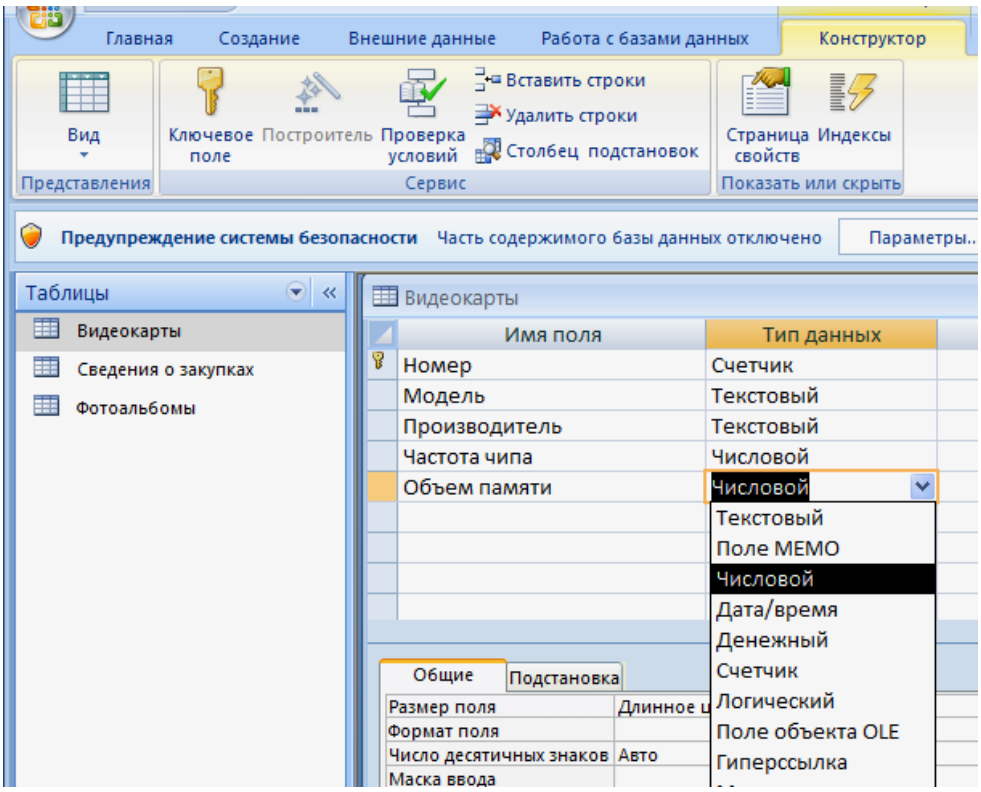



Рисунок 1.6 – Режим конструктора

Таблица 1.2 – Типы данных

| Тип данных | Хранимые данные | Ограничения |
|---|---|---|
| 1 | 2 | 3 |
| Текстовый | Алфавитно-цифровые данные (текст и числа) | Может храниться до 255 знаков. |
| Поле MEMO | Алфавитно-цифровые данные (текст и числа) | Может храниться до 2 Гб данных при программном заполнении полей. Добавление 2 Гб данных приводит к замедлению работы базы данных.<br>При вводе данных вручную в поле и в любой элемент управления, связанный с этим полем, можно ввести и просмотреть максимум 65535 знаков.<br>Поддерживают форматирование текста. |
| Числовой | Числовые данные | Используется параметр **Список полей**, управляющий размером значения, которое может содержать поле. Размер поля можно задавать равным 1, 2, 4, 8 или 16 байтам. |
| Дата/время | Значения даты и времени | Значения даты и времени хранятся в виде 8-байтовых целых чисел с двойной точностью. |
| Логический | Логические данные («истина» или «ложь») | Используется «1» для всех значений «Да» и «0» для всех значений «Нет». |

Продолжение таблицы 1.2

| 1 | 2 | 3 |
|---|---|---|
| Денежный | Денежные данные | Данные хранятся в виде 8-байтовых чисел с точностью до четырех знаков после запятой. Этот тип данных используется для хранения финансовых данных и в тех случаях, когда значения не должны округляться. |
| Счетчик | Уникальные значения, создаваемые приложением Access при введении новой записи | Данные хранятся в виде 4-байтовых значений; обычно используются в первичных ключах. |
| Поле объекта OLE | Изображения, документы, диаграммы и другие объекты из приложений Office и других программ Windows | Может храниться до 2 Гб данных. Создают растровые изображения исходных документов или других объектов, а затем отображают их в полях таблиц и элементах управления форм или отчетов в базе данных.<br>Чтобы в Access выводились эти изображения, необходимо, чтобы на компьютере, использующем базу данных, был зарегистрирован OLE-сервер (программа, поддерживающая этот тип файлов). Если для данного типа файлов OLE-сервер не зарегистрирован, отображается значок поврежденного изображения. Как правило, в ACCDB-файлах вместо типа данных «Поле объекта OLE» используется тип «Вложение». |
| Гиперссылка | Веб-адреса | Может храниться до 1 гигабайта данных. Это могут быть ссылки на веб-узлы, на узлы или файлы интрасети или локальной сети, а также на узлы или файлы локального компьютера. |
| Вложение | Файлы любого поддерживаемого типа | Новая функциональная возможность ACCDB-файлов Office Access 2007. В записи базы данных можно вкладывать изображения, файлы электронных таблиц, документы, диаграммы и другие файлы поддерживаемых типов. Можно также просматривать и редактировать вложенные файлы в зависимости от параметров, заданных разработчиком базы данных для поля с типом данных «Вложение». Эти поля дают большую свободу действий, чем поля с типом данных «Поле объекта OLE», и более рационально используют место для хранения, поскольку не создают растровые изображения исходного файла. |

При создании таблицы в режиме таблицы или конструктора можно проверить и задать свойства полей. Это можно сделать только в режиме конструк-

тора. Для отображения свойств полей щелкнуть поле на сетке конструктора. Свойства отображаются под сеткой конструктора в области **Свойства поля**. В таблице 1.3 приведены сведения о наиболее часто используемых свойствах полей.

Таблица 1.3 – Свойства полей

| Свойство | Описание |
|---|---|
| Размер поля | Для текстовых полей это свойство указывает максимально допустимое количество знаков, сохраняемых в этом поле. Максимальное значение - 255. Для числовых полей это свойство указывает тип сохраняемых чисел («Длинное целое», «Двойное с плавающей точкой»). Для более рационального хранения данных рекомендуется выделять для хранения данных наименьший необходимый размер памяти. Если потребуется, это значение позже можно изменить. |
| Формат | Это свойство определяет формат отображения данных. Оно не влияет на фактические данные, сохраняемые в этом поле. Можно выбрать встроенный формат или задать пользовательский формат. |
| Маска ввода | Это свойство используется для определения общего шаблона для ввода любых данных в это поле. Это позволяет обеспечить правильный ввод и нужное количество знаков для всех данных. Для получения справки по созданию маски ввода нажать кнопку ... справа от поля свойства. |
| Значение по умолчанию | Это свойство используется для задания значения по умолчанию, которое будет отображаться в этом поле при каждом добавлении новой записи. Например, для поля «Дата/время», в котором необходимо записывать дату добавления каждой записи, в качестве значения по умолчанию можно ввести «Date()» (без кавычек). |
| Обязательное | Это свойство указывает, обязательно ли вводить значение в это поле. Если для этого свойства задано значение **Да**, невозможно будет добавить новую запись, если в это поле не введено значение. |

Для удаления поля в режиме конструктор необходимо выделить поле (строку), которую требуется удалить (щелкнуть область выделения этой строки), нажать клавишу **Delete**, или на вкладке Конструктор в группе Сервис выбрать команду **Удалить строки**. Чтобы сохранить изменения, нажать кнопку **Сохранить** на панели быстрого доступа.

Перед удалением поля следует убедиться, что оно не участвует в связях между таблицами. При попытке удаления поля, участвующего в отношениях, последует сообщение о том, что сначала необходимо удалить это отношение.

### 1.1.4 Создание базы данных на основе шаблонов

В Access предусмотрены разнообразные шаблоны, с помощью которых можно быстро создать базу данных. *Шаблон* - это уже готовая к использованию база данных, включающая все необходимые таблицы, запросы, формы и отчеты

для выполнения определенной задачи. Например, предусмотрены шаблоны, которые можно использовать для отслеживания вопросов, управления контактами или учета расходов. Некоторые шаблоны содержат несколько примеров записей, позволяющих продемонстрировать их использование. Шаблоны баз данных можно использовать без изменений или настроить в соответствии с конкретными потребностями.

Если один из этих шаблонов точно соответствует потребностям, с его помощью обычно проще и быстрее всего создать необходимую базу данных. Так как в шаблонах уже определена структура данных, на изменение существующих данных в соответствии с этой структурой может потребоваться много времени.

Для создания базы данных на основе шаблона, необходимо выполнить следующие действия:

- в средней части страницы «Приступая к работе с Microsoft Office Access» отобразится несколько шаблонов; щелкнуть ссылки в области **Категории шаблонов**, чтобы отобразить другие шаблоны;

- выбрать шаблон (рисунок 1.7), который необходимо использовать;

- в поле **Имя файла** предлагается имя файла для базы данных, его можно заменить на любое другое имя; чтобы сохранить эту базу данных в другой папке, отличной от отображаемой под полем имени файла, нажать кнопку  , перейти к папке, в которой необходимо сохранить базу данных, и нажать кнопку **ОК**;

- нажать кнопку **Создать** (или **Загрузить** – для загрузки шаблона Office Online).

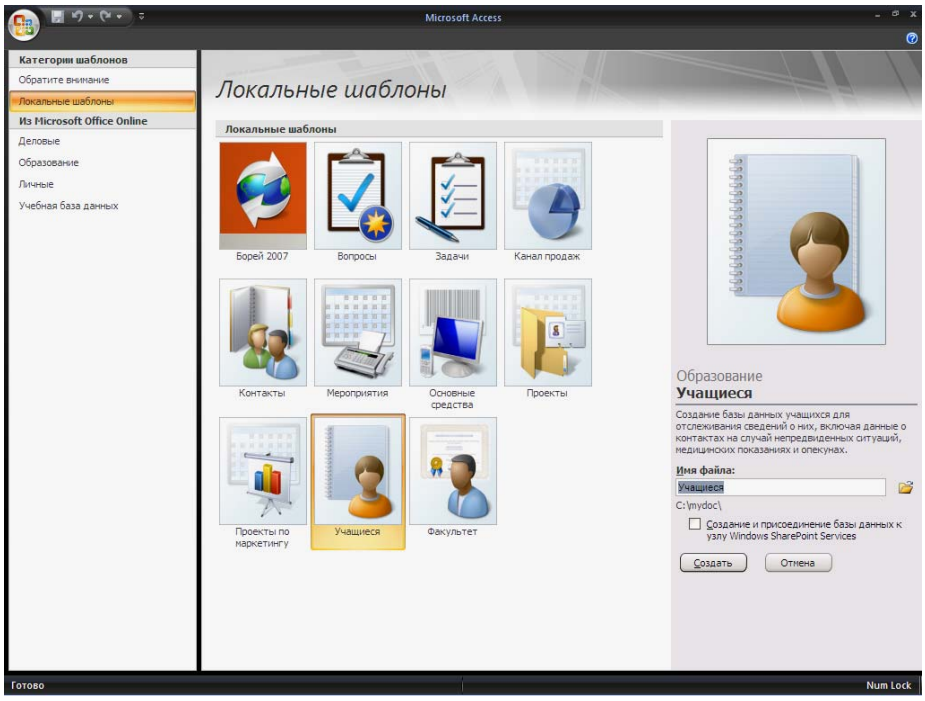



Рисунок 1.7 – Создание новой базы данных при помощи шаблонов

Access создаст или загрузит, а затем откроет базу данных. Отображается форма, в которой можно начать ввод данных. Если шаблон содержит примеры данных, можно удалить каждую из записей, щелкнув область выделения записи (затененное поле или полосу слева от записи).

Для начала работы щелкнуть первую пустую ячейку в форме и приступить к вводу данных. Используя область переходов, можно перейти к другим необходимым таблицам, формам или отчетам.

В Access имеются шаблоны для часто используемых типов таблиц. Одним щелчком мыши можно создать полную, сконфигурированную и готовую к использованию структуру таблицы. Чтобы привести таблицу в соответствие со своими запросами, пользователь может добавлять или удалять поля. Для этого нужно выполнить следующие действия:

- на вкладке **Создание** в группе «Таблицы» щелкнуть **Шаблоны таблицы** (рисунок 1.8) и затем выбрать из списка один из доступных шаблонов;
- ввести данные в первую пустую ячейку таблицы или вставить их из другого источника.

По возможности следует избегать удаления полей из баз данных, созданных на основе шаблонов: скорее всего, эти поля задействованы в других объектах базы данных, таких как формы и запросы. Таким образом, удаление поля окажет влияние на работу объектов базы данных, использующих это поле - объект базы данных не будет функционировать надлежащим образом. Чтобы такие объекты продолжали исправно работать, придется удалить все ссылки на это поле из всех объектов, использующих его.

При попытке удаления поля, участвующего в отношениях, последует сообщение о том, что сначала надо удалить это отношение.

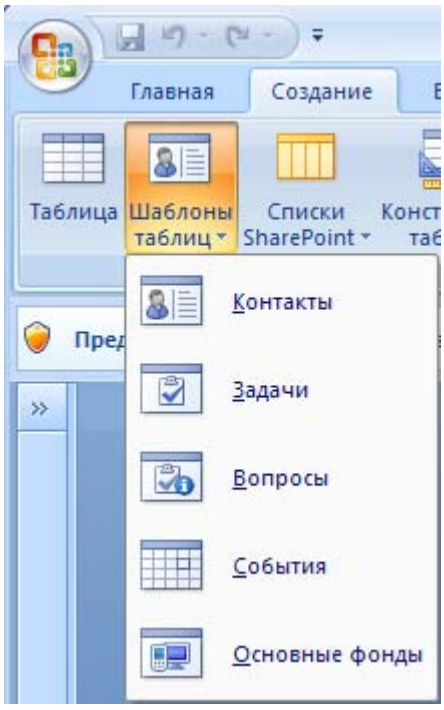



Рисунок 1.8 – Шаблоны таблиц

Удаление поля влечет за собой удаление всех данных, хранящихся в этом поле, без возможности восстановления. Поэтому следует с осторожностью удалять поля и перед этим создавать резервную копию базы данных.

### 1.1.5 Перемещение данных внутри таблицы

Для перемещения или копирования информации из одной части таблицы в другую можно использовать, как и в любом другом приложении Windows, стандартные средства: **Удалить в буфер**, **Копировать в буфер** и **Вставить из буфера**. Каждому из этих действий на панели инструментов соответствует собственная кнопка. Кроме этого можно выполнить эти действия, используя меню или клавишные команды (таблица 1.4).

Таблица 1.4 – Средства для перемещения данных

| Команда | Кнопка | Комбинация клавиш |
|---|---|---|
| Удалить в буфер |  | Ctrl+X или Shift+Delete |
| Копировать в буфер |  | Ctrl+V или Ctrl+Insert |
| Вставить из буфера |  | Ctrl+V или Shift+Insert |

Для перемещения данных внутри таблицы их необходимо предварительно выделить. Таблица 1.5 содержит описание действий с мышью, обеспечивающих выделение данных или записей в режиме таблицы.

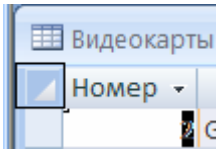
При ошибочном вводе информации не в то поле, которое нужно, можно перенести эти данные в другое место. Необходимо выполнить следующие действия:

- дважды нажать кнопку мыши на поле с ошибочно введённой информацией или установить курсор в крайнюю левую часть поля, содержащего данную информацию, и когда курсор изменится на большой белый крест, нажать на кнопку мыши для того, чтобы выделить текст всего поля;
- нажать кнопку **Вырезать** на **Ленте** в группе «Буфер обмена»;
- нажать кнопку мыши в любом месте поля, в которое нужно перенести информацию, и нажать на кнопку **Вставить**.

Для копирования данных из нескольких рядом расположенных полей в то же самое количество полей необходимо:

- выбрать копируемые поля, переместив указатель мыши по требуемым полям (начинать надо с левого края первого поля, где курсор принимает форму белого креста);
- скопировать их в буфер;

Таблица 1.5 – Действия для выделения данных

| Выделяемая область | Место установки указателя и нажатия кнопки мыши |
|---|---|
| Несколько символов в поле | В начальной позиции выделения. Для расширения области выделения перемещать указатель при нажатой кнопке мыши. |
| Всё поле | На левой границе поля таблицы, где указатель принимает вид большого белого крестика. |
| Несколько соседних полей | На левой границе поля. Для расширения области выделения перемещать указатель при нажатой кнопке мыши. |
| Столбец | В области выделения столбца. |
| Несколько соседних столбцов | В области выделения первого из выделяемых столбцов. Для расширения области выделения перемещать указатель при нажатой кнопке мыши. |
| Запись | В области выделения записи. |
| Несколько записей | В области выделения первой записи. Для расширения области выделения перемещать указатель при нажатой кнопке мыши. |
| Все записи таблицы | Нажать на значок, выделенный квадратом.   |

- выделить с помощью мыши поля, куда необходимо поместить данные; если в выбранных ячейках уже есть какая-то информация, то её можно заменить новой;

- вставить содержимое буфера.

Для изменения порядка следования полей в режиме таблицы необходимо выполнить следующую последовательность действий:

- поместить указатель мыши в верхнюю часть столбца, который надо переместить; при появлении чёрной стрелки нажать кнопку мыши, выделив этим весь столбец;

- установить указатель мыши примерно в середине имени поля, нажать кнопку мыши и держать её нажатой, при этом появится белая стрелка с маленьким прямоугольником на её конце;

- переместить поле в нужное место, при этом будет перемещаться широкая чёрная вертикальная линия; она указывает, где будет находиться левая граница поля, если отпустить кнопку.

### 1.1.6 Изменение внешнего вида таблицы

Access обладает широким набором средств для изменения внешнего вида таблиц. Можно расширить или сузить столбцы, сделать строки выше или ниже, или сделать их невидимыми, если они не нужны в данный момент.

Существует несколько способов изменения ширины столбцов. Первый способ заключается в следующем:

- поместить указатель мыши на правую границу расширяемого поля, при этом вид указателя изменится на двунаправленную стрелку;
- нажать кнопку мыши и удерживайте её, в результате появится вертикальная чёрная линия, показывающая, где будет находиться граница поля, после того как кнопка мыши будет отпущена;
- перемещать линию вправо/влево до тех пор, пока ширины поля не будет достаточно.

Второй способ увеличения ширины столбца:

- выделить столбец, ширину которого надо изменить;
- нажать на заголовке этого столбца правой клавишей мыши;
- выполнить команду **Ширина столбца…**;
- в открывшимся окне диалога «Ширина столбца» (рисунок 1.9) нажать кнопку **По ширине данных** в правом нижнем углу, а затем **ОК**.

В окне диалога «Ширина столбца» можно задать конкретное значение ширины столбца и нажать **ОК**.

На практике достаточно часто ширины экрана оказывается недостаточно, чтобы отобразить все поля таблицы. Команда **Скрыть столбцы** позволяет скрыть отдельные столбцы таблицы, оставив на экране только необходимые. Информация в скрытых столбцах не удаляется, а просто не отображается на экране.

Для того, чтобы скрыть столбец, сначала надо выбрать его нажатием на область маркировки этого столбца. Нажать на заголовке этого столбца правой клавишей мыши. Затем выполнить команду **Скрыть столбцы** из выпадающего меню.

Для отображения на экране скрытых столбцов используется команда **Показать столбцы**. Для её вызова необходимо нажать правой клавишей мыши на заголовке любого поля, в выпадающем меню выбрать команду. При выполнении данной команды на экране отрывается окно диалога «Отображение

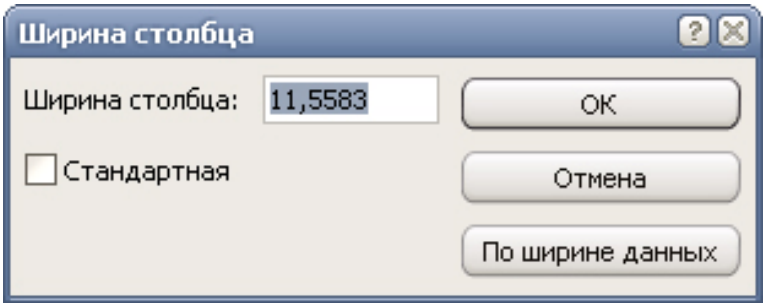



Рисунок 1.9 - Окно диалога «Ширина столбца»

столбцов» со списком всех полей таблицы (рисунок 1.10). Видимые столбцы отмечены значком «√». Чтобы сделать столбец видимым, надо установить соответствующий ему флажок. Если нужно скрыть какие-то столбцы, в этом же окне диалога нужно снять флажок отображения поля.

При просмотре таблицы с большим количеством столбцов можно перепутать информацию при переходе в правую часть таблицы, когда левые столбцы исчезают из вида. Access позволяет зафиксировать один или более столбцов, то есть сделать так, чтобы они оставались видимыми на экране при любых перемещениях других столбцов. Для этого надо воспользоваться командой **Закрепить столбцы**, предварительно выделив закреплённый столбец и нажав правую клавишу мыши. Теперь при перемещении по таблице зафиксированные столбцы останутся видимыми на экране. Для отмены закрепления столбцов надо воспользоваться командой **Освободить все столбцы**.

Access позволяет изменять параметры шрифта (размер, тип, начертание). Для этого необходимо выполнить следующие действия:

- на **Ленте** выбрать вкладку **Главная**; на ней будет присутствовать группа «Шрифт» (рисунок 1.11);

- выбрать наиболее подходящий шрифт, начертание, цвет и размер; изменения сразу отображаются на тексте.

Чтобы изменить формат отображения данных, необходимо нажать на значок  (на рисунке 1.11 находится в правом нижнем углу), откроется окно «Формат таблицы» (рисунок 1.12). В этом окне доступны команды изменения внешнего вида таблицы: **Оформление**, **Линия сетки**, **Цвет фона**, **Дополнительный цвет фона**, **Цвет линии**, **Вид границы и линий**, **Направление**.

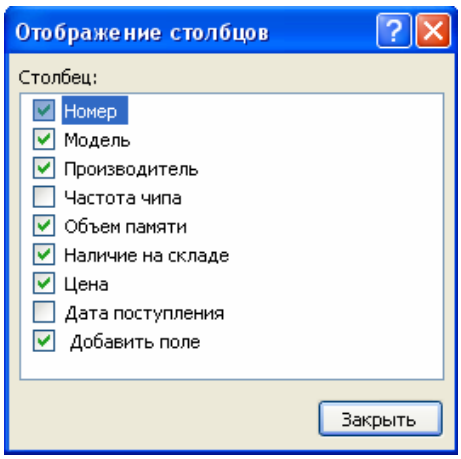



Рисунок 1.10 – Диалоговое окно «Отображение столбцов»

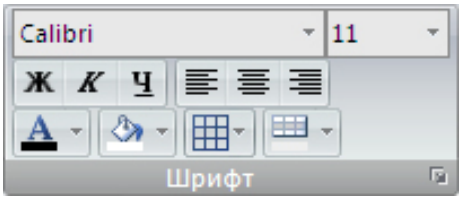



Рисунок 1.11 – Группа «Шрифт»

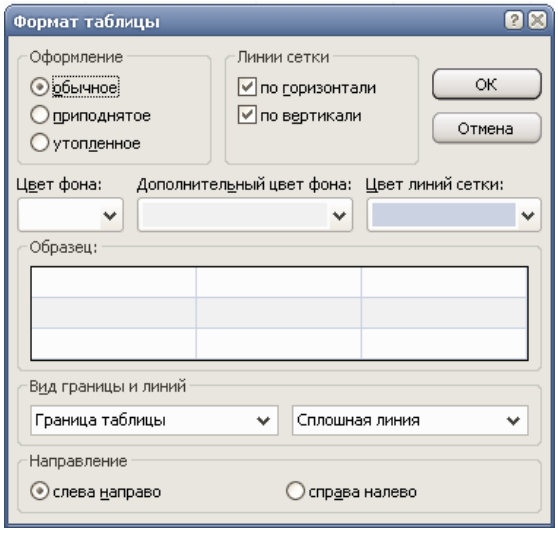



Рисунок 1.12 – Диалоговое окно «Формат таблицы»

## 1.2 Задание на выполнение работы

1.2.1 Запустить Microsoft Office Access 2007. Создать новую пустую базу данных под своим именем, сохранить её в папке на диске D:/, имеющей имя группы.

1.2.2 Создать таблицу 1.6 «Видеокарты» в режиме Таблицы.

1.2.3 Описать структуру таблицы 1.7 «Учет программного обеспечения» в окне Конструктора. Осуществить ввод данных в нее.

1.2.4 С помощью шаблона создать таблицу 1.8 «Заказы», выбрав необходимый прототип таблицы.

1.2.5 По заданию преподавателя осуществить модификацию структуры таблицы «Видеокарты».

1.2.6 Познакомиться с командами, позволяющими выполнять перемещение данных внутри таблицы.

1.2.7 Познакомиться с командами изменения внешнего вида таблицы.

1.2.8 Оформить отчёт по лабораторной работе.

Таблица 1.6 – Содержание таблицы «Видеокарты»

| Номер | Модель | Производитель | Частота чипа, МГц | Объем памяти, Мб |
|---|---|---|---|---|
| 1 | DDR-3 Sapphire ATI RADEON X1650 Pro | Sapphire | 590 | 256 |
| 2 | DDR-2 Palit GeForce 9600GT Super | Palit | 650 | 1024 |
| 3 | DDR-3 Sapphire ATI RADEON HD4670 | Sapphire | 750 | 1024 |
| 4 | DDR-2 ZOTAC GeForce 8600GT | ZOTAC | 540 | 1024 |
| 5 | DDR-3 ASUSTeK EN8800GT/HTDP GeForce 8800GT | ASUS | 600 | 512 |
| 6 | DDR-2 ASUS EN9500GT MAGIC/DI | ASUS | 650 | 512 |

Таблица 1.7 – Содержание таблицы «Учет программного обеспечения»

| Код | Программное обеспечение | Количество | Дата установки |
|---|---|---|---|
| 1 | ABBYY FineReader 7.0 | 3 | 08.10.08 |
| 2 | ASPLinux-OEM | 3 | 28.12.08 |
| 3 | Microsoft Windows Vista Business Рус. | 1 | 19.02.09 |
| 4 | Microsoft Windows Server 2003 | 2 | 06.12.08 |
| 5 | Symantec Norton AntiVirus 2008 | 1 | 12.09.08 |
| 6 | Microsoft Windows XP Professional | 20 | 18.01.09 |
| 7 | Microsoft Office 2007 | 15 | 15.12.08 |
| 8 | Антивирус Касперского 2009 Рус. | 20 | 08.02.09 |

Таблица 1.8 – Содержание таблицы «Заказы»

| Код записи | Описание | Дата | Количество | Цена |
|---|---|---|---|---|
| 1 | LG L1734S-BN | 15.10.08 | 50 | 3650 |
| 2 | LaserJet P1005 | 05.12.08 | 7 | 3600 |
| 3 | Samsung 2032BW BSFV | 23.06.08 | 24 | 9800 |
| 4 | Samsung SCX-4200 | 29.01.09 | 30 | 6200 |
| 5 | Genius KB-220 | 03.07.08 | 15 | 850 |
| 6 | Belinea 1730 S1 | 04.02.09 | 5 | 3370 |
| 7 | Logitech Deluxe 250 Y-SAF76 | 14.09.08 | 27 | 360 |

## 1.3 Содержание отчёта

1.3.1 Название работы.
1.3.2 Цель работы.
1.3.3 Структуры данных создаваемых таблиц.
1.3.4 Перечень команд, используемых при выполнении лабораторной работы, с указанием их назначения.

## 1.4 Тесты и контрольные вопросы

1.4.1 Что такое таблица в Access:
а) объект базы данных, в который добавляются элементы управления, реагирующие на действия пользователей или служащие для ввода, отображения и изменения данных в полях;
б) объект базы данных, в котором данные хранятся в виде записей (строк) и полей (столбцов);

в) объект базы данных, предназначенный для вывода на печать данных, организованных и отформатированных в соответствии с требованиями пользователя;

г) набор условий, применяемых для отбора подмножества данных или для сортировки данных?

1.4.2 Набор всех хранимых записей в одной таблице Access - это:

а) логическая схема базы данных;

б) представление базы данных;

в) хранимый файл;

г) хранимый кортеж;

д) физическая схема данных.

1.4.3 Укажите расширение файла базы данных, созданного в Access

а) accdb;

б) db;

в) dbc;

г) doc;

д) dbf.

1.4.4 Какой тип поля не поддерживает СУБД Access:

а) логический;

б) денежный;

в) знаковый;

г) текстовых примечаний;

д) числовой?

1.4.5 Какие группы команд располагаются на вкладке **Главная**?

а) «Представления», «Буфер обмена», «Шрифт», «Таблицы»;

б) «Сбор данных», «Буфер обмена», «Шрифт», «Записи»;

в) «Представления», «Форматирование и тип данных», «Шрифт», «Записи»;

г) «Представления», «Буфер обмена», «Шрифт», «Записи».

1.4.6 Каждое поле в базе данных может быть отнесено к одному из следующих типов:

а) символьный, лексический, цифровой;

б) логический, символьный, числовой, тип примечаний;

в) лексический, конкурентный, логический, физический;

г) лексический, символьный, конкурентный, логический.

1.4.7 Поле логического типа содержит:

а) величины, принимающие значения «истинно» или «ложно»;

б) логические высказывания;

в) суть логических рассуждений;

г) логические знаки.

1.4.8 Для чего служит поле типа *Memo*:

а) для хранения секретной информации;

б) для реализации других полей;

в) для хранения больших массивов данных в отдельном файле;

г) для графики?

1.4.9 Навигация - это операция:

а) хранения информации;

б) перемещения по базе данных;

в) создания эффективной структуры базы данных;

г) сортировка записей базы данных;

д) выполнения действий пользователя.

1.4.10 Ключом записи таблицы «Студент» в Access может быть поле, содержащее следующие данные:

а) номер группы;

б) факультет;

в) номер зачетной книжки;

г) изучаемая дисциплина;

д) имя студента.

1.4.11 Для чего предназначена Access?

1.4.12 Перечислите основные функции Access.

1.4.13 Что содержит **Лента** Access?

1.4.14 Какие режимы работы имеет Access?

1.4.15 Какие элементы входят в состав любой базы данных Access?

1.4.16 Какие способы создания базы данных поддерживает Access?

1.4.17 Что такое таблица?

1.4.18 В чем заключается основное преимущество **Ленты**?

1.4.19 Каким образом осуществляется создание пустой базы данных?

1.4.20 Что такое шаблон поля?

1.4.21 Каким образом создаётся таблица в окне Конструктора?

1.4.22 Какие типы данных используются в Access?

1.4.23 Что входит в описание структуры таблицы?

1.4.24 Каким образом создаётся таблица с помощью шаблонов?

1.4.25 Как создаётся таблица в режиме таблицы?

1.4.26 Как осуществить модификацию структуры таблицы?

1.4.27 Как ввести данные в таблицу?

1.4.28 Какие клавишные команды используются для перемещений по таблице?

1.2.29 Каким образом можно выделить поля и записи?

1.2.30 Как переместить данные из одного поля в другое?

1.2.31 Как копировать информацию из нескольких полей?

1.2.32 Как осуществляется изменение порядка следования полей?

1.2.33 Как можно осуществить изменение ширины столбцов?

1.2.34 С какой целью выполняется команда **Скрыть столбцы**?

1.2.35 Каким образом можно сделать столбец видимым?

1.2.36 С помощью какой команды осуществляется фиксация столбцов таблицы?

1.2.37 Как осуществляется изменение параметров шрифта?

1.2.38 Какие команды доступны в окне диалога «Формат таблицы»?

1.2.39 Что включает в себя группа «Шрифт»?

1.2.40 Как изменить имя столбца?

# 2 Поиск данных в Microsoft Office Access 2007

Целью работы является освоение различных способов поиска данных и формирование запросов в созданных таблицах Microsoft Office Access 2007.

## 2.1 Общие положения

### 2.1.1 Поиск и замена записей

Одним из основных назначений реляционных баз данных является быстрый поиск хранящейся в них информации. Простейшее средство поиска значений, содержащихся в одном из полей, заложено в команде **Найти.**

Основным достоинством этого средства является присутствие его в большинстве приложений, работающих под Windows. Практически все современные текстовые редакторы, электронные таблицы, ежедневники, средства коллективной работы представляют аналогичное средство. Другим достоинством является возможность поиска неуникальной информации, иными словами, за один поиск можно отыскать группу записей.

Для поиска записей необходимо, чтобы были открыты база данных и таблица, в которых ищется требуемая запись. Затем надо перейти в поле, по значению которого предполагается осуществить поиск, и выбрать на **Ленте** на закладке **Главная** команду **Найти**  в группе «Найти».

В результате на экране появится окно диалога «Поиск и замена», представленное на рисунке 2.1. Перейти в окно диалога «Поиск и замена» можно так же с помощью комбинации оперативных клавиш **Ctrl+F**.

Затем на вкладке «Поиск» необходимо выполнить следующие действия:

- ввести значение, которое ищется, в поле ввода **Образец**;
- щёлкнуть на кнопке **Найти далее**, чтобы найти первую запись в таблице, которая содержит указанное значение в этом поле;
- если найденная запись не отвечает заданному критерию поиска, продолжать щёлкать на кнопке **Найти далее**, пока нужная запись не будет найдена; диалоговое окно останется открытым до щелчка на кнопке **Закрыть**.

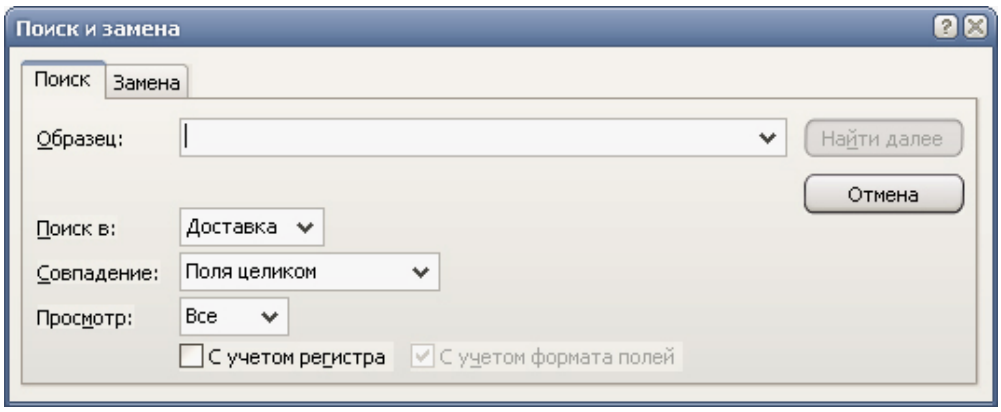



Рисунок 2.1 - Окно диалога «Поиск и замена»

Если Access не может найти запись, то на экран выводится сообщение о том, что поиск записи был завершён, и образец найден не был.

В дополнение к традиционному простому способу поиска на вкладке «Поиск» имеется несколько дополнительных опций.

Раскрывающийся список **Просмотр** содержит опции, с помощью которых определяется направление поиска:

- **Все**, когда осуществляется поиск по всему полю, начиная сначала;
- **Вверх**, когда просматриваются предыдущие записи от текущей позиции;
- **Вниз**, когда просматриваются последующие записи от текущей позиции.

По умолчанию Access ищет полностью совпадающие с образцом данные. Значения, введённые в поле ввода **Образец**, должны полностью соответствовать введённым в поля данным.

Раскрывающийся список **Совпадения** содержит три установки, которые определяют тип сравнения данных с образцом, введённым для поиска:

- **с любой частью поля** (осуществляет поиск по всему полю; если значение, которое ищется, окажется где угодно в поле поиска, то поиск завершается успешно);
- **поля целиком** (образец поиска должен совпадать с полным значением поля, что позволяет существенно сузить поле поиска);
- **с начала поля** (поиск завершается успешно только в том случае, если образец поиска находится в начале данного поля).

В средней части вкладки «Поиск» находятся два флажка, которые определяют, как будет осуществляться сравнение значения поля с образцом поиска:

- **с учётом регистра** (этот флажок указывает на то, что сравнение осуществляется с учётом регистра букв, то есть прописные и строчные буквы будут восприниматься по-разному);
- **с учётом формата полей** (формат отображения значений, отображаемых на экране, может отличаться от формата, в котором значения хранятся в таблице).

Access предоставляет возможность поиска данных, для которых точное значение неизвестно. Для замены любых неизвестных символов используются подстановочные символы. Перечень подстановочных символов, их назначение и примеры приведены в таблице 2.1.

В Access имеется возможность поиска и замены данных, которая позволяет быстро изменять записи на основе их содержимого. Для выполнения указанной операции необходимо открыть таблицу и перейти на вкладку «Замена» Аналогично команде **Поиск**, команду **Замена** следует использовать при работе с немногочисленными данным.

Окно диалога «Замена» показано на рисунке 2.2. Это окно похоже на окно диалога «Поиск». В поле **Образец** вводится значение, по которому осуществляется поиск. В поле **Заменить на** вводится то новое значение, на которое будет заменена цепочка символов, которая находится в текстовом поле **Образец**. Как и у команды **Поиск**, имеется раскрывающийся список **Просмотр**, в котором чаще используется значение **Все**.

Таблица 2.1 – Подстановочные символы

| Символ | Назначение | Пример |
|---|---|---|
| * | Соответствует любому количеству букв, цифр или других символов. Может использоваться в качестве первого или последнего символа текстовой строки | «*55*» найдёт значения «553-3586», «123-55-61» и «187-13-55» |
| ? | Соответствует любому текстовому символу | «Снег?рёв» найдёт значения «Снегирёв» и «Снегерёв» |
| [ ] | Соответствует любому одному символу из заключённых в скобки | «Снег[ие]рёв» найдёт значения «Снегирёв» и «Снегерёв» |
| [!] | Соответствует любому одному символу кроме заключённых в скобки | «Ив[!а]н» найдёт значения «Ивон», «Ивен», но не «Иван» |
| [-] | Соответствует любому символу из диапазона. Необходимо указывать этот диапазон по возрастанию (от A до Z, но не от Z до A) | «[а-в]*» соответствует любому значению, которое начинается с «а», «б» или «в» |
| # | Соответствует любой цифре | «1#3» найдёт значения «103», «113», «123» и т.д. |

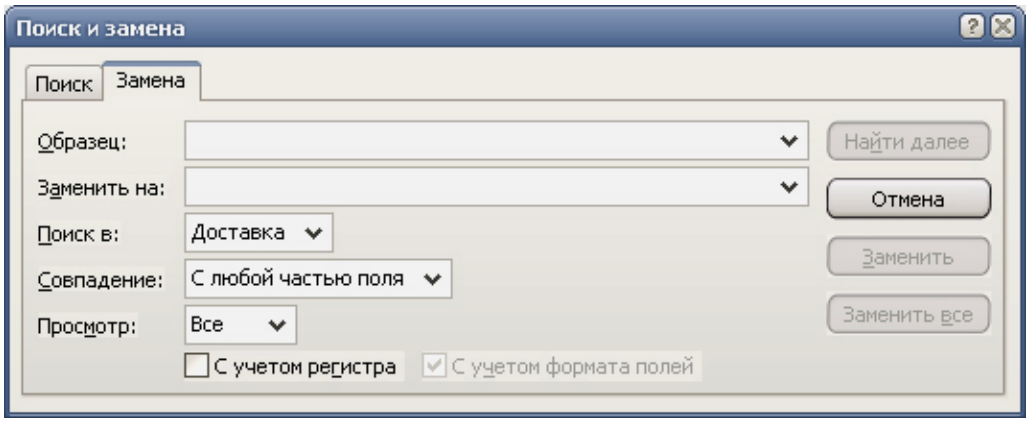



Рисунок 2.2 – Окно вкладки «Замена»

В диалоговом окне имеются характерные кнопки **Заменить** и **Заменить всё**. Если выбрать **Заменить всё**, Access заменит все вхождения, соответствующие образцу поиска, заданному в поле **Образец**. Опция **Заменить** позволяет изменять образец поиска после каждого вхождения образца в таблице.

Рекомендуется следующая стандартная процедура поиска и замены:

- ввести образец поиска в поле ввода **Образец** и нажать **Найти далее**;
- ввести новое значение, на которое надо изменить образец, в поле ввода **Заменить на** и нажать **Заменить**, Access заменит найденный образец на значение, указанное в **Заменить на**;
- после замены Access находит следующее вхождение образца и останавливает поиск, можно ввести новое значение в поле ввода **Заменить на** или оставить его без изменения.

### 2.1.2 Сортировка данных

Данные в таблицы вводятся в произвольном порядке по мере их поступления. Просмотр таких данных затруднён. Для приведения данных в определённый порядок в базах данных используется понятие сортировки, то есть упорядочивание данных по возрастанию или убыванию.

В отношении базы данных упорядочивание по возрастанию означает, что поля отсортированного текста начинаются с **А** и далее идут к **Я**, отсортированные цифровые данные идут от единицы до бесконечности, отсортированные поля дата/время располагаются по увеличению даты и времени. Очевидно, что упорядочивание по убыванию означает обратный порядок: от **Я** до **А** и так далее.

Записи можно сортировать в любом поле. После того, как выбрано поле для сортировки проведение сортировки возможно двумя способами:

а) нажать соответствующую кнопку на **Ленте** на вкладке **Главная** в группе «Сортировка и фильтр» ( - сортировка по возрастанию, - сортировка по убыванию), как показано на рисунке 2.3;

б) правой кнопкой мыши щёлкнуть по заголовку поля, которое надо использовать как основу для сортировки, и из появившегося контекстного меню выбрать опцию **Сортировка от А до Я** или **Сортировка от Я до А**.

Если сортировка проведена, то она сохранится в момент закрытия таблицы. И когда пользователь снова откроет таблицу, записи останутся отсортированными.

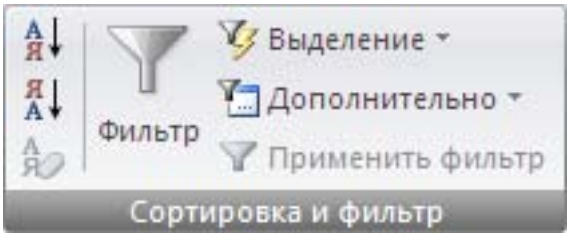



Рисунок 2.3 – Группа «Сортировка и фильтр» на **Ленте**

### 2.1.3 Использование фильтров

2.1.3.1 Общие сведения о фильтрах

*Фильтр* - набор условий, применяемых для отбора подмножества данных или для сортировки данных. Условия определяют, какие значения поля требуется отобразить. После применения фильтра в представление включаются только те записи, которые содержат указанные значения. Остальные записи будут скрыты до тех пор, пока фильтр не будет удален.

Access поддерживает следующие типы фильтров: обычный фильтр, фильтр по выделенному фрагменту, расширенный фильтр.

Для каждого типа данных в Access предусмотрено несколько готовых фильтров. Они доступны в виде команд меню в режимах таблицы, формы, от-

чета и макета. Пользователь, который может написать выражение самостоятельно, тем самым может добиться большей гибкости, создав собственные фильтры с помощью **Расширенного фильтра**.

При использовании нового фильтра для уже отфильтрованного столбца сначала удаляется старый фильтр. С отфильтрованными результатами можно работать так же, как и с первоначальным представлением, например, изменять данные или переходить к другим записям.

Чтобы вернуться к отображению без фильтров, необходимо удалить фильтры. При этом фильтр удаляется временно, чтобы можно было вернуться к исходному представлению. Для переключения отображений с фильтрами или без фильтров в строке выбора записи необходимо нажать кнопку **Без фильтра** или **С фильтром**.

Чтобы окончательно удалить фильтр из представления, необходимо очистить его. Чтобы очистить один фильтр из отдельного поля, щелкнуть левой кнопкой мыши столбец, затем на **Ленте** на вкладке **Главная** в группе «Сортировка и фильтр» команду **Фильтр** и выбрать команду **Снять фильтр с <имя_поля>**.

Чтобы очистить все фильтры из всех полей, выполняются следующие действия: на вкладке **Главная** в группе «Сортировка и фильтрация» щелкнуть **Дополнительно**, а затем выбрать команду **Очистить все фильтры** в контекстном меню.

Фильтр можно легко сохранить для дальнейшего использования. При закрытии таблицы, запроса, формы или отчета действующие параметры фильтра автоматически сохраняются вместе с объектом и становятся доступны для повторного применения. Однако по умолчанию параметры фильтра не применяются автоматически при следующем открытии объекта.

### 2.1.3.2 Фильтр по выделенному фрагменту

Если в данный момент выделено значение, которое предполагается использовать в качестве основы для фильтрации, можно быстро отфильтровать представление, выбрав одну из команды кнопки **Выделение** из группы «Сортировка и фильтр». Доступные команды будут отличаться в зависимости от типа данных выделенного значения. Эти команды доступны также в контекстном меню поля по щелчку поля правой кнопкой мыши.

Например, пусть в данный момент в таблице «Сведения о закупках» в поле «Дата покупки» выделено значение «23.06.2006». Если на вкладке **Главная** в группе «Сортировка и фильтр» нажать кнопку **Выделение**, то отобразятся команды фильтра по выделенному (рисунок 2.4). Выделенное значение автоматически включается в список команд, поэтому вводить его вручную не требуется.

Список команд зависит также от выделенной части значения. Например, при выделении только нескольких знаков значения появится другой список команд, который зависит от того, какая часть поля выделена (рисунок 2.5).

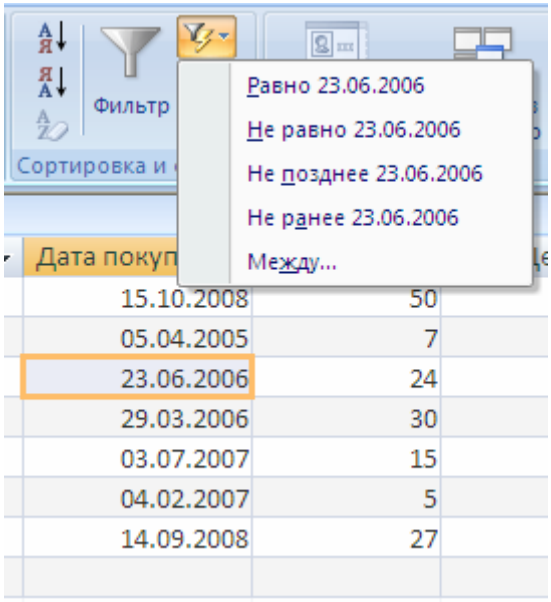
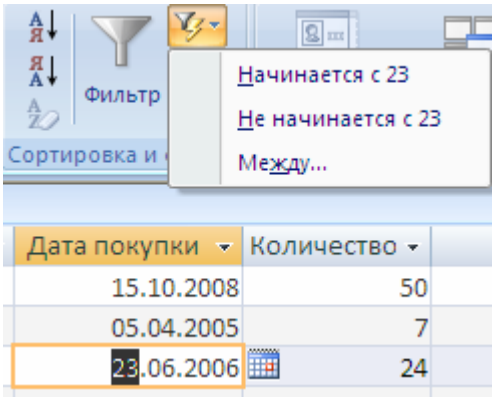
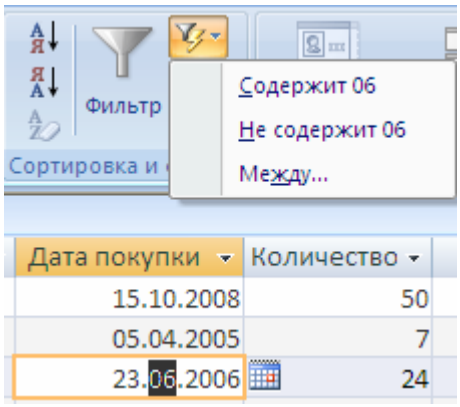


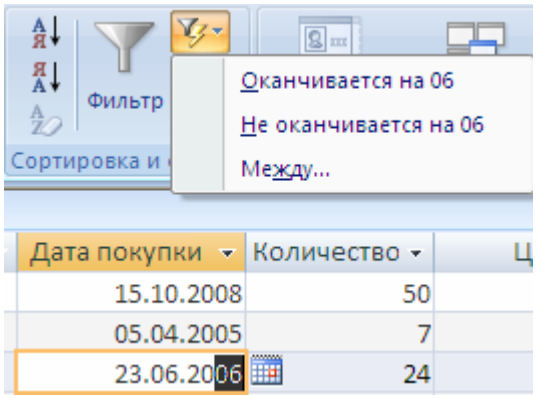

Рисунок 2.4 – Пример фильтра по выделенному




а




б




в

а – фильтр по началу значения поля; б – фильтр посередине значения поля;
в – фильтр по концу значения поля
Рисунок 2.5 – Пример фильтра по выделенной части значения

Чтобы применить **Фильтр по выделенному**, необходимо:

- открыть таблицу, запрос, форму или отчет в режиме таблицы, формы, отчета или макета;

- убедится, что представление еще не отфильтровано, в строке выбора записи проверить наличие значка **Без фильтра** или затененного значка **Нет фильтра**;

- перейти к записи, в которой содержится значение, используемое в качестве компонента фильтра, щелкнуть внутри столбца, чтобы применить фильтр по частичному выделению, и выделить нужные знаки;

- на вкладке **Главная** в группе «Сортировка и фильтрация» щелкнуть **Выделение**, а затем выбрать фильтр, который надо применить;

- чтобы отфильтровать другие поля по выделенному, повторить два предыдущих шага.

### 2.1.3.3 Обычный фильтр

Некоторые часто применяемые фильтры доступны в виде команд контекстных меню; тем самым нет необходимости тратить время на создание правильных условий фильтра. Для доступа к этим командам необходимо щелкнуть правой кнопкой мыши поле, к которому нужно применить фильтр.

При выборе нескольких столбцов или элементов управления параметры фильтра будут недоступны. Чтобы отфильтровать представление по нескольким столбцам или элементам управления, необходимо либо выбрать и отфильтровать каждый из них отдельно, либо воспользоваться параметром расширенного фильтра.

Обычные фильтры предлагаются для всех типов полей, за исключением полей объектов OLE и полей, в которых отображаются вычисленные значения. Список доступных фильтров зависит от типа данных и значений выбранного поля. Например, чтобы просмотреть доступные фильтры для поля «Дата покупки», на вкладке **Главная** в группе «Сортировка и фильтрация» необходимо щелкнуть кнопку **Фильтр**. Чтобы применить фильтр по определенным значениям, использовать список с флажками. В этом списке перечислены все значения, отображаемые в данный момент в поле. Чтобы применить фильтр по диапазону значений, щелкнуть один из этих фильтров и задать нужные значения (рисунок 2.6).

Например, чтобы просмотреть даты покупки, начиная с текущей даты и до конца года, нажать кнопку **Между** и указать начальную и конечную даты в диалоговом окне **Диапазон дат**. Чтобы увидеть все даты, можно воспользоваться списком значений. В представлении без фильтра в списке значений для каждого поля перечислены все уникальные значения, хранящиеся в этом поле.

Следует отметить, что значения в поле дат определяют список фильтров для конкретного типа. Если самое последнее значение даты приходится на последние два года, список фильтров будет более полным. Если все даты в поле имеют более чем двухгодичную давность, список фильтров будет короче.

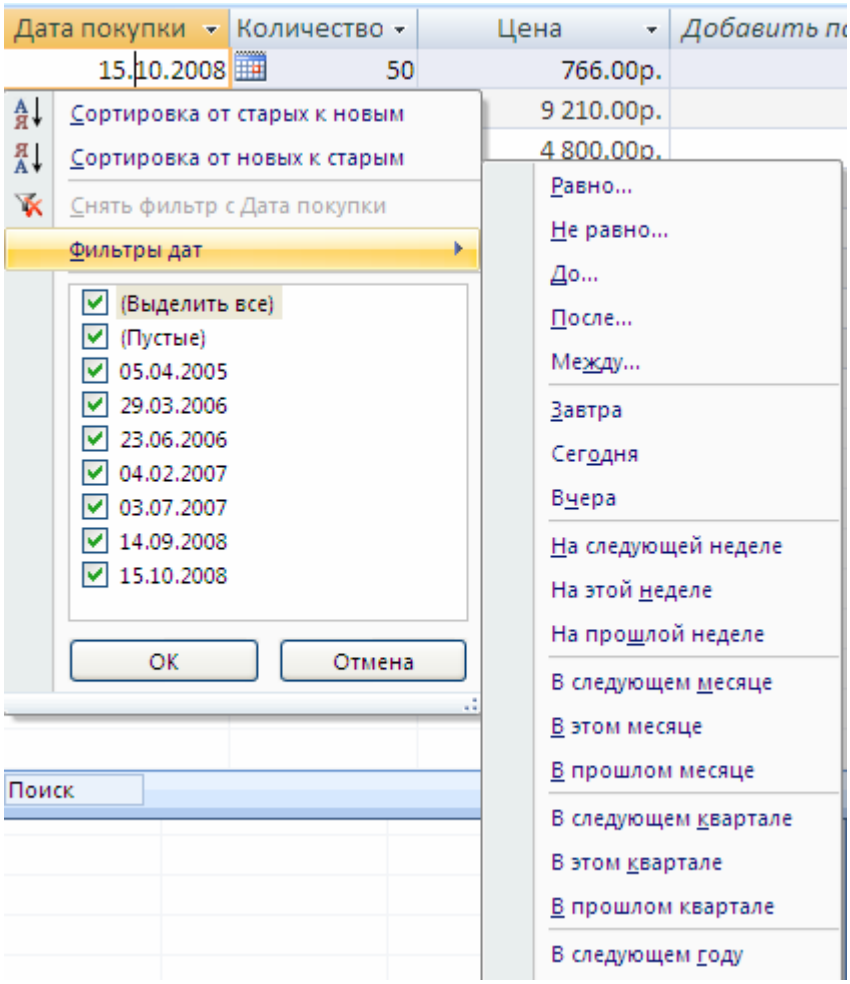



Рисунок 2.6 – Примеры фильтров по диапазону значений

Использование обычного фильтра осуществляется по следующим этапам:

- открыть таблицу, запрос, форму или отчет в режиме таблицы, формы, отчета или макета;

- убедиться, что представление еще не отфильтровано; в строке выбора записи проверить наличие значка **Без фильтра** или затененного значка **Нет фильтра**, чтобы снять все фильтры для определенного объекта, на вкладке **Главная** в группе «Сортировка и фильтр» нажать кнопку **Дополнительно** и выбрать команду **Очистить все фильтры**;

- щелкнуть в любом месте столбца или элемента управления, соответствующего первому полю, к которому следует применить фильтр;

- на вкладке **Главная** в группе «Сортировка и фильтрация» щелкнуть **Фильтр**;

- выполнить одно из следующих действий:

а) чтобы применить обычный фильтр, выделить пункт **Текстовые** (**Числовые**, **Даты**) **фильтры** и выберите нужный фильтр, для фильтров **Равно** и **Между** потребуется ввести необходимые значения, для формирования условий можно использовать подстановочные символы (таблица 2.1);

б) чтобы применить фильтр на основе значений поля, снять флажки возле значений, для которых не следует применять фильтр, и затем нажать

кнопку **ОК**; если список значений слишком велик, а фильтрации подлежит только одно или несколько из них, сначала следует снять флажок **(Выделить все)**, а затем выбрать нужные значения;

- для фильтрации пустых значений (пустое значение означает отсутствие данных) в текстовых и числовых полях, а также в полях дат, в списке с флажками снять флажок **(Выделить все)**, а затем установить флажок возле значения **(Пустые)**.

Для каждого фильтруемого поля повторяются указанные действия.

### 2.1.3.4 Команда **Изменить фильтр**

При выполнении команды **Изменить фильтр**, вызвать которую можно щелкнув **Дополнительно** на вкладке **Главная** в группе «Сортировка и фильтрация», Access отображает окно фильтра, показанное на рисунке 2.7. Это окно содержит одну запись из таблицы, расположенную так же, как и в режиме таблицы. В нижней части окна находятся ярлычки «Найти» и «Или». Если установить указатель в какой-либо из полей таблицы, появится кнопка раскрытия списка. При нажатии на эту кнопку открывается список всех возможных значений, которые встречаются в этом поле. Если необходимо сформировать выражение для критерия, то используются термы сравнения (< >, >, <, >=, <=). Для осуществления фильтрации нажать кнопку **Применить фильтр** на вкладке **Главная** в группе «Сортировка и фильтрация».

Если значения вводятся в несколько полей, то между ними будет установлено логическое отношение типа «И». Чтобы установить для условий отбора отношение «ИЛИ», надо щёлкнуть на закладке «ИЛИ» в нижней части этого окна для того, чтобы вывести на экран новый образец записи, и заполнить для него условия отбора. В данном случае при вводе условий Access добавляет каждое новое условие отбора в отношении «ИЛИ».

### 2.1.3.5 Расширенный фильтр

Иногда может потребоваться применить фильтр, отсутствующий в списке обычных фильтров, для этого используется команда **Расширенный фильтр**. На вкладке **Главная** в группе «Сортировка и фильтрация» щелкнуть **Дополнительно**, а затем выбрать команду **Расширенный фильтр** из раскрывающе-

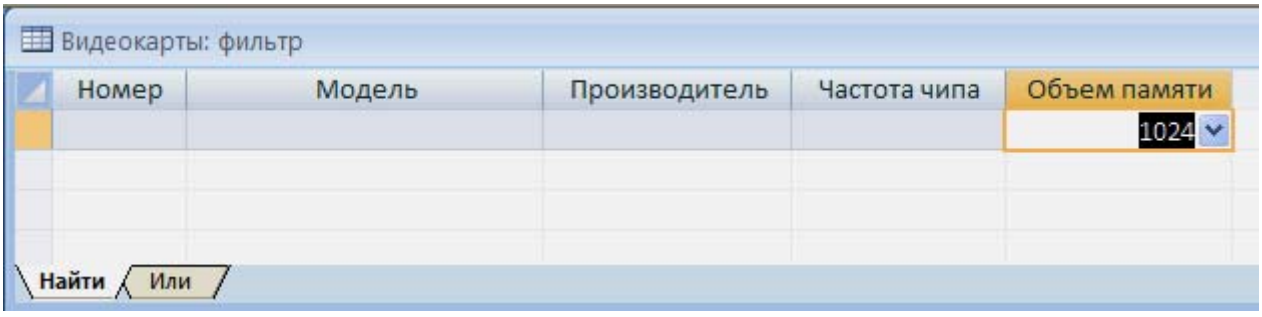



Рисунок 2.7 – Окно «Фильтр»

гося списка. Откроется окно диалога для установки критерия расширенного фильтра, аналогично представленному на рисунке 2.8. В верхней части окна расположен список полей фильтруемой таблицы, в нижней - бланк формирования выражения.

Необходимо добавить поля в сетку конструктора и указать условия отбора для каждого поля в строке **Условия**. Условия применяются в виде набора, и отображаются только записи, которые соответствуют всем условиям в строке **Условия отбора**. Чтобы указать альтернативные условия для отдельного поля, ввести первое условие в строке **Условия отбора**, второе условие в строке «**или»** и так далее.

Чтобы увидеть отфильтрованные строки, необходимо нажать кнопку **Применить фильтр** .

По сравнению с командой «Изменить фильтр» расширенный фильтр позволяет проводить не только фильтрацию, но одновременно и сортировку по возрастанию или убыванию по нескольким полям одновременно. Для этого используется строка **Сортировка**. Фильтр выводит на экран все поля таблицы и фильтрует только записи. Пользователь перетягивает поля из таблицы в бланк построения, чтобы ввести для них условия отбора под ними, но это не влияет на количество выводимых на экран полей.

На вкладке документа **Фильтр** доступны две специальные команды. Если щелкнуть правой кнопкой мыши в любом месте вкладки над сеткой в контекстном меню будут доступны команды **Загрузить из запроса** и **Сохранить как запрос**.

Команда **Загрузить из запроса** служит для загрузки макета выбранного запроса в сетку. В этом случае условия запроса становятся условиями фильтра. Команда **Сохранить как запрос** позволяет сохранить параметры фильтра в качестве нового запроса.

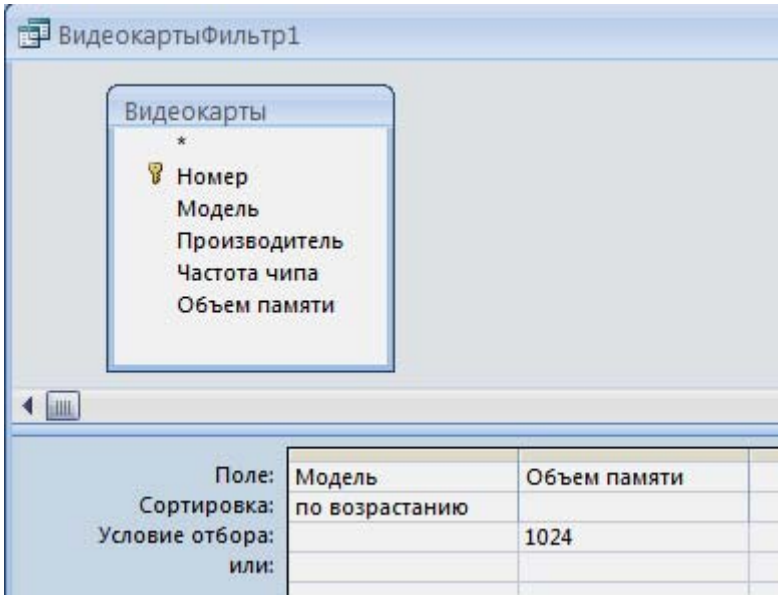



Рисунок 2.8 - Окно диалога для установки критерия расширенного фильтра

### 2.1.4 Выбор данных из таблицы с помощью запросов

2.1.4.1 Понятие о запросах

*Запрос* – объект, позволяющий пользователю получить нужные данные из одной или нескольких таблиц. Запрос представляет собой обращение к данным для получения информации и выполнения действий с данными. Запрос можно использовать для получения ответа на простой вопрос, выполнения расчетов, объединения данных из разных таблиц или даже 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*.

На практике часто требуется из исходной таблицы выделить часть записей, удовлетворяющих определённым критериям и упорядочить выборку. Критерии могут определяться сочетанием ряда условий. Запрос на выборку служит для создания подмножеств данных, которые можно использовать для получения ответов на определенные вопросы. При помощи такого запроса можно передавать данные в другие объекты базы данных. После создания запроса на выборку его можно использовать по мере необходимости. Таблицы или запросы, используемые для получения данных, называются *источниками записей*.

Создание простых запросов на выборку при использовании мастера и при работе в режиме конструктора происходит одинаково. Для этого следует выбрать источник записей и поля, которые требуется включить в запрос. При необходимости можно задать условия для уточнения результатов запроса.

В Access для формирования запросов используется конструктор запросов и инструкция **SELECT** языка Access.

Конструктор запросов позволяет создавать «запрос по образцу», являющимся интерактивным средством для выбора данных из одной или нескольких таблиц. При формировании запроса необходимо указать критерии выборки записей в исходной таблице. При этом вместо того, чтобы печатать предложение на специальном языке, необходимо заполнить бланк запроса QBE (Query By Example – язык запросов по образцу), который располагается в окне конструктора запроса. Метод формирования запроса путём заполнения бланка прост для изучения и понимания. Он способствует эффективному использованию возможностей MS Access пользователями, имеющими даже минимальные навыки работы с приложением или не имеющими их вовсе.

2.1.4.2 Создание запросов с помощью мастера

Для создания запросов с помощью мастера необходимо выполнить следующие действия:

- на **Ленте** на вкладке **Создание** в группе «Другие» щелкнуть **Мастер запросов** (рисунок 2.9);
- в диалоговом окне **Новый запрос** выбрать вариант **Простой запрос** и нажать кнопку **ОК**;

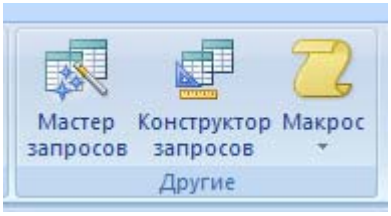



Рисунок 2.9 – Группа «Другие» на вкладке **Создание**

- в группе **Таблицы и запросы** выбрать таблицу, содержащую нужные данные; в качестве источника данных можно использовать и другой запрос;

- в группе **Доступные поля** дважды щелкнуть поля, которые войдут в формируемый запрос, при этом они добавляются в список **Выбранные поля** (рисунок 2.10); если необходимо выбрать все поля исходной таблицы, то нужно нажать на клавишу **>>**, если по ошибке были перенесены несколько лишних полей – не надо начинать всё сначала, можно дважды щелкнуть на ненужном поле, или воспользоваться клавишей <; порядок полей в бланке запроса определяет порядок появления их в результирующей таблице; для того, чтобы изменить расположение поля в этом списке, нужно навести на необходимый столбец и, зажав левую клавишу мыши, перетащить его;

- затем нажать кнопку **Далее**;

- в следующем диалоговом окне выбрать подробный или итоговый отчет и нажать кнопку **Далее**;

- в последнем диалоговом окне задать запросу имя, выбрать действие **Открыть запрос для просмотра данных** или **Изменить макет** и нажать кнопку **Готово**. Если выбрать действие **Изменить макет**, то откроется конструктор запросов, в котором можно выставить условия для отбора записей, а если **Открыть запрос для просмотра данных** - то выполнится одноименная операция.

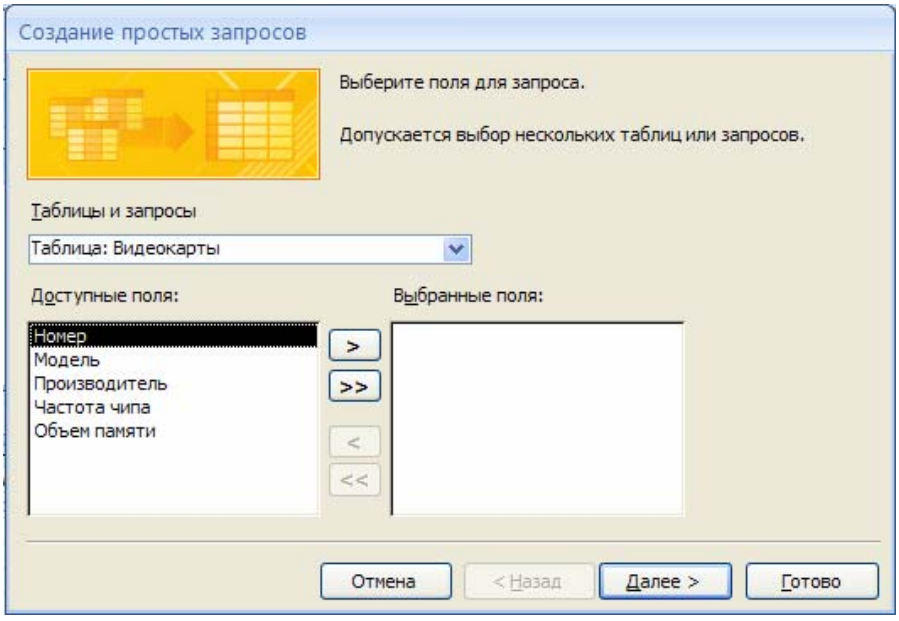



Рисунок 2.10 – Создание простого запроса

В результат выполнения запроса включаются все записи, но при этом отображаются только поля, указанные в запросе. Если закрыть запрос, то он сохраняется автоматически.

2.1.4.3 Конструктор запросов

Для вызова конструктора запросов необходимо на **Ленте** на вкладке **Создание** в группе «Другие» щелкнуть кнопку **Конструктор запросов**. Будет открыт конструктор запросов с открытой вкладкой **Конструктор**, и появится диалоговое окно **Добавление таблицы**. При помощи этого окна можно создать списки полей для добавления их в окно запроса. Эти списки полей могут базироваться как на таблице, так и на существующем запросе, и любые из этих полей могут быть включены в новый запрос. На вкладке **Конструктор** располагаются группы «Результаты», «Тип запроса», «Настройка запроса» и «Показать или скрыть», которые содержат контекстные инструменты для работы с запросами. Язык QBE, используемый в окне конструктора запросов, обеспечивает высокую наглядность и не требует указания алгоритма выполнения операции.

После выбора таблицы или ранее созданного запроса, нажать кнопку **Добавить**, а затем - кнопку **Закрыть**. Таблица будет отображена в окне в верхней части бланка запроса. В этом окне перечислены все поля таблицы. На рисунке 2.11 показан пример построения запроса в конструкторе.

Если в конструкторе запросов дважды щелкнуть звездочку (*) в выбранной таблице, то это обеспечит отображение запросом всех полей из записей, которые он возвращает. Поля можно добавлять перетаскиванием или двойным щелчком. Поле может участвовать в формировании запроса, но не выводится на экран, для этого необходимо снять флажок в строке **Вывод на экран**.

В строке **Условие отбора** и последующие за ней строки вводятся заданные условия. При этом необходимо использовать логические операторы и кавычки. Для запуска запроса на вкладке **Конструктор** в группе «Результаты» выбирается команда **Выполнить** ( ).

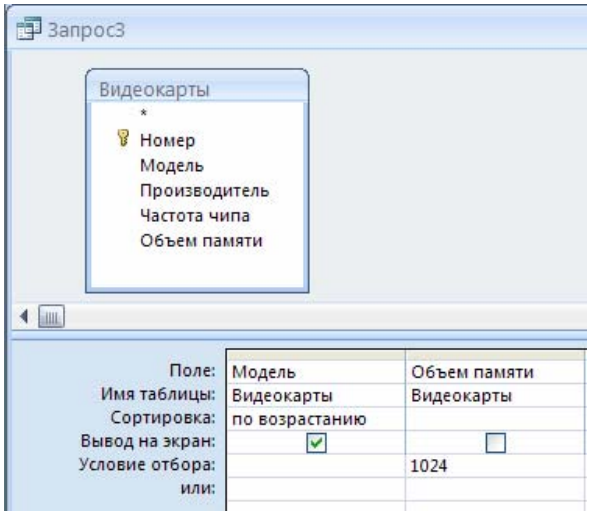



Рисунок 2.11 – Пример построения запроса в режиме конструктора

Если требуется точное несовпадение значений одного из полей, то для установки таких значений используется оператор **Not** или **< >**, которые печатаются перед сравниваемым значением.

При выборе записей по условию неточного совпадения значений используется оператор **Like**. Данный оператор позволяет найти требуемые записи, зная лишь приблизительное написание величины. Кроме того, оператор **Like** используется совместно с подстановочными символами. В таблице 2.1 приведены подстановочные символы, которые могут использоваться с оператором **Like**, а также количество цифр или символов, которым они соответствуют. Группа из одного или более символов, заключённая в квадратные скобки, используется для установления совпадения с одним символом выражения и может содержать любые символы. Для задания диапазона значений в окне конструктора запросов используются операторы **>** (больше), **>=** (не менее), **<** (меньше), **<=** (не более) и **Between**, которые можно использовать с текстовыми и цифровыми полями, а также полями дат.

Удобным средством для создания запросов, является команда **Построитель**, которая вызывается путем нажатия на пиктограмму  , располагающуюся на **Ленте** в режиме конструктора запросов на вкладке **Конструктор** в группе «Настройка запроса».

Например, в таблице «Запрос» необходимо получить сведения о товарах, имеющих код, который находится между 10 и 20. Для этого в построителе выражений надо сформировать условие, как показано на рисунке 2.12. В условиях отбора значение даты необходимо выделять с обоих сторон символом #. При задании диапазона строковых данных кроме операторов сравнения можно использовать и оператор **Like**. Например, для получения списка наименований, названия которых начинаются с В по Р, в **Условие отбора** поля «Наименование» надо ввести условие отбора: **Like <<[В-Р]* >>**.

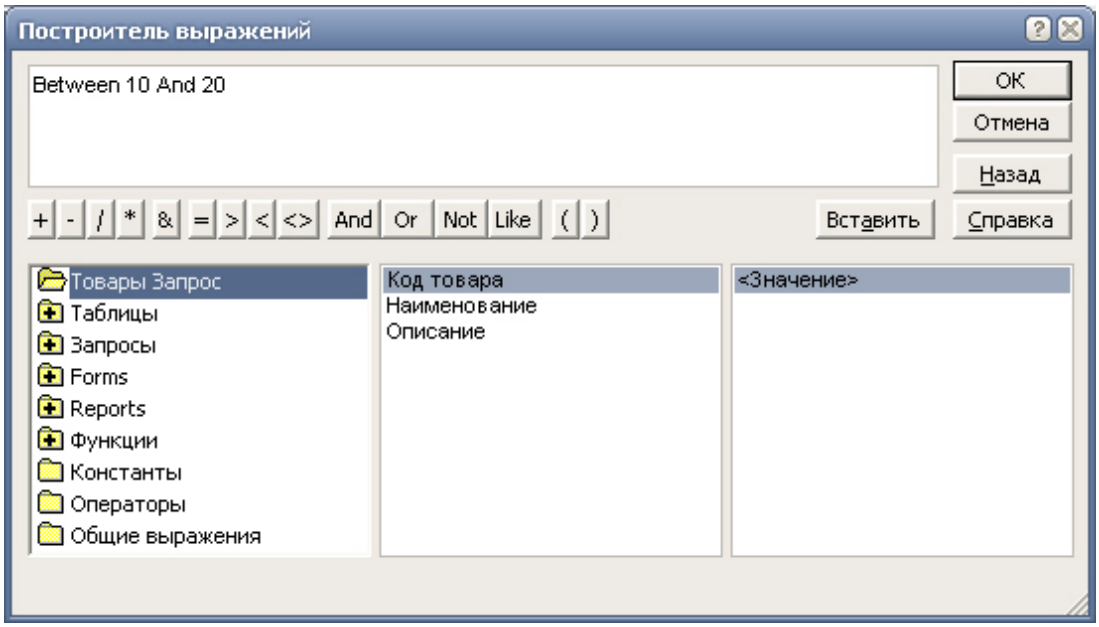



Рисунок 2.12 - Окно построителя выражений

При объединении критериев нескольких полей используются логические «И» и «ИЛИ». Для задания И – выражения задаётся условие в строке **Условие отбора** для каждого из полей, образующих критерий. При формировании ИЛИ – выражения каждое из условий выбора, образующих критерий, должно располагаться на отдельной строке бланка запроса.

### 2.2 Задание на выполнение работы

2.2.1 Запустить MS Access, затем базу данных, содержащую таблицы «Видеокарты», «Учет программного обеспечения» и«Заказы», являющиеся результатом выполнения лабораторной работы 1.

2.2.2 В таблице «Видеокарты» осуществить поиск тех моделей, которые имеют объем памяти 1024 Мб.

2.2.3 В таблице «Учет программного обеспечения» осуществить поиск программного обеспечения, имеющего название, начинающееся с «М»; программного обеспечения, установленного в 2008 году.

2.2.4 В таблице «Учет программного обеспечения» произвести поиск и замену даты установки «28.12.08» на «15.02.09».

2.2.5 Осуществить в таблице «Видеокарты» сортировку по убыванию значений объема памяти; по возрастанию названия производителя и частоте чипа.

2.2.6 В таблице «Видеокарты» создать фильтр по выделенному значению «1024» в поле «Объем памяти».

2.2.7 В таблице «Видеокарты» вывести на экран все записи кроме тех, которые включают в поле «Объем памяти» значение «1024».

2.2.8 В таблице «Видеокарты» вывести на экран с помощью обычного фильтра записи о видеокартах, которые содержат в названии модели «GeForce» и имеют объем памяти 1024 Мб.

2.2.9 В таблице «Заказы» с помощью обычного фильтра вывести на экран записи о заказах, ценой более 3000.

2.2.10 В таблице «Учет программного обеспечения» с помощью расширенного фильтра осуществить выбор записей об операционных системах Microsoft и выполнить сортировку по убыванию даты установки.

2.2.11 С помощью мастера запросов сформировать из таблицы «Видеокарты» новую таблицу с полями «Модель», «Производитель», «Объем памяти» с отсортированными по возрастанию названиями моделей. Удалить из созданного в пункте 2.2.11 запроса поле «Объем памяти».

2.2.12 Создать в конструкторе запросы к таблице «Учет программного обеспечения» по критериям, указанным в п. 2.2.3; а также получить сведения о программном обеспечении, установленном с 01.12.2008 по 31.01.2009.

2.2.13 Оформить отчет по лабораторной работе.

### 2.3 Содержание отчёта

2.3.1 Название работы.

2.3.2 Цель работы.

2.3.3 Перечень команд, используемых при выполнении лабораторной работы, с указанием их назначения.

## 2.4 Тесты и контрольные вопросы

2.4.1 На какой вкладке **Ленты** находится группа «Найти»:

а) **Создание**; б) **Главная**;

в) **Работа с базами данных**; г) **Режим таблицы**?

2.4.2 Какой подстановочный символ соответствует любому текстовому символу?

а) *; б) ?; в) [!]; г) #.

2.4.3 Подстановочный символ # соответствует:

а) любой цифре;

б) любому текстовому символу;

в) любому символу из диапазона;

г) любому количеству букв, цифр или других символов.

2.4.4 Назначением фильтрации является:

а) выделение записей по указанному критерию;

б) расположение записей в определенном порядке;

в) представление данных по указанному критерию;

г) сортировка записей по указанному критерию;

д) выбор записей по указанному критерию.

2.4.5 На какой вкладке **Ленты** находится команда **Сортировка**:

а) **Создание**; б) **Работа с базами данных**;

в) **Главная**; г) **Режим таблицы**?

2.4.6 Что является результатом команды **Применить фильтр**:

а) таблицы, определяемые заданным критерием;

б) записи, определяемые заданным критерием;

в) поля, определяемые заданным критерием;

г) ячейки, определяемые заданным критерием?

2.4.7 На какой вкладке **Ленты** находятся команды создания запросов:

а) **Создание**; б) **Работа с базами данных**;

в) **Главная**; г) **Режим таблицы**?

2.4.8 Что такое запрос в Access:

а) объект базы данных, в который добавляются элементы управления, реагирующие на действия пользователей или служащие для ввода, отображения и изменения данных в полях;

б) объект базы данных, в котором данные хранятся в виде записей (строк) и полей (столбцов);

в) объект базы данных, предназначенный для вывода на печать данных, организованных и отформатированных в соответствии с требованиями пользователя;

г) объект базы данных, позволяющий пользователю получить нужные данные из одной или нескольких таблиц?

2.4.9 Какая команда используется для формирования сложного критерия:

а) **Построитель**; б) **Формирователь**;

в) **Запрос**; г) **Выбор**?

2.4.10 Если в конструкторе запросов дважды щелкнуть звездочку (*) в выбранной таблице, то это обеспечит отображение запросом:

а) всех полей из записей, которые он возвращает;

б) всех записей, которые он возвращает;

в) всех критериев, которые он возвращает;

г) всех условий, которые он возвращает.

2.4.11Какой командой осуществляется поиск записей?

2.4.12 Что вводится в окно диалога «Поиск»?

2.4.13 Что используется для поиска данных, точное значение которых неизвестно?

2.4.14 Каким образом осуществляется поиск и замена данных?

2.4.15 Что включает в себя сортировка данных?

2.4.16 Назовите способы проведения сортировки.

2.4.17 Для чего в базе данных используется фильтр?

2.4.18 Как можно создать фильтр?

2.4.19 Какие виды фильтров имеются в Access?

2.4.20 Как выполняется создание фильтра по выделенному фрагменту?

2.4.21 Недостатки фильтра по выделенному фрагменту.

2.4.22 Как создаётся обычный фильтр?

2.4.23 Как формируется выражение для критерия фильтра?

2.4.24 Что включает в себя окно «Фильтр»?

2.4.25 Как задать критерий с помощью расширенного фильтра?

2.4.26 Назовите отличия расширенного фильтра от обычного.

2.4.27 Что включает в себя окно диалога для установки критерия расширенного фильтра?

2.4.28 Что называется запросом?

2.4.29 Как создаётся запрос с помощью мастера?

2.4.30 Что включает в себя окно конструктора запросов?

2.4.31 Каким образом можно выбрать поля таблицы для создания запроса?

2.4.32 Как удалить лишнее поле из запроса?

2.4.33 Как изменить порядок расположения полей в запросе?

2.4.34 Каким образом осуществляется запуск запроса?

2.4.35 Как сохранить запрос?

2.4.36 Как задаются условия для выбора записей?

2.4.37 Каким образом строится условие при точном несовпадении значений одного из полей?

2.4.38 Какой оператор используется при выборе записей по условию неточного совпадения значений?

2.4.39 Для чего используются подстановочные символы в запросах?

2.4.40 Что включает в себя окно построителя выражений?

# 3 Создание и использование форм для ввода и редактирования данных в Microsoft Office Access 2007

Целью работы является приобретение навыков создания пользовательских форм для ввода и редактирования данных на экране при работе с реляционными базами данных в Microsoft Office Access 2007.

## 3.1 Общие положения

### 3.1.1 Создание формы с помощью инструмента «Форма»

В Access данные можно просматривать непосредственно в таблицах. Однако это не всегда удобно, поскольку иногда невозможно вывести на экран все поля одной записи одновременно. Для облегчения работы пользователей служат формы.

*Форма* - объект базы данных Access, в который добавляются элементы управления, реагирующие на действия пользователей или служащие для ввода, отображения и изменения данных в полях.

Формы могут применяться для управления доступом к данным: с их помощью можно определять, какие поля или строки данных будут отображаться. Например, некоторым пользователям достаточно видеть лишь несколько полей большой таблицы. Если предоставить им форму, содержащую только нужные им поля, это облегчит для них использование базы данных. Для автоматизации часто выполняемых действий в форму можно добавить кнопки и другие функциональные элементы. Формы можно рассматривать как окна, через которые пользователи могут просматривать и изменять базу данных. Рационально построенная форма ускоряет работу с базой данных, поскольку пользователям не требуется искать то, что им нужно. Внешне привлекательная форма делает работу с базой данных более приятной и эффективной, кроме того, она может помочь в предотвращении неверного ввода данных. В Microsoft Office Access 2007 предусмотрены новые средства, помогающие быстро создавать формы, а также новые типы форм и функциональные возможности, благодаря которым база данных становится более практичной. Также можно вывести формы на печать – это довольно ценное приложение к распечатанным отчетам, но в первую очередь формы предназначены для представления данных на экране.

Самым легким способом работы является использование форм, созданных при помощи инструмента «Форма». Чтобы создать такую форму, необходимо:

- открыть окно базы данных;
- в окне переходов базы данных перейти на вкладку «Таблицы»;
- установить указатель на таблицу, для которой создается форма;
- выполнить команду **Форма** на **Ленте** на вкладке **Создание** в группе «Форма».

На экране появится готовая к использованию форма (рисунок 3.1). В эту форму включены все поля таблицы. Их названия расположены вертикально в том же порядке, в каком они находятся в таблице. Если полей слишком много,

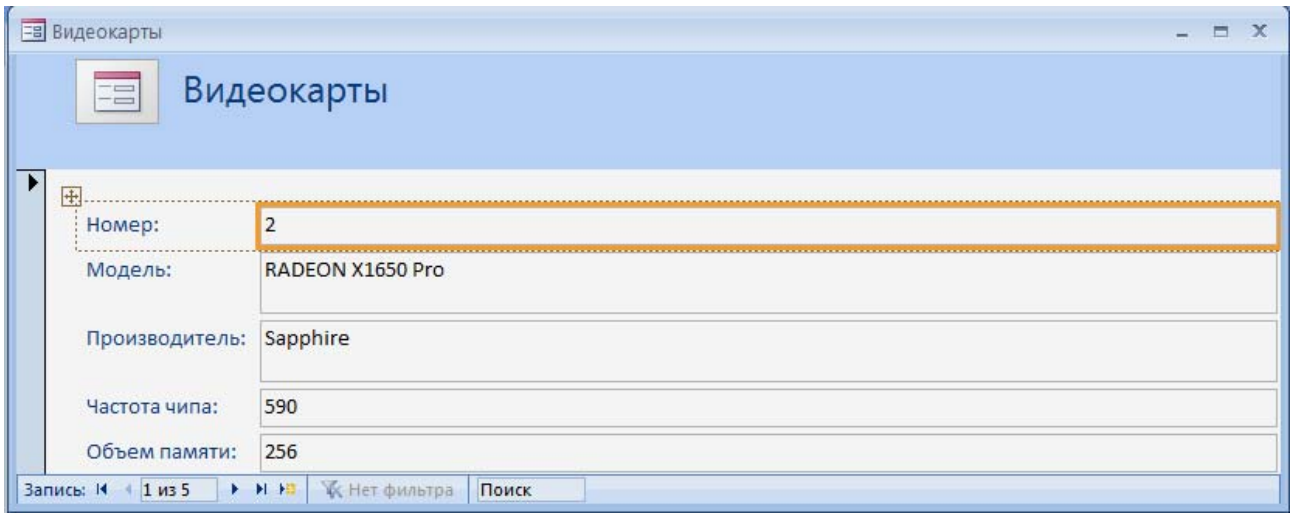



Рисунок 3.1 – Форма, созданная на основе таблицы «Видеокарты»

то Access располагает их в два столбца. Справа от названия каждого поля отображается значение его первой записи в таблице.

Созданная форма изначально включена в режиме макета формы. В этом режиме возможно перемещать/удалять/добавлять поля таблицы. Для начала работы с формой, необходимо переключиться в режим **Форма**, для этого щелкнуть правой клавишей мыши по пиктограмме формы, и выбрать соответствующий режим.

Чтобы увидеть следующую запись, можно нажать клавишу **Page Down**. Чтобы вернуться на предыдущую запись, нажать **Page Up**. Для перехода, как в соседние, так и в отдаленные записи, используются кнопки перемещения, расположенные в самом низу формы. Назначение этих кнопок приведено в таблице 3.1. Кроме того, для перехода к записи с определенным номером можно воспользоваться полем ввода **Запись**. Для этого необходимо выделить значение номера текущей записи в поле ввода, ввести номер требуемой записи и нажать **Enter**.

При просмотре данных в режиме формы может возникнуть необходимость просмотра данных в табличном виде, так как в табличном виде легче сравнивать данные. Для перехода из формы в режим таблицы необходимо заново открыть исходную таблицу.

Для изменения формата формы нужно использовать **Конструктор форм**.

Таблица 3.1 – Кнопки перемещения и выполняемые ими действия

| Кнопка | Выполняемые действия |
|---|---|
|  | Переход на первую запись |
|  | Переход на одну запись назад |
|  | Переход на одну запись вперед |
|  | Переход на последнюю запись |
|  | Переход на чистую страницу, где можно ввести новую запись |

### 3.1.2 Создание формы при помощи инструмента «Разделенная форма»

*Разделенная форма* - это новая возможность в Microsoft Office Access 2007, позволяющая одновременно отображать данные в двух представлениях - в режиме формы и в режиме таблицы.

Эти два представления связаны с одним и тем же источником 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 (при условии, что источник записей допускает обновление, а параметры формы не запрещают такие действия). Работа с разделенной формой дает преимущества обоих типов формы в одной форме. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы. Пример разделенной формы показан на рисунке 3.2.

Чтобы создать разделенную форму при помощи инструмента «Разделенная форма», выполняются следующие действия:

- в области переходов щелкнуть таблицу или запрос с данными, которые должны отображаться в форме, или открыть таблицу или запрос в режиме таблицы;

- на **Ленте** на вкладке **Создание** в группе «Формы» щелкнуть **Разделенная форма** (  ).

Access создаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

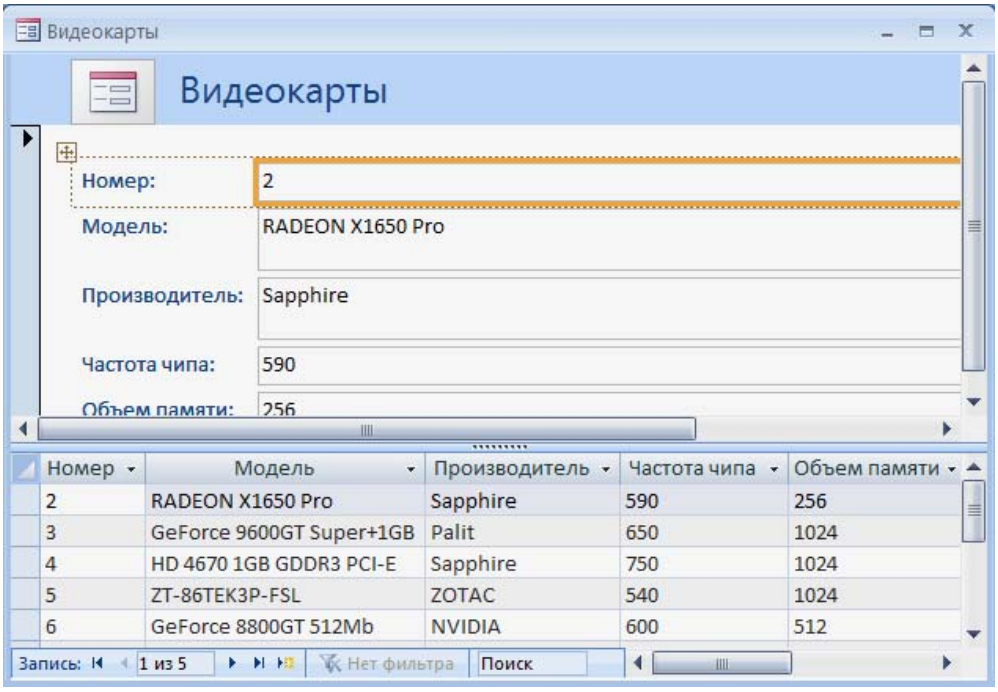



Рисунок 3.2 – Разделенная форма, созданная на основе таблицы «Видеокарты»

### 3.1.3 Использование мастера для создания форм

Одним из способов создания форм, предлагаемых Access, является Мастер форм. По сравнению с автоматически созданными формы, созданные с помощью мастера, более разнообразны по стилю оформления, содержат заданные поля. Кроме того, можно указать способ группировки и сортировки данных, а также включить в форму поля из нескольких таблиц или запросов, при условии, что заранее заданы отношения между этими таблицами и запросами.

Чтобы запустить **Мастер форм**, необходимо выполнить следующие действия: на вкладке **Создание** в группе «Формы» щелкнуть **Другие формы**, а затем в списке щелкнуть пункт **Мастер форм** (  ).

После нажатия кнопки раскрытия списка **Таблицы и Запросы** необходимо выбрать из списка таблиц базы данных таблицу или запрос, для которой создаётся форма. При этом в списке **Допустимые поля** появится перечень всех полей выбранной таблицы. Надо перенести из данного перечня в список **Выбранные поля** те поля, которые необходимо поместить в создаваемую форму. Для выбора полей из списка допустимых полей надо пометить их, а затем перенести в список выбранных полей, нажимая мышью кнопку с одиночной стрелкой, направленной вправо. Если надо включить все поля, то нажимается кнопка с двойной стрелкой. Таким же образом исключаются поля, выбранные из списка, пользуясь кнопками со стрелками, направленными влево.

Мастер формы позволяет создавать формы, используя поля не только одной таблицы, но и из нескольких связанных. В этом случае после выбора полей из первой таблицы необходимо выбрать из списка таблиц базы данных вторую таблицу и перенести требуемые поля в список **Выбранные поля**.

Завершив формирование списка полей формы, необходимо нажать кнопку **Далее,** чтобы перейти в следующее окно, которое позволяет задать внешний вид формы. Выбрав нужную опцию, надо нажать кнопку **Далее**.

Затем мастер позволяет выбрать стиль формы. Из списка, содержащего вариант стилей, выбирается стиль, который в наибольшей степени отвечает требованиям. Слева можно посмотреть, что представляет собой выбранный стиль. После установки стиля, необходимо нажать кнопку **Далее.** На экране откроется последнее окно диалога. Необходимо ввести в нем имя создаваемой формы. Затем указывается вариант дальнейшей работы установкой одной из опций:

- открытие формы для просмотра или ввода данных (сохраняет созданную форму и открывает ее для просмотра или ввода данных);
- изменение макета формы (сохраняет созданную форму и открывает ее в конструкторе форм для модификации).

После установки требуемых опций необходимо нажать **Готово** для завершения создания формы с помощью Мастера.

Все этапы работы с Мастером форм показаны на рисунке 3.3, где 1 – выбор полей для формы; 2 – изменение внешнего вида формы; 3 – применение требуемого стиля; 4 – задание имени формы и выбор действия после закрытия Мастера.

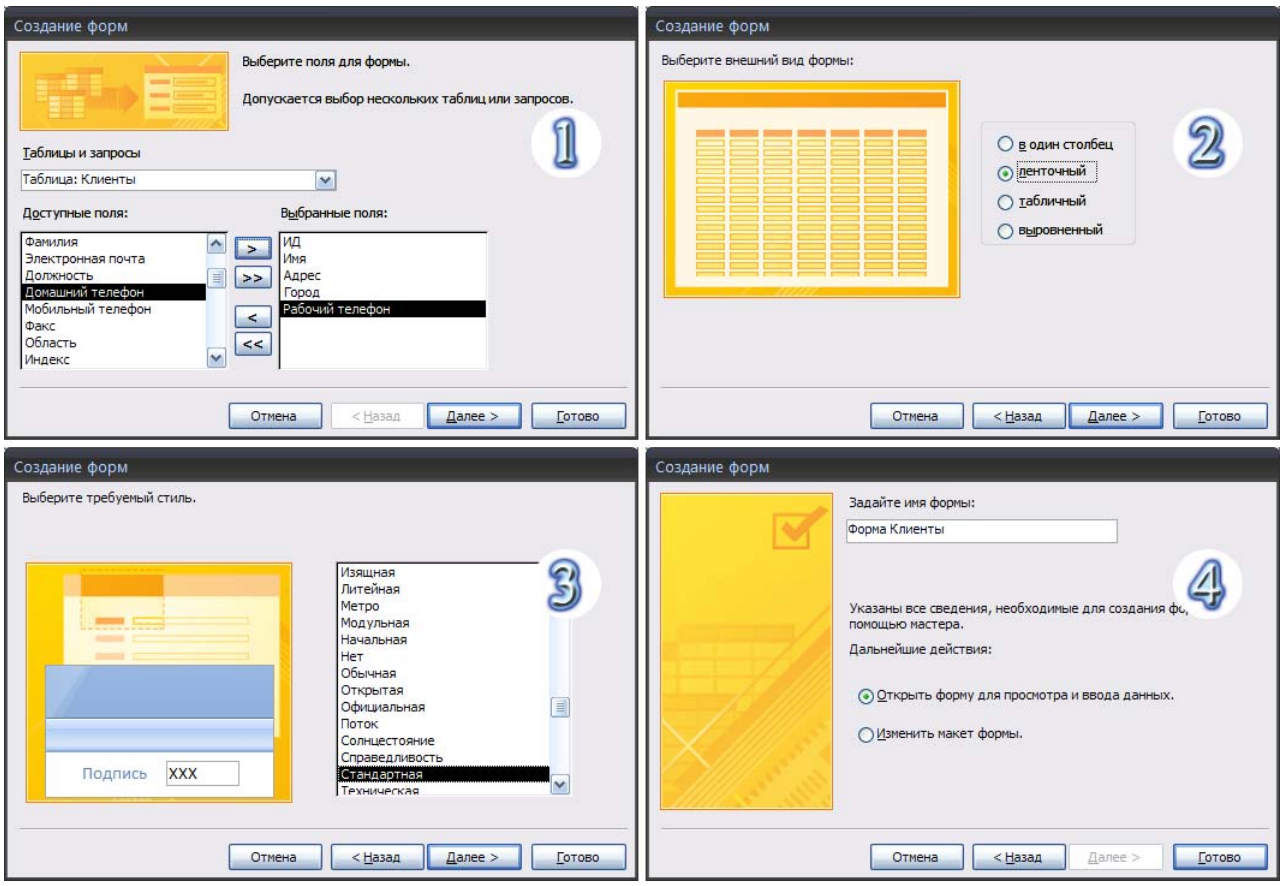



Рисунок 3.3 – Этапы работы с Мастером форм

### 3.1.4 Создание формы в конструкторе форм

3.1.4.1 Окно конструктора форм

Автоматически создаваемые формы и мастер форм позволяют быстро создать экранную форму для ввода и просмотра данных. Но для таблиц, имеющих много полей, эти формы не позволяют вместить на экран одновременно все поля таблицы. С таблицей легче работать, если реорганизовать все поля в таблицы так, чтобы можно было просмотреть их одновременно. Для создания сложных и более удобных для конечного пользователя форм средств простых форм и мастера недостаточно. Создавать формы любой степени сложности можно с помощью конструктора форм.

Любая форма в Access состоит из объектов формы, которые имеют характерные для них свойства. Для каждого объекта можно определить действия, выполняемые при наступлении определенных событий. Процесс создания формы состоит в размещении объектов в форме и определении для них свойств, связанных с ними событий и выполняемых действий.

Создать форму можно, выполнив следующие действия:

- открыть базу данных;
- в **области переходов** базы данных нажать на таблицу, на основе которой необходимо построить форму;

- нажать кнопку **Конструктор форм** на **Ленте** в группе «Формы» на закладке **Создание**.

На экране откроется конструктор форм. При этом на **Ленте** появляются элементы управления, позволяющие изменять внешний вид формы (рисунок 3.4). Установки линий сетки, заливки, шрифта, можно изменить соответственно в группах «Сетка», «Шрифт» на вкладке **Конструктор**, а также можно изменить оформление в группе «Элементы управления». Получить доступ к свойствам объектов таблицы, просмотреть код или добавить поля можно в группе «Сервис». Дополнительные настройки по оформлению и стилю формы доступны на **Ленте** во вкладке **Упорядочить**.

Конструктор форм состоит из вертикальной и горизонтальной линеек, области данных с вертикальными и горизонтальными линиями – это сетка формы, по ней выравниваются объекты. Помимо этого, форма может содержать область заголовка, примечания, верхний и нижний колонтитулы. Для добавления этих областей используются команды **Заголовок/примечание формы** и **Колонтитулы страницы** в группе «Отображение» на **Ленте** во вкладке **Упорядочить**, либо одноименные команды контекстного меню **области данных**, которое вызывается нажатием правой клавиши мыши на свободном пространстве. После выполнения этих команд рабочая область конструктора форм принимает вид, представленный на рисунке 3.5. Для изменения размера области установить указатель мыши на верхнюю часть границы между областями. Когда курсор мыши примет вид двусторонней стрелки с прямоугольником посередине, надо нажать кнопку мыши и переместить границу, не отпуская кнопку.

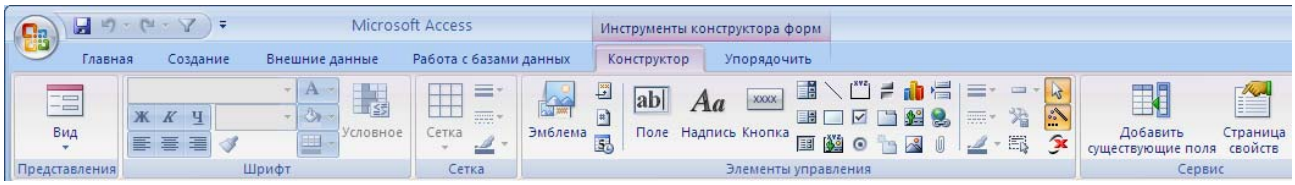



Рисунок 3.4 – Инструменты конструктора форм

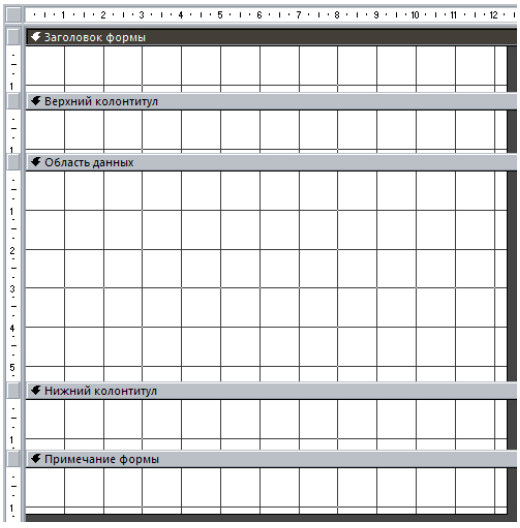



Рисунок 3.5 – Рабочая область конструктора формы

Данные, размещаемые в заголовке, в области данных и в области примечания, отображаются в форме. Области верхнего и нижнего колонтитула отображаются только при печати.

Краткое описание кнопок группы «Элементы управления» вкладки **Конструктор**, используемых для размещения объектов в форме, приведено в таблице 3.2.

Таблица 3.2 – Инструменты конструктора форм

| Кнопка | Наименование | Назначение |
|---|---|---|
| 1 | 2 | 3 |
|  | Выбор объектов | Осуществляет выделение элемента управления, раздела или формы |
|  | Построители элементов | Включает/выключает мастера создания элементов управления (поля со списком, группы параметров, кнопки, диаграммы и подчиненной формы) |
|  | Надпись | Позволяет разместить в форме текст в дополнении к размещенному по умолчанию Access |
|  | Поле | Осуществляет:<br>- отображение, ввод или изменение данных, содержащихся в источнике данных формы;<br>- ввод результатов вычислений;<br>- прием данных при их вводе пользователем |
|  | Группа переключателей | Используется для размещения в группе флажков, переключателей и выключателей, представляющих набор альтернативных значений |
|  | Выключатель | Используется в качестве:<br>- отдельного элемента управления, связанного с логическим полем;<br>- свободного элемента управления, принимающего действие пользователя в специальном окне диалога;<br>- компонента группы параметров, в которой отображаются значения для выбора |
|  | Конец страницы | Указывает начало нового экрана формы |
|  | Линия | Размещает линию для отделения логически связанных данных |
|  | Переключатель | Используется аналогично элементу управления «Выключатель» |
|  | Флажок | Используется аналогично элементу управления «Выключатель» |
|  | Поле со списком | Составной элемент управления, объединяющий поле и раскрывающийся список, для ввода значения в поле исходной таблицы можно непосредственно ввести значение в поле или выбрать его из предопределенного списка |
|  | Список | Создает список, допускающий прокрутку. В режиме формы выбранное из списка значение можно ввести в новую запись или использовать для замены уже существующего значения |

Продолжение таблицы 3.2

| 1 | 2 | 3 |
|---|---|---|
|  | Кнопка | Позволяет осуществить разнообразные действия в форме (например, поиск записей, формирование отчета, установка/снятие фильтра) |
|  | Рисунок | Осуществляет размещение рисунка, не являющегося объектом OLE |
|  | Свободная рамка объекта | Позволяет ввести свободный объект OLE, который остается неизменным при перемещении по записям |
|  | Присоединенная рамка объекта | Позволяет отобразить в форме объект OLE (например, набор рисунков). Предназначен для объектов, сохраненных в базовом источнике записей формы, поэтому при перемещении по записям в форме отображаются разные объекты |
|  | Подчиненная форма/отчет | Позволяет отобразить данные из нескольких источников |
|  | Прямоугольник | Размещает прямоугольник для группировки элементов управления или выделения логически связанных данных |
|  | Дополнительные элементы | |

3.1.4.2 Управление объектами

Все объекты Access характеризуются свойствами, которые можно настроить в соответствии с требованиями. Кроме того, для каждого объекта существуют встроенные события, которые выполняются при наступлении связанных с ними действий.

Для получения доступа к свойствам и событиям объекта необходимо выделить нужный объект и выполнить одно из следующих действий: выбрать команду **Свойства** из контекстного меню или нажать кнопку **Страница свойств** на **Ленте** в группе «Сервис».

В результате на экране появится окно диалога со свойствами и событиями выбранного объекта (рисунок 3.6). В верхней части окна диалога находятся ярлычки с перечнем сгруппированных по типам свойств и событий:

- **Макет** (содержит свойства объекта, связанные с его оформлением);
- **Данные** (содержит свойства объекта, связанные с источником данных);
- **События** (содержит список всех событий объекта);
- **Другие** (в данную вкладку собранны все свойства, не вошедшие во вкладки **Данные**, **Макет** и **События**);
- **Все** (содержит список всех свойств и событий формы в алфавитном порядке).

При размещении объекта в форме устанавливаются принятые по умолчанию значения свойств объекта. Для того, чтобы изменить стандартные установки для свойств какого-либо из объектов, необходимо выбрать объект и открыть

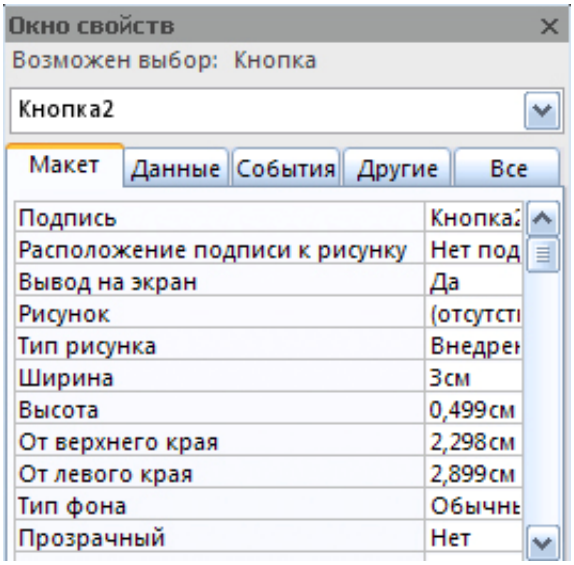



Рисунок 3.6 – Окно свойств элемента «Кнопка»

окно его свойств. Возможны следующие типы значений свойства:

- **свойство математического редактирования** (поле ввода свойства активно, в него можно ввести с клавиатуры требуемое значение);

- **возможны два или более различных вариантов значений свойства** (рядом с полем коррекции появляется кнопка раскрытия списка );

- **возможен выбор свойств с помощью окна настройки** (рядом с полем коррекции появляется кнопка открытия окна надстройки );

- **создание выражения** (рядом с полем коррекции свойства появляется кнопка раскрытия списка, а рядом с ней кнопка открытия окна процедур, макросов или построителя выражений в зависимости от выбранного значения списка ).

В процессе создания формы можно перемещать, удалять или изменять размеры объектов, а также изменять их свойства.

Для управления объектами прежде всего нужно выделить требуемый объект. Для выделения одного объекта достаточно установить указатель мыши на объект и нажать кнопку мыши. Для выделения одновременно нескольких объектов можно воспользоваться любым из следующих способов: нажать клавишу **Shift** и, удерживая ее в нажатом состоянии, нажать мышью все выделяемые объекты или выбрать инструмент «Выбор объектов» на панели элементов ( ); установить указатель мыши на объект и, не отпуская кнопки мыши, переместить рамку выделения так, чтобы внутри нее оказались выделенные объекты.

Для снятия выделения со всех объектов установить указатель мыши вне выделенных объектов и нажать кнопку мыши. Если необходимо снять выделение с отдельных объектов, то, удерживая нажатой клавишу **Shift**, выделить мышью объекты, с которых требуется снять выделение. Выделенный объект имеет маркеры выделения в виде квадратиков, расположенные по углам и серединам сторон. Для изменения размера объекта надо переместить один из маркеров до достижения нужного размера.

Если требуется установить точные размеры объекта, можно воспользоваться альтернативным способом. Для этого необходимо открыть окно свойств объекта и установить требуемые значения высоты и ширины в полях ввода значения свойств **Ширина** и **Высота**.

Для изменения местоположения объекта необходимо выделить его, установив указатель мыши на область выбранного объекта, появляется перекрестье со стрелками. Теперь, удерживая кнопку мыши нажатой, переместить объект в требуемое место.

Поле ввода и надпись являются связанными объектами, то есть они перемещаются вместе. Для перемещения связанных объектов отдельно друг от друга используются метки перемещения (большой квадрат в верхнем левом углу). Когда подводится указатель мыши к метке перемещения, как поля ввода, так и связанной с ними надписи, он превращается в перекрестье со стрелками. Если теперь нажать мышь и, удерживая кнопку нажатой, можно переносить объект, при этом остальные объекты остаются на месте.

Если требуется удалить объект, необходимо выделить его, затем нажать клавишу **Del**.

Для улучшения внешнего вида формы применяется выравнивание выбранных объектов относительно сетки формы или друг друга, а также установка одинаковых размеров объектов. Для этих целей используются команды на закладке **Упорядочить**. Блоки данной закладки содержат набор функций выравнивания и изменения размеров объектов. Чтобы сделать все объекты одинаковыми по размеру, используется группа «Размер», имеющая следующие опции: по размеру данных; по узлам сетки; по самому высокому; по самому низкому; по самому широкому; по самому узкому.

Access обеспечивает средствами точного размещения объектов при использовании сетки. Для этого необходимо выполнить команду **Привязать** из группы «Макет элемента управления». Затем переносить объект при помощи мыши. Во время перемещения контур объекта отображается на горизонтальной и вертикальной линейках.

Во время ввода данных переход от одного объекта к другому при нажатии клавиши **Tab** осуществляется в соответствии с заданным в экранной форме порядком объектов. Для того, чтобы определить порядок обхода объектов в форме, надо выбрать команду **Переходы** из группы «Макет элемента управления». На экране откроется окно диалога «Последовательность перехода». Выделить в окне списка полей перемещаемый объект, нажав мышью кнопку, расположенную слева от названия объекта. Затем нажать кнопку еще раз и переместить ее в требуемое место в списке объектов.

### 3.1.4.3 Общие рекомендации по созданию формы

Процесс по созданию формы может включать в себя все или часть из приведенных ниже процедур:

- размещение текста;
- размещение полей;

- создание управляющих кнопок;
- размещение линий, прямоугольников и рисунков;
- установка цвета объектов формы;
- перемещение объектов формы.

Тщательно разработанная форма, используемая пользователем при вводе, редактировании или просмотре данных, позволяет предотвратить возникновение большого количества ошибок.

При создании форм необходимо придерживаться следующих рекомендаций:

- если пользователи привыкли к использованию стандартных бланков, формы должны выглядеть также, как и эти бланки, чтобы меньше времени тратилось на поиск места нахождения информации;
- для группировки элементов управления использовать линии и прямоугольники;
- не концентрировать элементы управления в какой-либо части формы, чтобы не затруднять чтение информации;
- пояснительный текст формы должен быть максимально информативным и иметь минимальную длину;
- использовать условия контроля правильности ввода данных для предотвращения ввода неверных данных;
- использовать маски ввода для ввода стандартизированной информации;
- для облегчения восприятия чисел использовать форматы ввода.

Для создания формы с помощью конструктора необходимо открыть закладку **Создать** на **Ленте** в группе «Формы» выбрать команду **Конструктор форм**.

Первое, что нужно сделать – это определить свойства самой формы, как объекта. Каждая форма имеет свойства, определяющие расположение ее в основном окне Access, размер, заголовок, стиль и некоторые другие параметры.

Для определения стиля формы выполнить следующие действия:

а) находясь в конструкторе форм, открыть закладку **Упорядочить**, затем нажать на клавишу **Формат**, и выбрать команду **Мастер автоматического сохранения**. На экране откроется окно диалога «Формат», в котором выбирается стиль из предложенного списка, затем нажать кнопку **ОК**;

б) для задания размеров формы необходимо установить указатель в нижний правый угол формы, при этом курсор примет вид двунаправленной стрелки, затем нажать кнопку мыши и установить требуемый размер формы;

в) для настройки остальных параметров открыть окно ее свойств, выполнить команду **Свойства** из группы «Сервис», при этом откроется окно свойств со стандартными значениями свойств формы;

г) перейти на вкладку «Макет» окна свойств, на которой, редактируя свойства формы, можно задать различные параметры окна формы, например, наличие кнопок оконного меню в верхнем правом углу с помощью свойства **Кнопка оконного меню**; наличие кнопок свертывания и развертывания окна с помощью свойства **Кнопки размеров окна**; отображение кнопки закрытие формы с помощью свойства **Кнопка закрытия**; расположение формы в центре окна в

приложении установкой значения свойства **Выравнивание по центру**, равным **Да**; заголовок окна, вводом текста заголовка в поле свойства **Подпись**;

д) для связывания формы с таблицей или запросом в свойстве **Источник записи** вкладки «Данные» из раскрывающегося списка выбирается источник записей, для которого создается форма.

При использовании формы только для просмотра можно установить для свойства **Разрешить изменение** значение **Нет**. Данная установка не позволит пользователю редактировать содержимое элементов управления, связанных с таблицей или запросом. При запрете пользователю добавления новой записи в таблицу или удаления уже существующей необходимо установить для свойства **Разрешить добавление** значения **Нет**. При этом становятся недоступными команды **Новая запись**, **Ввод данных**, **Удалить запись**.

### 3.1.4.4 Размещение текстовой информации и полей ввода

Размещение текста в экранной форме осуществляется с помощью элемента управления формы **Надпись**, который находится в группе «Элементы управления». Под *текстом* понимается любая текстовая информация: заголовки, поясняющая информация.

Для размещения текстов на форме необходимо выполнить следующие действия:

а) выбрать элемент управления формы **Надпись** на **Ленте**;

б) установить указатель мыши на место предполагаемого расположения текстового объекта и ввести текст;

в) закончив ввод текста нажать клавишу **Enter**;

г) выделить созданный объект;

д) используя группы «Шрифт» и «Элементы управления» или окно свойств созданного объекта, задать для него параметры оформления.

Наиболее часто используемыми параметрами оформления являются: **Тип шрифта**; **Размер шрифта**; **Цвет текста**; **Цвет заливки/фона**; **Выравнивание**; **Цвет линии**; **Тип линии**; **Толщина линии**; **Оформление**.

Следующим шагом в создании формы является добавление в нее полей различных типов. Наиболее простым типом поля является поле ввода. Для его размещения в форме необходимо выполнить следующие действия:

- выбрать элемент **Поле** на панели элементов;
- нажать мышью место, в котором предполагается разместить поле, в форме появится связанный объект, состоящий из поля ввода и его надписи;
- выделить поле ввода и открыть для него окно свойств;
- чтобы связать созданное поле с полем таблицы или запроса выбрать свойство **Данные** вкладки «Данные», в поле ввода свойства воспользоваться кнопкой раскрытия списка, и выбрать из списка всех полей открытой таблицы поля, которое надо добавить в форму (этот список содержит все поля таблицы или запроса, заданных в свойстве **Источник записи** самой формы). Если надо связать поле с выражением, необходимо нажать кнопку  . Откроется окно диалога «Построитель выражений», в котором в центре окна диалога, используя

поля, функции и кнопки арифметических и логических операторов, надо задать выражение;

- используя панель инструментов форматирования или окно свойств поля ввода, можно задать для него тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры;

- если надо создать поле, информация из которого должна быть доступна только для чтения необходимо установить значение свойства **Доступ**, равное **Нет;**

- свойство **Всплывающая подсказка** вкладки «Другие» позволяет создать краткое пояснение к полю, которое будет появляться на экране, когда указатель установлен на поле и удерживается на панели некоторое время;

- для определения значения поля по умолчанию надо задать свойство **Значение по умолчанию**;

- для изменения свойств надписи к полю ввода необходимо выделить его, открыть для него окно свойств, затем можно изменить для надписи тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры.

Создать поле ввода можно более быстрым способом:

- выбрать на **Ленте** на вкладке **Конструктор** в группе «Сервис» команду **Добавить существующие поля**, на экране появиться панель со списком полей данной формы, который был задан при определении свойств формы;

- установить указатель мыши на поле из данного списка и, удерживая его в нажатом состоянии, перенести в то место формы, в котором предполагается разместить поле, в форме появится созданный объект, состоящий из поля ввода и надписи к нему; поле ввода уже связано с полем таблицы, а надпись к нему – соответствует имени поля;

- используя панель инструментов форматирования или окно свойств поля ввода и надписи, задать для них тип шрифта, размер, цвет шрифта и другие параметры.

### 3.1.4.5 Создание кнопок управления

Кнопки используются в формах для выполнения определенного действия или ряда действий. Например, можно создать в форме кнопку, открывающую другую форму, или создать набор кнопок для перемещения по записям таблицы, если не устраивают стандартные средства перемещения, предусмотренные в форме. Для того, чтобы кнопка выполняла какое-либо действие, необходимо создать макрос или процедуру обработки события и связать их со свойством кнопки **Нажатие кнопки**.

В Access предусмотрено создание более 30 разных кнопок, что значительно облегчает работу пользователя, избавляя его от необходимости самостоятельно разрабатывать макросы, достаточно лишь воспользоваться **Мастером** по созданию кнопки. Для добавления в форму командной кнопки необходимо установить режим использования **Мастера** на **Ленте** в группе «Элементы управления» и нажать элемент **Кнопка**. Затем установить мышь в то место формы, где будет находиться кнопка, определив тем самым ее размеры и местонахож-

дение. Когда будет отпущена кнопка мыши, Access выведет на экран диалоговое окно **Мастера** по созданию кнопок.

Открывшееся первое диалоговое окно **Мастера** позволяет выбрать категорию, то есть тип команды, которую должна выполнить кнопка. Содержащиеся в окне команды для навигации по записям совпадают с теми действиями, которые осуществляются из интерфейса Access. При перемещении по списку **Категории** список **Действия** обновляется. Окно **Образец** показывает изображение кнопки со стандартной иконкой, которая представляет данную команду. После выбора нужного значения из списка **Действия** нажать кнопку **Далее**. Второе диалоговое окно позволяет разными способами заменить изображение на кнопке. Можно щёлкнуть на переключателе **Рисунок** и воспользоваться стандартными иконками Access или использовать растровый файл в роли иконки кнопки. Кроме того, есть возможность выбора переключателя **Текст**, чтобы вместо иконки в кнопке присутствовал текст. При использовании стандартных кнопок надо выбрать флажок **Показать все рисунки**, чтобы отобразить все имеющиеся стандартные иконки, или отменить установку флажка, чтобы вывести только те иконки, которые предлагаются программой для этой кнопки.

При использовании рисунков в растровых файлах необходимо щёлкнуть по переключателю **Рисунок**, нажать на кнопку **Обзор**, чтобы вывести на экран диалоговое окно, которое позволяет выбрать любой файл формата BMP в качестве иконки элемента управления. При использовании текста необходимо выбрать переключатель **Текст**, затем ввести нужный текст в поле справа.

Последнее диалоговое окно позволяет дать имя и завершить создание командной кнопки.

В качестве примера рассмотрим создание кнопки, позволяющей перейти по нажатию на эту кнопку к первой записи:

- открыть разрабатываемую форму в режиме конструктора;
- перейти в нижнюю часть формы и курсором мыши раздвинуть **Область данных** на два шага сетки, чтобы на эту свободную часть формы поместить кнопку;
- нажать инструмент **Кнопка**, затем поместить кнопку на форму, после этого откроется окно Мастера кнопок;
- в первом диалоговом окне выбрать раздел **Переходы по записям** в левом окне формы **Категории**, в правом окне формы **Действия** выбрать команду **Первая запись**, нажать кнопку **Далее**;
- во втором диалоговом окне установить переключатель на **рисунок** и выбрать **Первая запись** из меню, нажать кнопку **Далее**;
- в следующем окне задать наименование кнопки, в верхнее поле ввести наименование «Первая», нажать кнопку **Готово**.

### 3.1.4.6 Использование линий и прямоугольников

Использование линий и прямоугольников в экранной форме применяется для улучшения внешнего вида формы и восприятия информации, а также для объединения в группу схожих по смыслу объектов.

Для добавления в экранную форму линий используется элемент **Линия** на панели элементов. Линии обычно полезны для подчеркивания или для акцентирования внимания на различных частях формы, а также для визуального деления формы на несколько частей. Линия может быть вертикальной, горизонтальной или по диагонали.

Чтобы нарисовать линию на экране надо выбрать элемент **Линия**, установить указатель мыши в то место, где должна начинаться линия, и переместить его до получения линии нужной длины. Настройка параметров линии осуществляется с помощью ее свойств. Свойство **Цвет границы** задает цвет линии, а свойство **Тип границы** позволяет указать стиль линии, например, «Сплошная», «Штриховая». Свойство **Ширина границы** предназначено для задания толщины линии.

Для добавления в экранную форму прямоугольников используется элемент **Прямоугольник** на панели элементов. Прямоугольники помогают выделить отдельные части формы или визуально объединить в группы несколько элементов управления, очертив вокруг них рамку.

Чтобы нарисовать прямоугольник, надо выбрать элемент **Прямоугольник**, установить указатель мыши в место расположения одного из углов прямоугольников и переместить указатель мыши до получения прямоугольника нужного размера. Свойство **Тип фона** объекта позволяет задать, будет ли объект прозрачным или нет. Свойства **Цвет границы** и **Ширина границы** задают цвет и толщину прямоугольника соответственно.

### 3.1.4.7 Специальные средства, используемые для ввода данных

Access предоставляет в распоряжение пользователей ряд специальных средств, предназначенных для упрощения процедуры ввода данных с помощью форм. К ним относятся поля со списком, списки, переключатели и флажки. Основное назначение этих объектов заключается в том, чтобы ускорить ввод информации в таблицу и сделать его менее рутинным.

*Поле со списком* широко используется в Access. Оно предоставляет возможность выбора из списка одного из допустимых значений или непосредственно ввода в поле значения, которое отсутствует в списке. Список значений хранится в свернутом виде до тех пор, пока не нажать кнопку раскрытия списка, расположенную в правой части объекта.

Предположим, что для одного из полей необходимо создать на форме раскрывающийся список, в который можно добавлять значения. Выполняются следующие действия:

- открыть форму в режиме конструктора;
- выделить заданное поле и нажать правую кнопку мыши, на экране появиться контекстное меню этого поля;
- выбрать **Преобразовать элемент в**, а затем **Поле со списком**, при этом появится кнопка раскрытия списка в правой части поля;
- для определения элементов списка вызвать еще раз меню поля и выбрать **Свойства**, на экране появится окно свойств поля;

- перейти на вкладку «Данные»;

- установить для свойства **Тип источника строк** значение **Список значений**;

- задать с помощью свойства **Источник строк** список предопределенных значений, для этого ввести в поле элементы списка по одному в строчку, разделяя точкой с запятой;

- установить для свойства **Ограничиться списком** значение **Нет**;

- закрыть окно свойств;

- сохранить созданную форму, воспользовавшись командой **Сохранить**, после чего необходимо провести тестирование.

Тестирование осуществляется по следующим этапам:

а) убедиться, что допустимые значения полностью помещаются в отведенном для них поле;

б) поэкспериментировать с непосредственным вводом значений, проверить, что они не были урезаны в этом поле;

в) если были наложены условия по проверке достоверности данных на поле, убедиться, что ни одно из возможных значений не противоречит ограничениям, наложенным на поле;

Если после тестирования потребуется изменить список допустимых значений для раскрывающегося списка, то вновь открыть для него окно свойств, а затем внести требуемые изменения.

Назначение *списка* аналогично полю со списком, за исключением того, что список не допускает ввода значений, отсутствующих в списке. Его можно использовать в формах в тех случаях, когда все альтернативные значения можно поместить в список, при этом пользователю не потребуется вводить в поле значения, отсутствующие в списке. Создание списка выполняется с помощью окна свойств. При этом выполняется та же последовательность действий, что и при создании раскрывающегося списка с помощью элемента **Поле со списком**.

Для индикации состояния, которое может иметь только одно из двух допустимых значений, используются *флажки*. Они могут использоваться по одному или группами. Например, поле «Наличие на складе» в таблице «Видеокарты» может принимать значения «Да» или «Нет». Установленный флажок будет соответствовать значению «Да», а снятый – значению «Нет».

Для поля «Наличие на складе» необходимо задать логический тип. Если это не было сделано при задании структуры таблицы «Видеокарты», надо осуществить модификацию в окне конструктора таблицы. Для создания флажка на форме рекомендуется следующая последовательность действий:

- открыть форму в режиме конструктора;

- выбрать элемент **Флажок** на **Ленте** в группе «Элементы управления» (  );

- нажать мышью место предполагаемого размещения элемента в форме;

- открыть окно свойств размещенного в форме флажка;

- для связи созданного поля с полем таблицы «Видеокарты» выбрать свойство **Данные** вкладки «Данные», в поле ввода значения свойства, воспользовавшись кнопкой раскрытия списка, выбрать поле «Наличие на складе»;

- выделить надпись созданного флажка и в окне свойств скорректировать свойство **Имя** вкладки «Макет», введя значение «Наличие на складе (Да/Нет)»;

- просмотреть форму в режиме формы.

Пример вида формы с элементом **Флажок** приведен на рисунке 3.7.

Для редактирования поля «Наличие на складе», который имеет логический тип, можно было бы также использовать элемнты **Группа переключателей** или **Выключатель**, имеющие аналогичные свойства. Эти объекты позволяют выбрать одно из нескольких значений поля. Переключатели широко используются не только в Access, но и в других приложениях Windows. Объекты типа **Группа переключателей** представляют из себя составные объекты, содержащие внутри себя элементы, наделенные собственными свойствами.

Кнопка **Группа переключателей** позволяет создать набор флажков или группу выключателей, то есть кнопок с фиксацией, которые принимают «нажатый» или «ненажатый» (выступающий вид). В любом случае, если пользователь выбирает одну из этих опций, то остальные автоматически деактивируются. Мастер позволяет присвоить каждой опции значения, которые могут быть сохранены в поле или таблице для последующего использования. Если группа замещает собой в форме какое-либо из полей, то необходимо удалить это поле из макета, если изначально оно в него было включено.

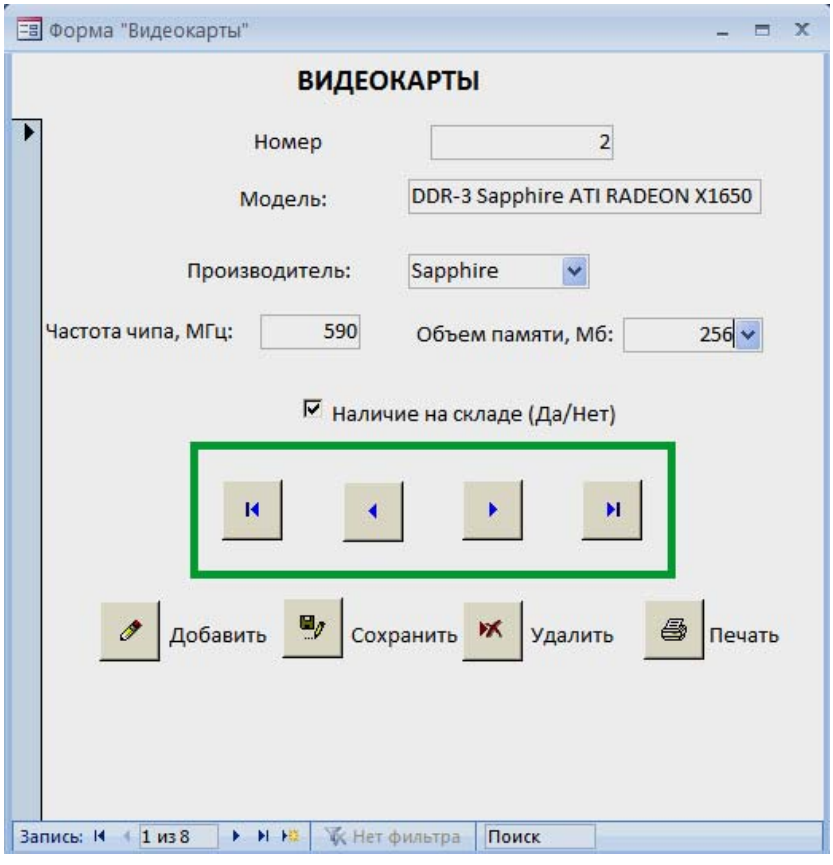



Рисунок 3.7 – Экранная форма «Видеокарты»

Рассмотрим создание переключателя для просмотра и редактирования поля «Вид оплаты» таблицы «Сведения о заказах», которое может принимать одно из значений: **Наличные \ Безналичные \ Бартер \ Электронная карточка \ В кредит**. Необходимо выполнить следующие действия:

- создать экранную форму «Сведения о заказах» в конструкторе форм, расположить в ней заголовок формы, текстовые объекты и все поля, за исключением «Вид оплаты»;

- выбрать инструмент **Группа переключателей** на **Ленте** в группе «Элементы управления»;

- открыть окно «Свойства» для вновь созданного объекта и задать в качестве значения поля **Данные** имя поля, значение которого определяется с помощью группы переключателей (в примере - «Вид оплаты»);

- выбрать инструмент **Выключатель** на панели элементов ( ) ;

- скорректировать для переключателя свойства, определяющие параметры шрифта, цвет фона и так далее;

- аналогичным образом разместить остальные переключатели группы и скорректировать их свойства;

- с помощью свойства **Подпись** изменить стандартную подпись группы переключателей на «Вид оплаты:»;

- сохранить форму и просмотреть её в режиме формы.

Вид формы с группой переключателей показан на рисунке 3.8. Теперь при редактировании заказов в поле «Вид оплаты» таблицы «Сведения о заказах» будет заноситься значение, которое устанавливается с помощью выключателя.

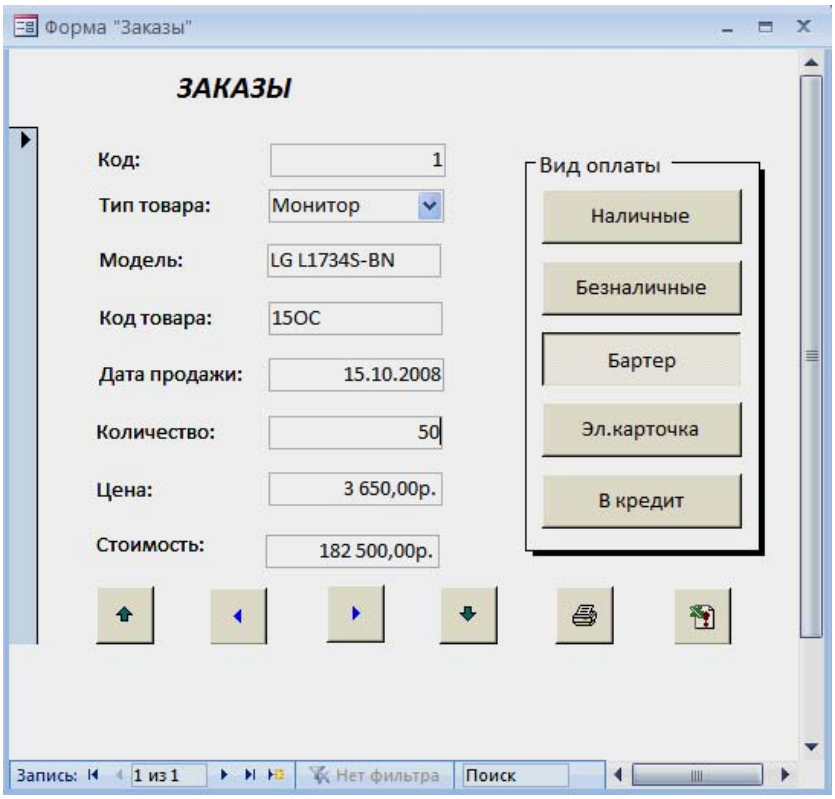



Рисунок 3.8 – Экранная форма «Сведения о заказах»

### 3.2 Задание на выполнение работы

3.2.1 Создать таблицу «Видеокарты» в соответствии с таблицей 3.3.

3.2.2 Создать и сохранить форму для таблицы «Видеокарты» с помощью инструмента «Форма».

3.2.3 Создать и сохранить разделенную форму для таблицы «Видеокарты».

3.2.4 Создать и сохранить экранную форму с помощью мастера для таблицы «Видеокарты».

3.2.5 Создать форму с помощью конструктора. Ознакомиться с его окном, открыть всю рабочую область. Настроить форму для таблицы «Видеокарты», разместить текстовую информацию, поля ввода, кнопки управления, создать поля со списком «Производитель» и «Объем памяти», флажок поля «Наличие на складе» в соответствии с рисунком 3.7.

3.2.6 Создать таблицу «Сведения о заказе» (таблица 3.4).

3.2.7 Создать форму для ввода данных, как показано на рисунке 3.8. Для определения значений поля «Стоимость» использовать построитель выражений. Значения поля «Вид оплаты» задавать с помощью переключателей.

3.2.8 Протестировать работу всех созданных форм.

3.2.9 Оформить отчет по лабораторной работе.

### 3.3 Содержание отчета

3.3.1 Название работы.

3.3.2 Цель работы.

3.3.3 Структуры создаваемых таблиц.

3.3.4 Перечень команд, используемых при выполнении лабораторной работы, с указанием их назначения.

Таблица 3.3 – Содержание таблицы «Видеокарты»

| Номер | Модель | Производитель | Частота чипа, МГц | Объем памяти, Мб | Наличие на складе |
|---|---|---|---|---|---|
| 1 | DDR-3 Sapphire ATI RADEON X1650 Pro | Sapphire | 590 | 256 | Да |
| 2 | DDR-2 Palit GeForce 9600GT | Palit | 650 | 1024 | Нет |
| 3 | DDR-3 Sapphire ATI RADEON HD4670 | Sapphire | 750 | 1024 | Да |
| 4 | DDR-2 ZOTAC GeForce 8600GT | ZOTAC | 540 | 1024 | Да |
| 5 | DDR-3 ASUSTeK EN8800GT GeForce 8800GT | ASUS | 600 | 512 | Нет |
| 6 | DDR-2 ASUS EN9500GT MAGIC/DI | ASUS | 650 | 512 | Да |
| 7 | DDR-2 ASUS EN9600GT MG | ASUS | 650 | 512 | Нет |
| 8 | DDR-2 ZOTAC GeForce 8600GT | ZOTAC | 540 | 1024 | Да |

Таблица 3.4 – Содержание таблицы «Сведения о заказе»

| Код | Тип товара | Модель | Код товара | Дата продажи | Коли-чество | Цена | Вид оплаты |
|---|---|---|---|---|---|---|---|
| 1 | Монитор | LG L1734S-BN | 15ОС | 15.10.08 | 50 | 3650 | Бартер |
| 2 | Принтер | LaserJet P1005 | 12ВК | 05.04.08 | 7 | 3600 | Эл. карта |
| 3 | Монитор | Samsung 2032BW BSFV | 37ОС | 23.06.08 | 24 | 9800 | Безналичные |
| 4 | Принтер | Samsung SCX-4200 | 42ОС | 29.03.08 | 30 | 6200 | В кредит |
| 5 | Клавиатура | Genius KB-220 | 5ВК | 03.07.08 | 15 | 850 | Наличные |
| 6 | Монитор | Belinea 1730 S1 | 25ОС | 04.02.08 | 5 | 3370 | Эл. карта |
| 7 | Клавиатура | Logitech Deluxe 250 Y-SAF76 | 7ВК | 14.09.08 | 27 | 360 | Наличные |

## 3.4 Тесты и контрольные вопросы

3.4.1 Форма в Access - это:

а) объект базы данных, в который добавляются элементы управления, реагирующие на действия пользователей или служащие для ввода, отображения и изменения данных в полях;

б) объект базы данных, в котором данные хранятся в виде записей (строк) и полей (столбцов);

в) объект базы данных, предназначенный для вывода на печать данных, организованных и отформатированных в соответствии с требованиями пользователя;

г) набор условий, применяемых для отбора подмножества данных или для сортировки данных.

3.4.2 Что является достоинством формы, созданной при помощи инструмента «Форма»:

а) легкий способ создания формы;

б) позволяет распечатать данные;

в) позволяет вводить данные;

г) позволяет изменять данные?

3.4.3 Что представляет собой распределенная форма:

а) отображение данных в режиме формы;

б) одновременное отображение данных в режимах формы и таблицы;

в) отображение данных в режиме таблицы;

г) отображение данных в режиме формы с возможностью перехода по записям;

д) отображение данных в режиме таблицы с возможностью перехода по записям?

3.4.4 Назовите правильную последовательность этапов создания формы с помощью **Мастера форм**:

а) изменение внешнего вида формы; применение требуемого стиля; выбор полей для формы; задание имени формы; выбор действия;

б) выбор полей для формы; применение требуемого стиля; задание имени формы; изменение внешнего вида формы; выбор действия;

в) применение требуемого стиля; выбор полей для формы; изменение внешнего вида формы; задание имени формы; выбор действия;

г) выбор полей для формы; изменение внешнего вида формы; применение требуемого стиля; задание имени формы; выбор действия;

д) выбор действия; выбор полей для формы; изменение внешнего вида формы; применение требуемого стиля; задание имени формы.

3.4.5 Какой внешний вид формы нельзя выбрать при создании формы с помощью **Мастера**:

а) в один столбец;

б) ленточный;

в) в одну строчку;

г) выровненный;

д) табличный?

13.4.6 На закладке **Упорядочить** находятся группы:

а) «Шрифт»; «Макет элемента управления»; «Элементы управления»; «Размер»; «Положение»;

б) «Автоформат»; «Макет элемента управления»; «Выравнивание элементов»; «Размер»; «Сетка»;

в) «Шрифт»; «Элементы управления»; «Выравнивание элементов»; «Размер»; «Положение»;

г) «Автоформат»; «Макет элемента управления»; «Выравнивание элементов»; «Сервис»; «Положение»;

д) «Автоформат»; «Макет элемента управления»; «Выравнивание элементов»; «Размер»; «Положение».

3.4.7 На какой вкладке **Ленты** находится команда **Конструктор форм**:

а) **Создание**;

б) **Работа с базами данных**;

в) **Главная**;

г) **Режим таблицы**?

13.4.8 Какой элемент, помещаемый на форму, позволяет разместить текст:

а) надпись;

б) поле;

в) выключатель;

г) флажок;

д) кнопка?

3.4.9 Какой элемент, помещаемый на форму, позволяет задать значение логического поля:

а) надпись;

б) поле;

в) выключатель;

г) флажок;

д) кнопка?

3.4.10 Какой элемент, помещаемый на форму, позволяет осуществить различные действия:

а) надпись;

б) поле;

в) выключатель;

г) флажок;

д) кнопка?

3.4.11 Для чего используются экранные формы?

3.4.12 Какие команды используются для создания форм в Access?

3.4.13 Какая команда предназначена для создания простейшей формы?

3.4.14 Что представляет собой форма, созданная с помощью инструмента «Форма»?

3.4.15 Как перейти из режима формы в режим таблицы?

3.4.16 Каким образом создается форма с помощью мастера?

3.4.17 Что включает окно конструктора форм?

3.4.18 Для каких случаев используется конструктор форм?

3.4.19 Какие области может включать окно конструктора форм?

3.4.20 Как изменяется размер области в форме?

3.4.21 Какие группы инструментов находятся на вкладке **Конструктор**?

3.4.22 Какие инструменты находятся в группе «Элементы управления»?

3.4.23 Как получить доступ к свойствам объекта?

3.4.24 Что включают свойства объекта?

3.4.25 Какие возможны действия над объектами?

3.4.26 Как можно выделить объект?

3.4.27 Как можно изменить размер объекта?

3.4.28 Как изменяется местоположение объекта?

3.4.29 Каким образом можно перемещать связанные объекты?

3.4.30 Как удалить объект?

3.4.31 Как изменить размер объекта?

3.4.32 Как изменить порядок обхода объектов в форме?

3.4.33 Какие процедуры выполняются при создании формы?

3.4.34 Какие действия выполняются при настройке формы?

3.4.35 Какие действия выполняются при размещении текста в форме?

3.4.36 Какие свойства задаются для текста?

3.4.37 Как добавить в форму поле ввода?

3.4.38 Каким образом можно связать поле с выражением?

3.4.39 Для чего используются в форме кнопки?

3.4.40 Как добавить в форму командную кнопку?

3.4.41 Что задается при работе мастера по созданию кнопок?

3.4.42 Для чего используются в формах линии и прямоугольники?

3.4.33 Для чего используется раскрывающийся список?

3.4.44 В чем заключаются отличия «списка» и «поля со списком»?

3.4.45 Для чего используется «флажок»?

3.4.46 Какой тип должно иметь поле, для которого используется флажок?

3.4.47 Для чего используется «группа переключателей»?

# 4 Создание отчетов в Microsoft Office Access 2007

Целью работы является создание и использование в Microsoft Office Access 2007 отчетов, построенных для реляционной базы данных.

## 4.1 Общие положения

### 4.1.1 Способы создания отчета

Access предоставляет в распоряжение пользователей средства для создания отчета. *Отчет* - объект базы данных Microsoft Access, предназначенный для вывода на печать данных, организованных и отформатированных в соответствии с требованиями пользователя.

При создании отчета можно воспользоваться стандартными средствами, ускоряющими процесс создания отчета или разработать специальный формат с помощью конструктора отчетов. Конструктор отчетов позволяет создавать отчеты, как в табличном виде, так и в свободной форме.

Табличный отчет представляет собой напечатанную таблицу, в которой данные упорядочены по столбцам и строкам. Каждый из столбцов содержит поле исходной таблицы или вычисляемое поле, а строка представляет собой запись. Табличный отчет позволяет напечатать данные из таблиц в наиболее простом и естественном виде. Однако они не пригодны в тех случаях, когда поля исходной таблицы должны располагаться в специально отведенных для них местах отчета (почтовые этикетки, чеки, письма).

Отчеты в свободной форме позволяют устранить ограничения, свойственные табличным отчетам. При получении отчета в свободной форме можно воспользоваться стандартным форматом, автоматически создаваемым Access для каждой таблицы. В этом формате поля исходной таблицы расположены вертикально. Однако с помощью конструктора отчетов можно разработать специальный формат, где поля исходной таблицы расположены в требуемых местах отчета.

В Access используются следующие средства для создания отчета:

а) **Отчет**, позволяющий автоматически создать отчет с полями, расположенными в один или несколько столбцов;

б) **Мастер отчетов**, позволяющий создать настраиваемый отчет на основе выбранных полей;

в) **Конструктор отчетов,** в котором можно самостоятельно разработать собственные отчеты с заданными свойствами;

д) **Пустой отчет**, позволяющий самостоятельно вставлять поля и элементы управления и дорабатывать форму отчета;

е) **Наклейки**, позволяющий создать отчет для почтовых наклеек или другие этикетки.

Для создания самого простого отчета в Access необходимо выполнить следующие действия:

- открыть окно базы данных;
- убедиться, что в области переходов выбрана таблица;

- выполнить команду **Отчет** в группе «Отчеты», на **Ленте** во вкладке **Создание**.

На экране появится готовый к использованию отчет, в который включены все поля таблицы. Их названия располагаются горизонтально в том порядке, в каком они находятся в таблице. Снизу под названием каждого поля отображается его значение в таблице.

Расположение полей и записей в автоматически созданных отчетах подходит не для всех случаев, в частности, при их большом количестве. Однако отчеты могут пригодиться, когда создаются на основе запросов, выводящих на экран только нужные поля. Можно также использовать запрос для определения тех записей и порядка их сортировки, которые будут включены в отчет.

### 4.1.2 Использование мастера для создания отчета

Создание отчета с помощью мастера не требует специальных знаний и сводится к выбору таблиц, входящих в отчет, определению списка полей отчета и порядка их размещения. В мастере отчетов предоставляется больше возможностей относительно выбора полей для включения в отчет. При этом можно указать способ группировки и сортировки данных, а также включить в отчет поля из нескольких таблиц или запросов, если отношения между этими таблицами и запросами заданы заранее.

Чтобы запустить мастер отчетов, необходимо нажать на кнопку **Мастер отчетов** в группе «Отчеты» на вкладке **Создание.**

После запуска мастера построения отчета, на экране откроется окно диалога, в котором необходимо определить поля будущего отчета (рисунок 4.1). Необходимо нажать кнопку раскрытия списка **Таблицы и запросы** и из списка таблиц баз данных выбрать таблицу, для которой создается отчет. При этом в списке **Доступные поля** появляется перечень всех полей выбранной таблицы. Необходимо из данного перечня перенести в список **Выбранные поля** - поля, которые надо поместить в создаваемый отчет. Завершив выбор полей, необходимо нажать кнопку **Далее** для перехода к следующему шагу.

На втором шаге создания отчета с помощью мастера необходимо определить, требуется ли сгруппировать данные по какому-либо из полей (рисунок 4.2). Если поля не группировать, отчет произведет итоговые вычисления по всем полям с числовым типом данных для всей таблицы или запроса, на которых он основан. Можно для группировки выбрать одно поле. В этом случае отчет обеспечит для группы промежуточные вычисления, а для таблицы целиком – итоговую сумму. Можно применять до четырех группировок, вложенных одна в другую. Этот отчет включает итоговое вычисление, промежуточные результаты и подпромежуточные результаты для всех групп. Поля, по которым будет осуществляться группировка, помещаются в верхней части правого списка в отдельной рамке и выделяются на экране синим цветом. Access предлагает свой вариант группировки данных. Можно согласиться с предложенным вариантом или задать свой, используя кнопки окна диалога (таблица 4.1).

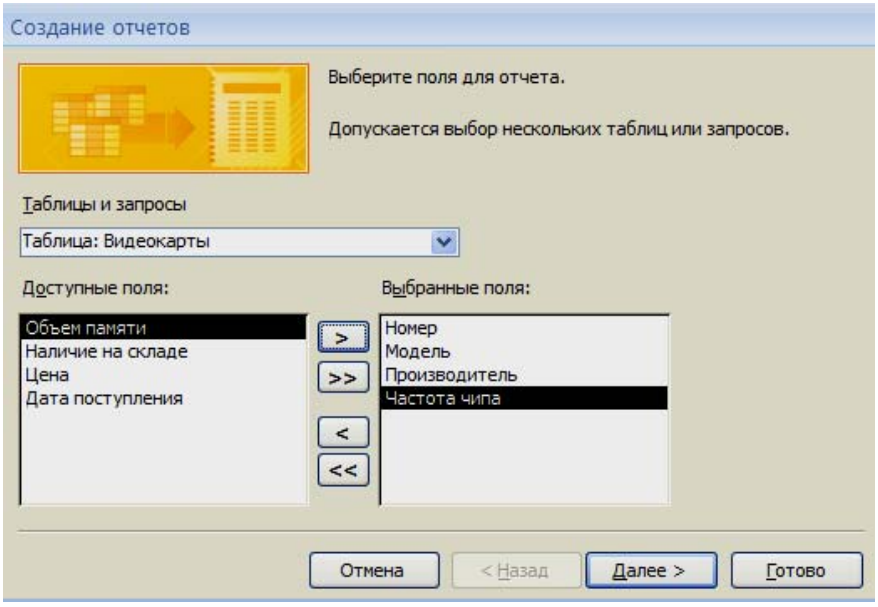



Рисунок 4.1 – Окно диалога для выбора полей отчета

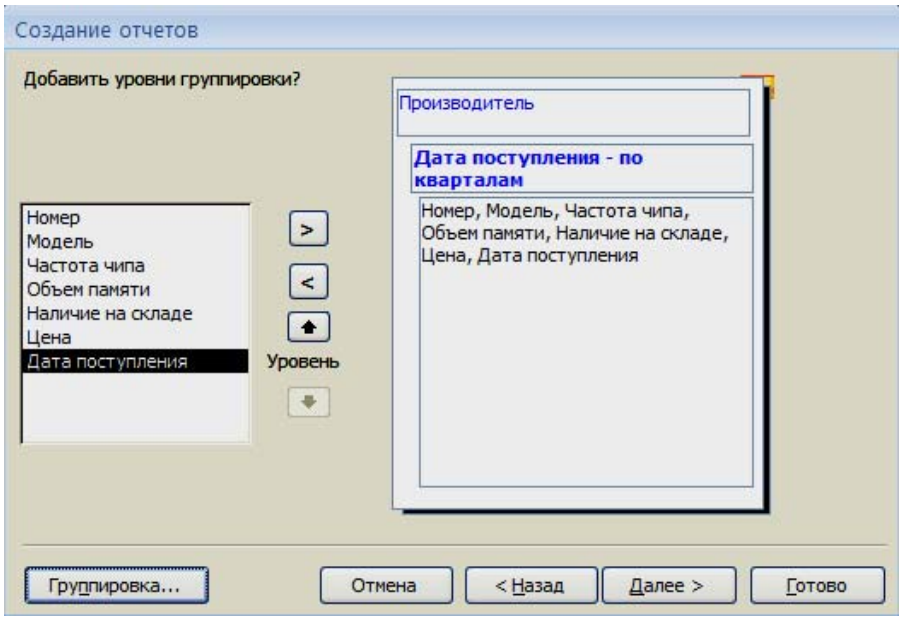



Рисунок 4.2 – Создание отчетов с помощью **Мастера отчетов**, шаг второй

Таблица 4.1 – Назначение кнопок окна диалога при определении группировки

| Кнопка | Выполнимое действие |
|---|---|
|  | Добавляет поле в рамку группировки |
|  | Удаляет поле из рамки группировки |
|  | Повышает уровень группировки, выделенного в рамке группировки поля |
|  | Понижает уровень группировки, выделенного в рамке группировки поля |

Установив группировку данных, можно изменить интервал группировки, для этого необходимо нажать кнопку **Группировка**. Появляется диалоговое окно «Интервалы группировки». Данное окно позволяет проводить группировку по диапазону значений в записи, что предпочтительнее, чем группировка по отдельным записям. Если, например, используется поле с датой как основа для группировки, данные можно сгруппировать в отдельные группы для каждого года или провести группирование по месяцам этих данных в поле. Виды диапазонов, которые можно задавать, зависят от типа данных. Диалоговое окно «Интервалы группировки» включает в себя поля, на основе которых проводится группировка. Справа от каждого поля в окне имеется раскрывающийся список, который можно использовать для выбора соответствующего интервала для типа данных этого поля.

Для перехода к следующему окну диалога надо нажать кнопку **Далее**. В этом окне диалога задается порядок сортировки записей внутри каждой группы (до четырех полей) и вычисления, выполняемые для записей, на задание которых можно перейти по кнопке «Итоги…» (рисунок 4.3). Для числовых полей можно вывести на экран среднюю сумму, минимальное или максимальное значения. Для возврата в окно сортировки необходимо нажать кнопку **ОК**.

На следующих двух шагах создания отчета с помощью мастера необходимо определить вид макета отчета и стиль оформления.

На заключительном шаге создания отчета можно задать имя отчета и выбрать один из двух вариантов дальнейшей работы с отчетом:

- просмотр отчета;
- изменение структуры отчета.

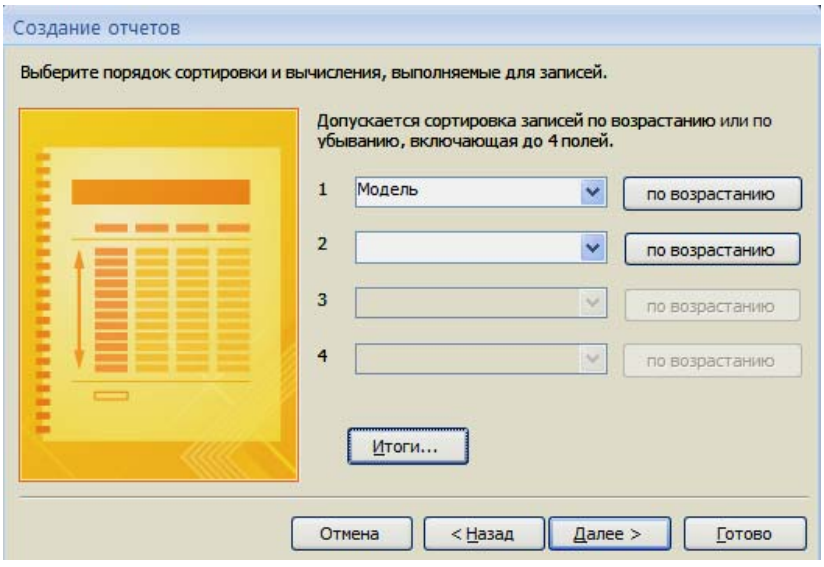



Рисунок 4.3 – Создание таблиц с помощью **Мастера отчетов**, шаг третий

### 4.1.3 Просмотр отчета

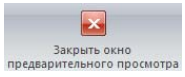
Access располагает большим набором средств для просмотра на экране созданного отчета. Для просмотра отчета необходимо нажать кнопку **Office**

 , выделить стрелку рядом с пунктом **Печать** и выбрать команду **Предварительный просмотр**.

При просмотре отчета можно использовать кнопки панели инструментов и кнопки перемещения по страницам, расположенным в нижней части окна просмотра. Назначение этих кнопок приведено в таблице 4.2.

Пользователь не имеет возможности редактировать данные, отображаемые в отчете, так же как и при работе с данными в таблицах, запросах и формах. Окно предварительного просмотра аналогично окну, появляющемуся на экране во время просмотра таблицы, запроса или формы в режиме таблицы.

Таблица 4.2 – Кнопки, используемые в окне просмотра

| Кнопка | Назначение |
|---|---|
|  | Выводит отчет на печать |
|  | Изменяет масштаб отображения отчета |
|  | Переходит в режим просмотра одной страницы отчета |
|  | Переходит в режим просмотра двух страниц отчета |
|  | Переходит в режим просмотра других страниц отчета |
|  | Устанавливает масштаб отображения отчета |
|  | Закрывает окно просмотра отчета |
|  | Переходит к просмотру первой страницы отчета |
|  | Переходит к просмотру предыдущей страницы отчета |
|  | Переходит к просмотру следующей страницы отчета |
|  | Переходит к просмотру последней страницы отчета |

### 4.1.4 Создание и редактирование отчета в конструкторе

#### 4.1.4.1 Окно конструктора отчета

Режим конструктора позволяет более подробно изучить структуру отчета. Пользователь может просматривать заголовки и примечания для отчета, определенной страницы и групп. В этом режиме отчет не выполняется, поэтому во время работы невозможно просматривать базовые данные. Однако некоторые задачи удобнее выполнять в режиме конструктора, а не макета.

К отчету можно добавлять различные элементы управления, такие как надписи, рисунки, линии и прямоугольники. Можно изменять источник эле-

мента управления «Поле» непосредственно в самом поле, без использования окна свойств. Можно также изменять определенные свойства, недоступные в режиме макета.

Для открытия окна конструктора отчетов необходимо открыть на **Ленте** вкладку **Создание**, нажать кнопку **Конструктор отчетов** в группе «Отчеты».

На экране откроется окно конструктора отчетов (рисунок 4.4). Конструктор отчетов во многом схож с конструктором форм и имеет аналогичную панель элементов. **Лента** содержит следующие группы «Представление», «Шрифт», «Группировка и итоги», «Сетка», «Элементы управления», «Сервис». В группе «Сервис» команда **Добавить существующие поля** откроет окно «Список полей», из которого путем «перетаскивания» можно разместить поля таблиц в область данных.

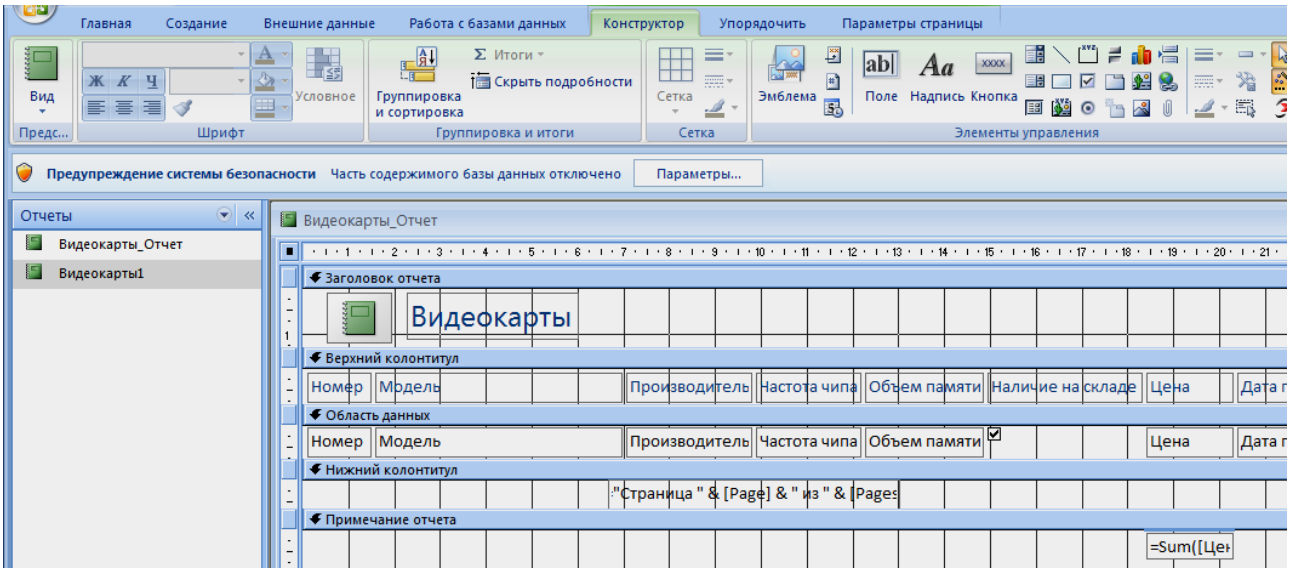



Рисунок 4.4 – Окно конструктора отчетов

Также как и в формах, в отчетах существуют области заголовка отчета, верхнего и нижнего колонтитулов, данных, примечаний:

**- Заголовок отчета** размещается в начале отчета;

**- Верхний колонтитул** отображается в начале каждой страницы и под заголовком отчета;

**- Область заголовка группы** отображается перед первой записью каждой группы;

**- Область данных** отображается для каждой записи, входящей в отчет;

**- Область примечания группы** отображается после области данных последней записи каждой группы;

**- Нижний колонтитул** отображается в нижней части каждой страницы;

**- Область примечаний** отображается в конце отчета.

При формировании отчета для предварительного просмотра или печати Access начинает печать с заголовка отчета и верхнего колонтитула. Затем до конца страницы печатается область данных, после которой Access помещает нижний колонтитул, разделитель страниц и верхний колонтитул следующей

страницы. Каждая страница содержит верхний колонтитул, столько областей данных, сколько помещается на странице, и нижний колонтитул. На последнюю страницу отчета, перед нижним колонтитулом, помещается область примечаний отчета.

Если в отчете имеются области заголовка и примечания группы, порядок печати практически не меняется. Разница заключается в том, что перед первой записью каждой группы помещается область заголовка группы и после последней записи каждой группы помещается область примечания группы.

Объекты, расположенные в верхнем колонтитуле, предназначены для печати информации или рисунков в верхней части каждой страницы отчета, а объекты, расположенные в нижнем колонтитуле, – в нижней части. И соответственно, объекты заголовка отчета печатаются на первой странице, а объекты области итогов – на последней. Если в отчете есть группы, то для каждой группы можно создать области верхнего и нижнего колонтитула.

Основными объектами отчета являются пояснительный текст и поля отчета. Для улучшения восприятия информации могут добавляться рисунки, линии, прямоугольники. Поля отчета могут непосредственно соответствовать полям исходной таблицы или являться результатом работы над ними.

При создании и модификации отчетов конструктор позволяет удалять, добавлять, перемещать области вместе с расположенными в них объектами. Пользователь может установить цвет и управлять параметрами отображения любых элементов и областей отчета.

### 4.1.4.2 Размещение даты печати отчета

В свойстве **Формат поля** можно использовать встроенные форматы даты и времени или задать пользовательские форматы для типа данных «Дата/время». Встроенные форматы даты и времени включают длинный, средний и краткий форматы даты или времени.

Пользовательские форматы отображаются в соответствии со значениями, установленными в региональных настройках Microsoft Windows. Пользовательские форматы, противоречащие заданным региональным параметрам, игнорируются. Пользовательский формат можно использовать для добавления букв «н. э.» перед номером года или «до н. э.» после него, в зависимости от введенного числа (положительного или отрицательного). Чтобы увидеть, как действует этот формат, необходимо создать новое поле таблицы, задать числовой тип данных и ввести формат.

Обычно в заголовке отчета или в верхнем колонтитуле размещают дату печати отчета. Для создания данного элемента в отчете необходимо выполнить следующие действия:

- выбрать инструмент **Поле**  на панели инструментов;

- установить указатель мыши на место в область верхнего колонтитула, в котором предполагается разместить поле даты, в отчете появится связанный объект, состоящий из поля ввода и надписи к нему;

- выделить надпись связанного поля и удалить ее, нажав клавишу **Delete**;

- выделить поле ввода и открыть для него окно свойств;

- перейти на вкладку «Данные» и нажать кнопку вызова построителя свойства **Данные** ( ), на экране откроется окно диалога «Построитель выражений»;

- открыть отдел «Встроенные функции» папки **Функции** и, используя функцию **Now**, задать выражение для значения поля **= Now ( )**;

- нажать кнопку раскрытия списка свойства **Формат** вкладки «Макет» и выбрать устраивающий формат даты в отчете.

Разместить дату и время печати отчета можно еще более простым способом, так как в Access для этих целей имеется команда меню:

- находясь в окне конструктора отчетов, поместить на отчет элемент **Дата и время** , взяв его из группы «Элементы управления» на закладке **Конструктор**;

- в открывшемся окне диалога «Дата и время» выбрать форматы отображения даты и времени, можно поместить в отчет только дату или только время, удалив установку соответствующей опции; завершив установку, нажать кнопку **ОК**;

- выделить созданный объект и перенести в требуемую область отчета.

### 4.1.4.3 Группировка данных

Разбиение данных на группы зачастую облегчает их восприятие. Например, отчет с группировкой продаж по странам может выявить тенденции, которые иначе остались бы незамеченными. Кроме того, подведение промежуточных итогов для каждой группы позволит избежать многочисленных подсчетов вручную на калькуляторе.

Приложение Microsoft Office Access 2007 упрощает работу с такими отчетами. С его помощью можно создавать базовый отчет с группировкой с использованием мастера отчетов, выполнять группировку или сортировку готового отчета либо изменять ранее заданные параметры группировки и сортировки.

Для удобства работы с отчетом часто возникает необходимость объединять записи в группы. Для этой цели используется команда **Группировка и сортировка**, находящаяся на **Ленте** на закладке **Конструктор** в группе «Группировка и итоги». Она позволяют создавать до 10 уровней вложенности групп и выполнять над ними следующие операции:

- напечатать текст, идентифицирующий конкретные группы;

- напечатать каждую группу с новой страницы.

Для добавления группы в отчет необходимо в окне конструктора отчета выполнить следующие действия:

- нажать кнопку **Группировка и сортировка**, на экране откроется окно диалога «Группировка, сортировка и итоги», которое содержит весь список ранее созданных групп, в нем можно редактировать или удалять имеющиеся группы, а также добавлять новые (рисунок 4.5);

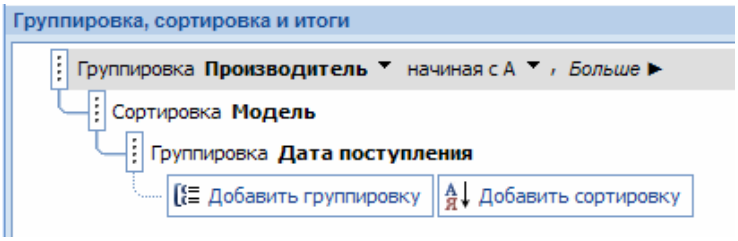



Рисунок 4.5 – Область **Группировка, сортировка и итоги**

- для ввода поля или выражения группировки необходимо нажать кнопку **Добавить группировку** , в появившемся списке выбрать имя поля, по которому осуществляется группировка;

- для осуществления сортировки нажать кнопку **Добавить сортировку** и задать имя поля, по которому осуществляется сортировка, ее порядок и дополнительные параметры;

- скорректировать свойство **Заголовок группы**;

- закрыть окно диалога «Группировка, сортировка и итоги»;

- в конструкторе отчетов перенести поле и надпись к нему в созданную область «Заголовок группы», теперь при просмотре отчета в начале каждой группы выводятся наименования объектов, по которым осуществляется группировка.

4.1.4.4 Работа со страницами отчета

На каждой странице отчета можно поместить номер страницы. Для этого необходимо выполнить следующие действия:

- находясь в окне конструктора отчетов, выполнить команду **Номера страниц** , взяв элемент на **Ленте** с вкладки **Конструктор** из группы «Элементы управления»;

- в открывшемся окне диалога «Номера страниц» (рисунок 4.6) задать формат и расположение создаваемого объекта, список **Выравнивание** определяет расположение объекта на странице;

- установив все нужные значения, нажать кнопку **ОК**.

При необходимости нумерации записей в группе или по всему отчету необходимо выполнить следующее:

- добавить в область данных отчета несвязанное поле;

- открыть окно свойств элемента управления;

- в поле ввода свойства **Данные** ввести **=1**;

- для свойства **Сумма с накоплением** установить значение **Для всего** или **Для группы** в зависимости от того, что надо нумеровать.

Для экономии бумаги отчет с данными, расположенными в столбец, можно преобразовать в отчет с заголовком и с областью данных, выводимой в два столбца. Для создания подобного отчета необходимо использовать опции макета страницы:

- создать отчет, у которого информация располагается в один столбец;

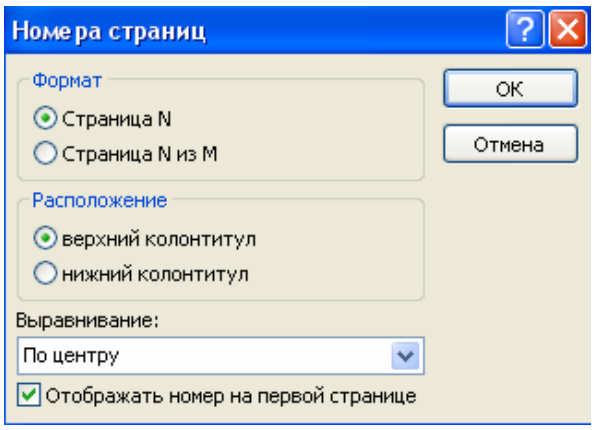



Рисунок 4.6 – Окно диалога «Номера страниц»

- для расположения данных в два столбца перейти на вкладку **Параметры страницы**;
- в группе «Разметка страницы» выполнить команду **Столбцы**;
- в поле ввода **Число столбцов** указать количество, равное **двум;**
- в рамке **Размер столбца** уменьшить значение параметра **Ширина**;
- выбрать макет столбцов, установив соответствующую опцию;
- нажать **ОК**.

Просмотрев отчет в окне предварительного просмотра, можно убедиться, что вид отчета изменился.

### 4.2 Задание на выполнение работы

4.2.1 Запустить Access, открыть свою базу данных.

4.2.2 Добавить в таблицу «Видеокарты», созданную в предыдущей лабораторной работе, новые поля «Цена» и «Дата поступления», заполнить поля значениями.

4.2.3 Создать отчет с помощью команды **Отчет** для таблицы «Видеокарты».

4.2.4 Создать отчет с помощью мастера для таблицы «Видеокарты», группировку задать по полям «Производитель» и «Дата поступления», задать интервал группировки по кварталам поля «Дата поступления», внутри групп отсортировать по возрастанию значений в поле «Модель». В отчет не включать поле «Номер», итоговое значение задать по полю «Цена».

4.2.5 Создать отчет в конструкторе отчетов по заданию пункта 4.2.4. Сформировать заголовки отчета, групп, разместить дату печати отчета в заголовке отчета, пронумеровать записи в группе.

4.2.6 Оформить отчет по выполненной работе.

### 4.3 Содержание отчета

4.3.1 Название работы.

4.3.2 Цель работы.

4.3.3 Перечень команд, используемых при выполнении лабораторной работы, с указанием их назначения.

### 4.4 Тесты и контрольные вопросы

4.4.1 Отчет – объект базы данных, основное назначение которого:

а) описание и вывод на печать документов на основе данных базы;

б) ввод, отображение и изменение данных в полях;

в) указание, какие данные из базы данных будут отображаться;

г) создание и использование базы данных;

д) структурирование описания автоматически выполняемых действий.

4.4.2 На какой вкладке **Ленты** находится команда **Конструктор отчетов**:

а) **Главная**;

б) **Работа с базами данных**;

в) **Создание**;

г) **Режим таблицы**?

4.4.3 Команда **Мастер отчетов** позволяет:

а) автоматически создать отчет с полями, расположенными в один или несколько столбцов;

б) создать настраиваемый отчет на основе выбранных полей;

в) самостоятельно разработать собственные отчеты с заданными свойствами;

д) самостоятельно вставлять поля и элементы управления и дорабатывать форму отчета.

4.4.4 Укажите правильную последовательность этапов создания отчета мастером отчетов:

а) выбор полей, добавление уровня группировки, задание порядка сортировки записей, выбор варианта дальнейшей работы;

б) выбор полей, добавление уровня группировки, задание порядка сортировки записей, выбор варианта дальнейшей работы, задание имени отчета;

в) выбор полей, задание порядка сортировки записей, задание имени отчета, добавление уровня группировки, выбор варианта дальнейшей работы;

г) выбор полей, задание порядка сортировки записей, добавление уровня группировки, задание имени отчета, выбор варианта дальнейшей работы;

д) выбор полей, добавление уровня группировки, задание порядка сортировки записей, задание имени отчета, выбор варианта дальнейшей работы.

4.4.5 Кнопка  в окне диалога мастера отчета при определении группировки предназначена для:

а) добавления поля в рамку группировки;

б) удаления поля из рамки группировки;

в) повышения уровня группировки, выделенного в рамке группировки поля;

г) понижения уровня группировки, выделенного в рамке группировки поля.

4.4.6 На какой вкладке **Ленты** находится команда **Группировка и сортировка**:

а) **Главная**;

б) **Работа с базами данных**;

в) **Создание**;

г) **Режим таблицы**;

д) **Конструктор**?

4.4.7 Для чего применяется разбиение данных на группы:

а) для нумерации записей;

б) для облегчения восприятия данных;

в) для изменения расположения информации в отчете;

г) для сортировки полей?

4.4.8 На какой вкладке **Ленты** находится команда **Столбцы**, используемая для изменения расположения информации в отчете:

а) **Упорядочить**;

б) **Работа с базами данных**;

в) **Создание**;

г) **Параметры страницы**;

д) **Конструктор**?

4.4.9 Какая функция позволяет вставить текущую дату печати отчета:

а) Now;

б) Time;

в) Date;

г) Day?

4.4.10 Укажите пиктограмму команды **Номера страниц**:

а)  ; б)  ; в)  ; г)  ; д)  .

4.4.11 Что понимается под отчетом в MS Access?

4.4.12 Что представляет собой отчет, созданный командой **Отчет**?

4.4.13 Какие в Access используются средства для создания отчета?

4.4.14 Каким образом создается отчет с помощью мастера?

4.4.15 Как осуществляется предварительный просмотр отчета?

4.4.16 Что включает окно конструктора отчета?

4.4.17 Как добавить в окно конструктора заголовок отчета, если эта область не была сформирована?

4.4.18 Что является основными объектами отчета?

4.4.19 Как разместить дату печати отчета в заголовке отчета?

4.4.20 Как в конструкторе отчета осуществить группировку данных?

4.4.21 Как удалить сортировку или группировку в отчете?

4.4.22 Каким образом можно поместить на каждой странице отчета ее номер?

4.4.23 Какие действия выполняются при необходимости нумерации записи в группе?

# 5 Многотабличная база данных в Microsoft Office Access 2007

Целью работы является создание и работа с реляционной базой данных, состоящей из нескольких таблиц, связанных друг с другом посредством общих ключевых полей, в Microsoft Office Access 2007.

## 5.1 Общие положения

### 5.1.1 Отношения между таблицами в базе данных

Access является системой управления реляционными базами данных. Реляционные базы данных в настоящее время наиболее распространены и фактически являются промышленным стандартом. Единицей хранящейся в реляционной базе данных информации является таблица. Каждая таблица представляет собой совокупность строк и столбцов, где строки (записи) соответствуют конкретному объекту, событию или явлению, а столбцы - атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.

Между отдельными таблицами базы данных могут существовать *связи* [1, 3 - 7]. Установление связи между таблицами обеспечивает следующее:

- повышение достоверности хранимой в базе данных информации, так как многие СУБД автоматически выполняют контроль целостности данных, вводимых в базу, в соответствии с установленными связями;

- облегчение доступа к данным при выполнении таких операций, как поиск, просмотр, редактирование, выборка и подготовка отчетов, при этом уменьшается количество явных обращений к таблицам данных и число манипуляций в каждой их них.

В каждой таблице базы данных может существовать первичный ключ - поле или набор полей, однозначно идентифицирующий запись. Значение первичного ключа в таблице базы данных должно быть уникальным, то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа. Первичные ключи облегчают установление связи между таблицами.

В базах данных возможно четыре типа отношений между таблицами: «один-к-одному», «один-ко-многим», «много-к-одному», «много-ко-многим».

При отношении «один-к-одному» каждая запись в первой таблице может иметь не более одной связанной записи во второй таблице и наоборот. Отношения этого типа используются нечасто, поскольку обычно сведения, связанные таким образом, хранятся в одной таблице. Отношение «один-к-одному» используется для разделения таблицы, содержащей много полей, с целью отделения части таблицы по соображениям безопасности, а также с целью сохранения сведений, относящихся к подмножеству записей в главной таблице. После определения такого отношения у обеих таблиц должно быть общее поле.

Например, имеются две таблицы «Персональная карточка студента» и «Медицинская карточка студента». В таблице «Персональная карточка студента» содержатся общие сведения о студенте (код, фамилия, имя, отчество, адрес,

дата рождения и другие данные), а в таблице «Медицинская карточка студента» - сведения для студенческой поликлиники. Между этими таблицами существует отношение «один-к-одному», поскольку для одного студента может существовать только одна запись, содержащая сведения. Связь между этими таблицами поддерживается при помощи совпадающих полей: «Код студента» (таблица «Персональная карточка студента») и «Код студента» (таблица «Медицинская карточка студента»). Эти поля имеют одинаковые наименования, однако могут иметь и различные. Связь между таблицами устанавливается на основании значений совпадающих полей, но не их наименований.

Отношение «один-ко-многим» означает, что одной записи из родительской таблицы может соответствовать несколько записей в дочерней таблице. Данный тип связи является самым распространенным для реляционных баз данных. Связанные отношениями таблицы взаимодействуют по принципу «главная – подчиненная». Главную таблицу часто называют родительской, а подчиненную - дочерней. Одна и та же таблица может быть главной по отношению к одной таблице базы данных и дочерней по отношению к другой. Например, связь «один-ко-многим» имеется между таблицами «Читательский билет студента» и «Выдача книг». Связь между таблицами осуществляется на основании значений совпадающих полей «Код читательского билета».

Отношение «много-к-одному» аналогичен типу «один-ко-многим». Тип отношения зависит от выбранной точки зрения. Например, если рассматривать отношение между сделанными заказами и клиентами, то получится отношение «много-к-одному».

Отношение «много-ко-многим» возникает между таблицами тех случаях, когда:

- одна запись из первой таблицы может быть связана более, чем с одной записью из второй таблицы;
- одна запись из таблицы может быть связана более чем с одной записью из первой таблицы.

Например, отношение «много-ко-многим» существует между таблицами «Дисциплины» и «Преподаватели». Одну дисциплину могут вести несколько преподавателей. Каждый преподаватель может вести несколько разных предметов. Обычно связь с отношением «многие-ко-многим» представляется в виде двух связей с отношением «один-ко-многим» через третью таблицу, назовем ее «Связи». Эта таблица будет содержать только два поля «Код дисциплины» и «Код преподавателя». Между таблицами «Дисциплина» и «Связи» отношение «один-ко-многим», между таблицами «Связи» и «Преподаватели» - «много-к-одному».

### 5.1.2 Нормализация базы данных

Для реляционной базы данных проектирование логической структуры заключается в том, чтобы разбить всю информацию по таблицам (или в терминах реляционной модели - по отношениям), а также определить состав полей (в терминах реляционной теории - атрибутов) для каждой из этих таблиц.

Процесс проектирования является творческим, в немалой степени зависит от опыта и интуиции разработчика. Основными целями при разработке эффективной структуры данных являются:

- обеспечение быстрого доступа к данным в таблицах;
- исключение ненужного повторения данных, которое может являться причиной ошибок при вводе и нерационального использования дискового пространства компьютера;
- обеспечение целостности данных таким образом, чтобы при изменении одних объектов автоматически происходило соответствующее изменение связанных с ними объектов.

Процесс построения эффективной структуры данных для уменьшения избыточности информации в базе данных называется *нормализацией*. В теории нормализации баз данных разработаны достаточно формализованные подходы по разбиению данных, обладающих сложной структурой, среди нескольких таблиц. Теория нормализации оперирует с пятью нормальными формами таблиц. Избыточность информации уменьшается от первой до пятой нормальной формы. Поэтому каждая последующая нормальная форма должна удовлетворять требованиям предыдущей формы и некоторым дополнительным условиям. При практическом проектировании баз данных четвертая и пятая формы, как правило, не используются, поэтому ограничимся рассмотрением первых трех нормальных форм.

В качестве примера рассмотрим таблицу «Продажи», которая содержит информацию о клиентах-предприятиях и их представителях, заказах, проданном товаре. Структура таблицы «Продажи» приведена в таблице 5.1. Эту таблицу можно рассматривать как однотабличную базу данных. Однако в ней содержится значительное количество повторяющейся информации. Например, сведения о каждом клиенте повторяются для каждого сделанного им заказа. Такая структура является причиной следующих проблем, возникающих при работе с базой данных:

- тратится значительное время на ввод повторяющихся данных;
- при изменении адреса или телефона клиента необходимо корректировать все записи, содержащие сведения о его заказах;
- наличие повторяющейся информации приведет к неоправданному увеличению размера базы данных;
- любые внештатные ситуации потребуют значительного времени для получения требуемой информации, например, при поиске ошибок.

Таблица в первой нормальной форме должна удовлетворять следующим требованиям:

а) таблица не должна иметь повторяющихся записей;

б) в таблице должны отсутствовать повторяющиеся группы полей;

в) поля должны быть неделимыми.

Для удовлетворения первого условия каждая таблица должна иметь уникальный ключ. Таблица «Продажи» не содержит уникального ключа, что допускает наличие в ней повторяющихся записей. Для выполнения этого условия создадим уникальный ключ, содержащий поле «Код клиента».

Таблица 5.1 – Структура таблицы «Продажи»

| Наименование | Тип |
|---|---|
| 1 Предприятие | Текстовый |
| 2 ФИО руководителя | Текстовый |
| 3 Телефон руководителя | Числовой |
| 4 Адрес | Текстовый |
| 5 Банковские реквизиты | Текстовый |
| 6 ФИО представителя | Текстовый |
| 7 Паспортные данные представителя | Текстовый |
| 8 Телефон представителя | Числовой |
| 9 Дата заказа | Дата/Время |
| 10 Код товара | Числовой |
| 11Количество | Числовой |
| 12 Дата продажи | Дата/Время |
| 13 Накладная | Числовой |
| 14 Наименование товара | Текстовый |
| 15 Цена | Денежный |
| 16 Характеристики | Текстовый |

Второе требование постулирует устранение повторяющихся групп. Поскольку каждый клиент может сделать несколько заказов, а каждый из которых может содержать несколько товаров, то разобьем таблицу «Продажи» на две таблицы «Клиенты» и «Заказы». Каждая запись первой таблицы будет содержать сведения об одном из клиентов, а второй - информацию о каждом из заказов. Между таблицами будет отношение «один-ко-многим» при помощи поля «Код клиента». Таблица «Заказы» имеет уникальный составной ключ, состоящий из полей «Код клиента», «Код товара» и «Дата заказа».

Неделимость поля означает, что содержащиеся в нем значения не должны делиться на более мелкие. Например, поле, содержащее фамилию, имя и отчество, следует разделить на три поля - отдельно для фамилии, имени и отчества, адрес - на «Индекс», «Город», «Улица_дом». Структуры таблиц «Клиенты» и «Заказы» в первой нормальной форме показаны на рисунке 5.1.

Таблица находится во второй нормальной форме, если:

- она удовлетворяет условиям первой нормальной формы;

- все поля таблицы зависят от первичного ключа, то есть первичный ключ однозначно определяет запись и не является избыточен.

Если все возможные ключи отношения содержат по одному атрибуту, то это отношение задано во второй нормальной форме, так как в этом случае все атрибуты, не являющиеся первичными, полностью зависят от возможных ключей. Приведение отношений ко второй нормальной форме заключается в обеспечении полной функциональной зависимости всех атрибутов от ключа. *Функциональной зависимостью* между полями A и B называется зависимость, при которой каждому значению A в любой момент времени соответствует единственное значение B из возможных. *Полной функциональной зависимостью* между составным полем A и полем B называется зависимость, при которой поле B

зависит функционально от поля А и не зависит функционально от любого подмножества поля А.

Понятие второй нормальной формы применимо только к таблицам, имеющим составной ключ. В рассматриваемом примере такой таблицей является «Заказы», в которой составной ключ образуют поля «Код клиента», «Код товара» и «Дата заказа». Данная таблица не является таблицей во второй нормальной форме, поскольку поля «Наименование товара», «Цена», «Характеристики» однозначно определяются только одним из ключевых полей («Код товара»). Для приведения таблицы ко второй нормальной форме выделим из таблицы «Заказы» таблицу «Товары», которая будет содержать информацию о товарах каждого типа. Для связывания таблиц «Заказы» и «Товары» используется поле «Код товара». Результат представлен на рисунке 5.2.

Таблица «Клиенты»

| Наименование | Тип |
|---|---|
| 1 Код клиента * | Числовой |
| 2 Предприятие | Текстовый |
| 3 Фамилия руководителя | Текстовый |
| 4 Имя | Текстовый |
| 5 Отчество | Текстовый |
| 6 Телефон руководителя | Числовой |
| 7 Индекс | Числовой |
| 8 Город | Текстовый |
| 9 Улица_дом | Текстовый |
| 10 Банковские реквизиты | Текстовый |
| 11 ФИО представителя | Текстовый |
| 12 Паспортные данные | Текстовый |
| 13 Телефон представителя | Числовой |

Таблица «Заказы»

| Наименование | Тип |
|---|---|
| 1 Код клиента * | Числовой |
| 2 Дата заказа * | Дата/Время |
| 3 Код товара * | Числовой |
| 4 Количество | Числовой |
| 5 Дата продажи | Дата/Время |
| 6 Накладная | Числовой |
| 7 Наименование товара | Текстовый |
| 8 Цена | Денежный |
| 9 Характеристики | Текстовый |

Рисунок 5.1 – Первая нормальная форма

Таблица «Клиенты»

| Наименование | Тип |
|---|---|
| 1 Код клиента * | Числовой |
| 2 Предприятие | Текстовый |
| 3 Фамилия руководителя | Текстовый |
| 4 Имя | Текстовый |
| 5 Отчество | Текстовый |
| 6 Телефон руководителя | Числовой |
| 7 Индекс | Числовой |
| 8 Город | Текстовый |
| 9 Улица_дом | Текстовый |
| 10 Банковские реквизиты | Текстовый |
| 11 ФИО представителя | Текстовый |
| 12 Паспортные данные | Текстовый |
| 13 Телефон представителя | Текстовый |

Таблица «Заказы»

| Наименование | Тип |
|---|---|
| 1 Код заказа * | Числовой |
| 2 Дата заказа | Дата/Время |
| 3 Код клиента | Числовой |
| 4 Количество | Числовой |
| 5 Дата продажи | Дата/Время |
| 6 Накладная | Числовой |
| 7 Код товара | Числовой |

Таблица «Товары»

| Наименование | Тип |
|---|---|
| 1 Код товара * | Числовой |
| 2 Наименование товара | Текстовый |
| 3 Цена | Денежный |
| 4 Характеристики | Текстовый |

Рисунок 5.2 – Вторая нормальная форма

Таблица находится в третьей нормальной форме, если:

- она удовлетворяет условиям второй нормальной формы;
- каждый атрибут этого отношения, не являющийся первичным, не транзитивно зависит от каждого возможного ключа этого отношения.

*Транзитивная зависимость* между полями A и C существует в том случае, когда поле C функционально зависит от поля B, а поле B функционально зависит от поля A; при этом не существует функциональной зависимости поля A от поля B. Требование третьей нормальной формы сводится к тому, чтобы все неключевые поля зависели только от первичного ключа и не зависели друг от друга. Сведение таблицы к третьей нормальной форме предполагает разделение таблицы с целью помещения в отдельную таблицу (или несколько таблиц) столбцов, которые не зависят от значения составного индекса. В результате такого разбиения каждое из неключевых полей должно оказаться независимым от какого-либо другого неключевого поля.

В таблице «Клиенты» поля «Паспортные данные» и «Телефон представителя» содержат сведения о представителе предприятия-клиента, которые однозначно определяются значением поля «ФИО представителя». Поскольку неключевые поля «Паспортные данные» и «Телефон представителя» однозначно определяются другим неключевым полем «ФИО представителя», таблица «Клиенты» не является таблицей в третьей нормальной форме. Для приведения этой таблицы к третьей нормальной форме создадим новую таблицу «Представитель» (рисунок 5.3).

После определения структуры таблиц, отношений между ними и совпадающих полей, которые будут использованы для связывания отдельных таблиц, можно создавать многотабличную базу данных в Access.

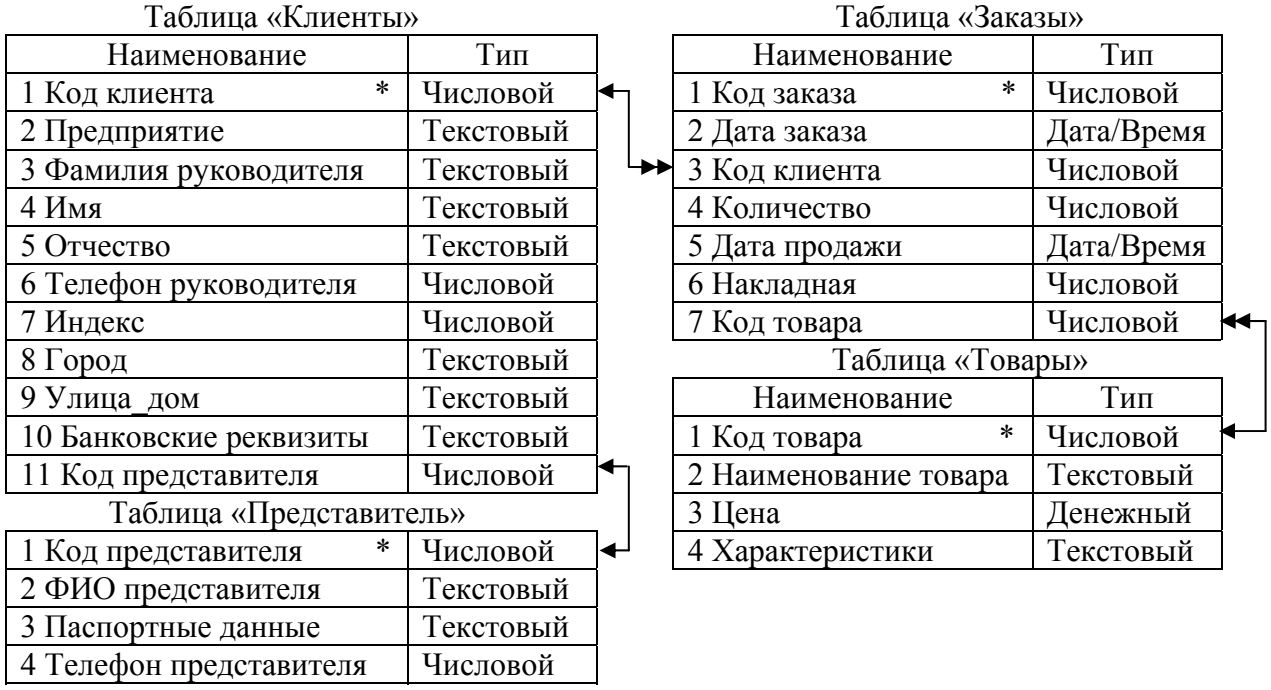



Рисунок 5.3 – Третья нормальная форма

### 5.1.3 Установление связей между таблицами в Access

В Access можно устанавливать постоянные связи между таблицами, которые будут поддерживаться при создании форм, отчетов и запросов. Устанавливая связи между двумя таблицами, необходимо выбрать поле, которое содержит одну и ту же информацию. Поля, с помощью которых устанавливается связь между таблицами, могут иметь различные имена, но удобнее использовать совпадающие имена.

Перед определением связей между таблицами или их удалением необходимо закрыть все открытые таблицы. Создание связей между таблицами в Access осуществляется в окне диалога «Схема данных», которое открывается по команде **Схема данных**  , находящейся на **Ленте** в группе «Показать или скрыть» на вкладке **Работа с базами данных**. Если в базе данных никаких связей не определено, автоматически открывается диалоговое окно «Добавление таблицы». Если оно не отображается, то на вкладке **Конструктор** в группе «Связи» нажать кнопку **Добавить таблицу**. В диалоговом окне «Добавление таблицы» отображаются все таблицы и запросы базы данных. Чтобы просмотреть только таблицы, выбрать пункт **Таблицы**. Чтобы просмотреть только запросы, выбрать пункт **Запросы**. Чтобы просмотреть и таблицы, и запросы, выбрать пункт **Все**.

На рисунке 5.4 показана схема данных для рассмотренной выше базы данных «Продажи».

Для установления связей необходимо:

- в окне «Добавление таблицы» в списке таблиц выделить первую добавляемую таблицу (например, «Клиенты») и нажать кнопку **Добавить**, затем выбрать вторую добавляемую таблицу (например, «Заказы») и также нажать кнопку **Добавить**, затем закрыть окно диалога «Добавление таблицы» кнопкой **Закрыть**; в окне диалога «Схема данных» появятся две связываемые таблицы;
- для связывания таблиц выбрать поле первой связываемой таблицы и переместить его с помощью мыши на соответствующее поле второй таблицы, для связывания сразу нескольких полей выбрать эти поля при нажатой клавише **Ctrl** и переместить во вторую таблицу группу выделенных полей;
- на экране откроется окно диалога «Изменение связей», как показано на рисунке 5.5, в котором необходимо проверить правильность имен связываемых полей, находящихся в столбцах, при необходимости выбрать другие имена полей; затем нажать кнопку **Создать**, по нажатию которой осуществится возврат в окно диалога «Схема данных», где будет отображена связь между таблицами.

В окне диалога «Схема данных» имеется возможность не только создавать связи между таблицами, но и выполнить следующие действия:

- изменить структуру таблицы;
- изменить существующую связь;
- удалить связь;
- удалить таблицу из окна диалога «Схема данных»;

- вывести на экран все существующие связи или связи только для конкретной таблицы;

- определить связи для запросов, не задавая условия целостности данных.

Если при создании связи в окне диалога «Схема данных» возникает необходимость изменения структуры таблицы, то можно, не покидая окна диалога, внести нужные изменения:

- установить указатель мыши на модифицируемую таблицу;

- нажать правую клавишу мыши и выбрать из контекстного меню команду **Конструктор таблиц**;

- внести в структуру таблицы необходимые изменения;

- закончив внесения изменений, нажать кнопку закрытия окна в строке заголовка; в ответ на запрос о сохранении изменений выбрать **Да** для сохранения изменений и возвращения в окно диалога «Схема данных».

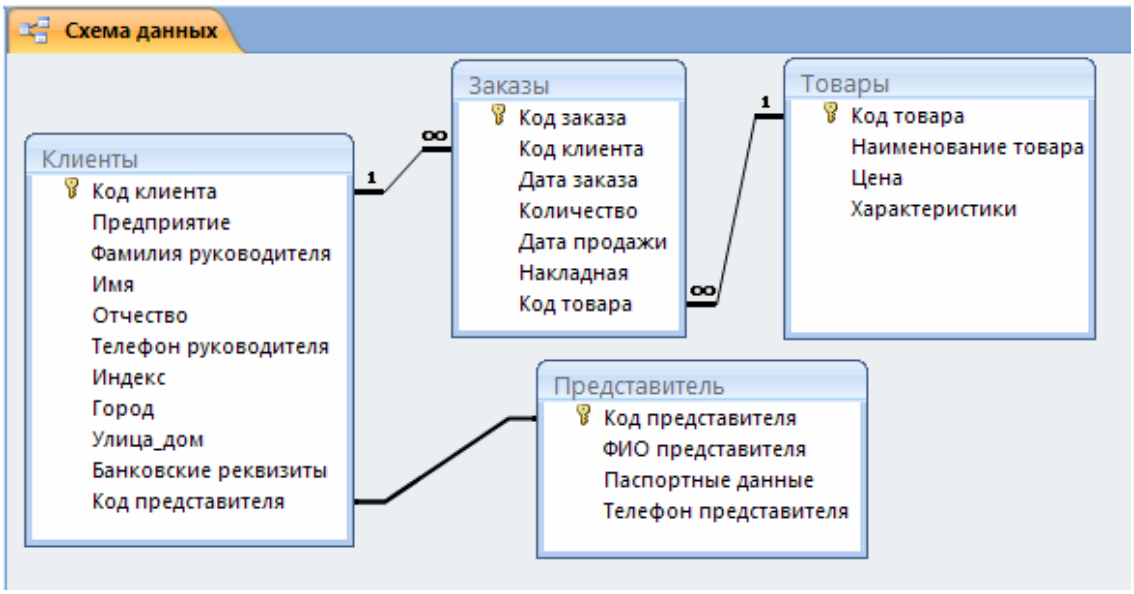



Рисунок 5.4 – Окно диалога «Схема данных» с добавленными таблицами

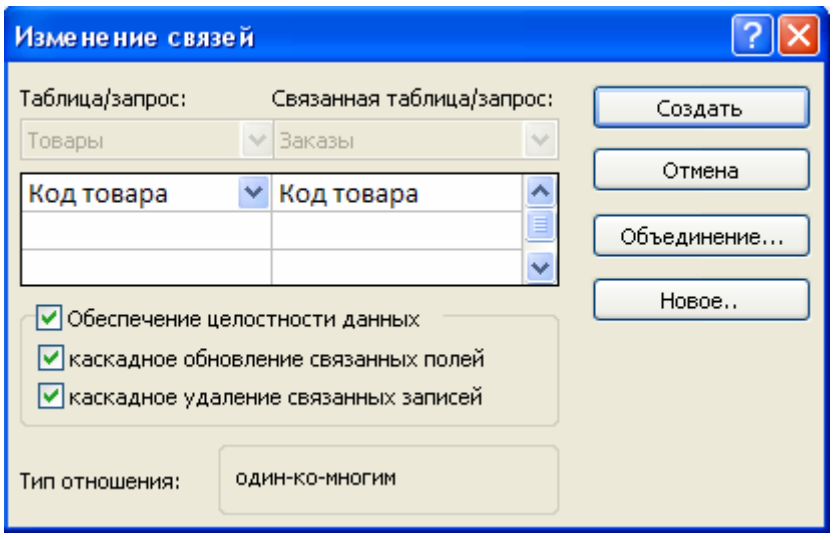



Рисунок 5.5 – Окно диалога «Изменение связей»

Для изменения существующей связи выполняется следующая последовательность действий:

- находясь в окне базы данных, нажать кнопку **Изменить связи** на **Ленте** в группе «Сервис»;

- установить указатель на линию связи, которую требуется изменить, и дважды нажать кнопку мыши;

- в открывшемся окне диалога «Изменение связей» внести нужные изменения и нажать кнопку **ОК.**

Для удаления связи необходимо открыть окно диалога «Схема данных», установить указатель на удаляемую линию связи, выделить ее, нажав кнопку мыши, а затем нажать клавишу **Delete**. Когда Access предложит подтвердить удаление связи, нажать кнопку **Да**.

Для удаления таблицы из макета схемы данных необходимо открыть окно диалога «Схема данных», выбрать удаляемую таблицу и нажать клавишу **Delete**. Таблица будет удалена из макета схемы данных вместе с определенными для нее связями.

В окне диалога «Изменение связей» возможно определение условий целостности данных. Целостность данных является одним из самых важных требований, предъявляемых к базам данных. *Условиями целостности данных* называют набор правил, используемых для поддержания связей между записями в связанных таблицах. Эти правила делают невозможным случайное удаление или изменение связанных данных.

Условия целостности данных выполняются при следующих условиях:

- связанное поле главной таблицы является ключевым полем или имеет уникальный индекс;

- связанные поля имеют один тип данных;

- обе таблицы принадлежат одной базе данных Access.

Невозможно определить условия целостности данных для присоединенных таблиц из баз данных других форматов.

При определении условия целостности данных действуют следующие ограничения:

- невозможно ввести в поле внешнего ключа связанной таблицы значение, не содержащееся в ключевом поле главной таблицы, однако возможен ввод в поле внешнего ключа пустых значений, показывающих, что записи не являются связанными; например, это означает, что нельзя сохранить запись, регистрирующую заказ в таблице «Заказы», сделанный несуществующим клиентом, но можно создать запись для заказа, который пока не отнесен ни к одному из клиентов, если ввести пустое значение в поле «Код клиента»;

- не допускается удаление записи из главной таблицы, если существуют связанные с ней записи в подчиненной таблице; например, невозможно удалить запись из таблицы «Клиенты», если в таблице «Заказы» имеются заказы, относящиеся к данному покупателю;

- невозможно изменить значение ключевого поля в главной таблице, если имеются записи, связанные с этой записью; например, невозможно удалить код клиента в таблице «Клиенты», если в таблице «Заказы» имеются заказы, отно-

сящиеся к данному покупателю.

Определение целостности данных предполагает выполнение следующих действий:

а) в окне диалога «Схема данных» нажать два раза мышью на линии связи между двумя таблицами, откроется окно диалога «Связь»;

б) установить флажок **Обеспечение целостности данных**;

в) нажать **ОК**.

После выполнения указанных действий откроется снова окно диалога «Схема данных». Темная линия между двумя списками полей стала гораздо темнее, и около нее появились два новых символа. Около главной таблицы теперь стоит символ «**1**», который указывает на часть «один» отношения «один-ко-многим». Рядом с подчиненной таблицей отображается символ «∞» (бесконечность), который обозначает часть отношения «много».

Теперь любая попытка выполнить действие, нарушающее перечисленные выше ограничения, приведет к открытию окна диалога с предупреждением, а само действие выполнено не будет.

При установке опции **Обеспечение целостности данных** стали доступны опции **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей**. При выборе этих опций Access выполняет изменения в связанных таблицах таким образом, чтобы сохранить целостность данных, даже если изменяются значения ключевых полей или удаляется запись в главной таблице.

Если опция **Каскадное обновление связанных полей** не установлена, то Access не позволит внести изменения в ключевое поле записи, связанной по типу «один-ко-многим». Вместо изменения будет выдано предупреждение о том, что нарушена целостность данных. Если установлена опция **Каскадное обновление связанных полей**, то Access выполнит все необходимые изменения в связанных таблицах автоматически. Если установлена опция **Каскадное удаление связанных полей**, можно удалять запись из главной таблицы. Связанные с нею записи в подчиненных таблицах будут также автоматически удалены, соблюдая правила целостности данных.

### 5.1.4 Многотабличные запросы

При выборе данных из таблиц часто используются *многотабличные запросы*, поскольку информация в реляционных базах данных содержится не в отдельной таблице, а в совокупности связанных таблиц. Запросы в реляционных базах данных аналогичны обыкновенным запросам. Наличие в Access языка запросов по образцу QBE позволяет задавать многотабличные запросы к базе данных путем заполнения предлагаемой СУБД запросной формы. При включении в запрос связанных таблиц добавление поля в список запроса и задание других свойств окна запроса осуществляется практически также, как при работе над созданием запроса, который основан на одной таблице.

Рассмотрим составление запросов к связанным таблицам с отношением «один-ко-многим» на примере базы данных, схема данных которой приведена на рисунке 5.4. База данных содержит таблицы «Клиенты», «Заказы», «Пред-

ставитель» и «Товары». Пусть необходимо получить информацию о заказах, сделанных после 01.09.2008. Для формирования запроса необходимо выполнить следующее:

- открыть окно конструктора запросов и добавить в него таблицы «Клиенты», «Заказы», предварительно установив между ними связь;

- перенести в бланк запроса из таблицы «Клиенты» поля «Предприятие», «Фамилия руководителя», из таблицы «Заказы» - «Дата заказа», «Количество» и «Накладная» (рисунок 5.6);

- ввести условие отбора «>01.09.2008»;

- выполнить запрос, нажав на кнопку **Выполнить**  , расположенную на **Ленте** в группе «Результаты» закладки **Конструктор**.

Логические **И/ИЛИ** в многотабличных запросах работают точно так же, как и в однотабличных. Условия выбора, образующие выражение для логического **И**, должны располагаться в одной строке, соединенные оператором **And.** При задании выражения для логического **ИЛИ** условие можно расположить в разных строках или воспользоваться оператором **Or**.

Для изменения внешнего вида полученной при выполнении запроса результирующей таблицей можно использовать те же средства, что и для обычных таблиц. Например, можно сделать поля невидимыми, зафиксировать их, изменить шрифт, размеры столбцов и строк. Для выполнения этих функций надо выбрать соответствующую команду из контекстного меню или на **Ленте**.

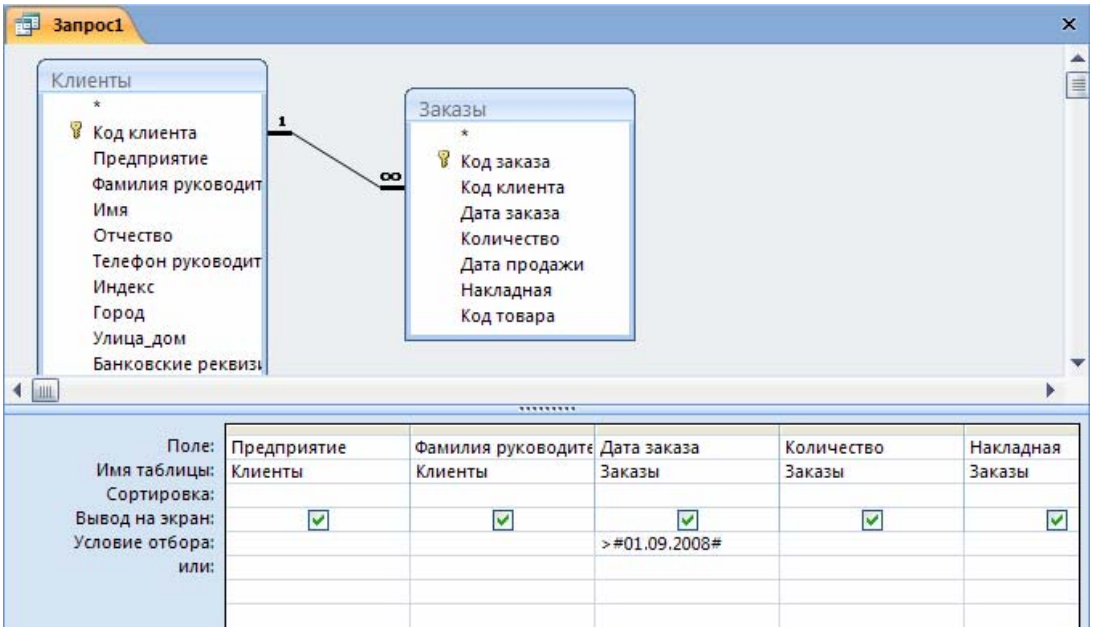



Рисунок 5.6 - Окно конструктора запросов для выборки из двух таблиц

### 5.1.5 Отчеты в реляционных базах данных

Отчет для реляционной базы данных целесообразно создавать на запросах, включающих несколько таблиц. В общем случае можно воспользоваться мастером отчетов для данного типа запроса, как и для любого другого запроса.

При работе с повторяющимися данными полезно создавать групповые отчеты, основанные на запросах, построенных на связи «один-ко-многим».

В качестве примера рассмотрим отчет по клиентам и сделанным ими заказах. Для отчета используется запрос, создание которого показано в пункте 5.1.4 и сохраненного под именем «Запрос_связанные таблицы». Рекомендуется следующая последовательность действий:

- не закрывая запрос, щелкнуть на **Ленте** на закладке **Создание**, затем в группе «Отчеты» выбрать команду **Мастер отчетов**;

- на первом шаге мастера отчетов выбрать включаемые в отчет поля, добавить их в список выбранных полей и щелкнуть **Далее >**;

- на втором шаге убедиться, что таблица «Клиенты» выделена в качестве основы для предоставления данных; затем щелкнуть **Далее >,** на следующем этапе выбора уровня группировки просто щелкнуть по кнопке **Далее >**, поскольку не требуется добавлять подчиненные группы;

- задать сортировку по дате заказа и щелкнуть **Далее >**;

- на следующем шаге выделить переключатель **ступенчатый** в качестве вида макета отчета и **Книжная** для задания ориентации, нажать **Далее>** , выбрать стиль **Нет**, щелкнуть **Далее >;**

- на последнем этапе ввести заголовок отчета, нажать кнопку **Готово.**

Если полученный отчет не удовлетворяет требованиям пользователя, то его можно использовать в качестве чернового наброска и модифицировать с помощью конструктора отчетов. Окончательный вид отчета показан на рисунке 5.7.

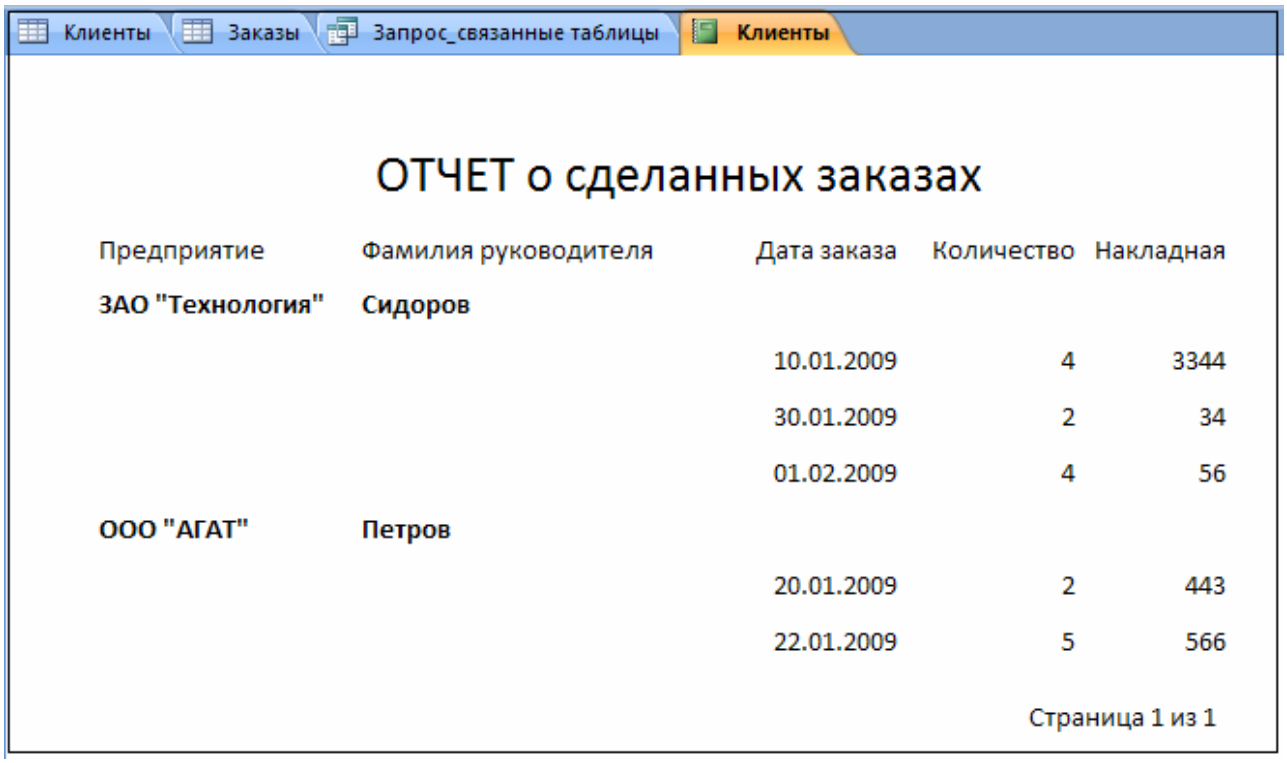



Рисунок 5.7 - Групповой отчет

### 5.1.6 Создание пользовательских форм с подчиненными формами

При необходимости создания формы для просмотра и редактирования данных в двух таблицах, которые находятся в связи «один-ко-многим», и если надо вывести на экран все записи из таблицы «с одним» и связанные с ними записи из таблицы «со многими», лучше всего, как правило, воспользоваться основной формой и подчиненными формами. Основная форма содержит запись из таблицы «с одним», а подчиненная форма содержит только связанные с ней записи из таблицы «со многими», Поскольку акцент делается на записи из таблицы «с одним», не имеет значения, если вдруг не найдется соответствующей записи из таблицы «со многими».

Мастер форм облегчает создание этого вида форм. На первом этапе необходимо включить в форму поля из обеих таблиц. Второй шаг имеет опцию, позволяющую выбрать группировку для формы: при группировке по таблице «с одним» можно создать форму с подчиненной формой или связанными формами. Если выделить переключатель «Подчиненные формы», то мастер форм создают одну форму с полями из таблицы «с одним» в верхней части и соответствующими полями из таблицы «со многими», которые перечислены в нижней ее части. Если выбрать переключатель «Связанные формы», то мастер форм создаст форму с полями из таблицы «с одним», которая будет иметь кнопку, щелкнув на которой можно вывести на экран вторую форму с соответствующими полями таблицы «со многими». Это свойство используется в случаях, когда необходимо отобразить большое количество полей, которые не войдут полностью в одну форму. В обоих случаях мастер форм позволяет в завершение ввести два заголовка, один из них - для главной формы, а другой – для подчиненной или связанной формы. А также в обоих случаях мастер форм связывает таблицы при помощи связи, устанавливаемой по умолчанию, которую необходимо определить заранее.

В качестве примера рассмотрим создание формы с подчиненной формой для базы данных, состоящей из двух таблиц «Клиенты» и «Заказы». Для создания формы «Клиенты», представленной на рисунке 5.8, необходимо выполнить следующие действия:

- щелкнуть на **Ленте** на закладке **Создание**, а затем выбрать команду **Мастер форм** в раскрывающемся списке **Другие формы** группы «Формы», на экране появится диалоговое окно «Создание форм»;

- на первом шаге мастера форм выбрать таблицу «Клиенты» из раскрывающего списка **Таблицы / Запросы** и добавить поля «Код клиента», «Предприятие», «Фамилия руководителя», «Имя», «Отчество», «Телефон руководителя», «Город» в список **Выбранные поля**; затем выбрать таблицу «Заказы» из того же раскрывшегося списка **Таблицы / Запросы** и добавить поля «Код заказа», «Дата заказа», «Количество», «Дата продажи», «Накладная» в список полей **Выбранные поля**, затем щелкнуть на кнопке **Далее >**;

- на следующем шаге убедиться, что установлен переключатель **Подчиненные формы**, после чего щелкнуть **Далее >**; в следующем окне диалога оставить установленное по умолчанию значение «табличный» для внешнего вида

подчиненной формы и щелкнуть **Далее >,** затем выбрать стиль и щелкнуть **Далее >;**

- задать имя созданной форме, выбрать последующие действия и нажать **Готово.**

Access выведет на экран новую форму. Данная форма имеет два комплекта кнопок со стрелками и два номера записей, отображаемых одновременно: один для главной формы, другой для подчиненной. Работа с такой формой не вызывает сложностей. Главная форма используется для просмотра, добавления или редактирования данных в таблице «Клиенты» также, как и другими формами, которые для нее можно создать. Подчиненную форму можно использовать для просмотра, добавления и редактирования данных для конкретного клиента в таблице «Заказы», как при работе с любой таблицей.

Вид созданной формы после сделанных изменений в режиме конструктора приведен на рисунке 5.8.

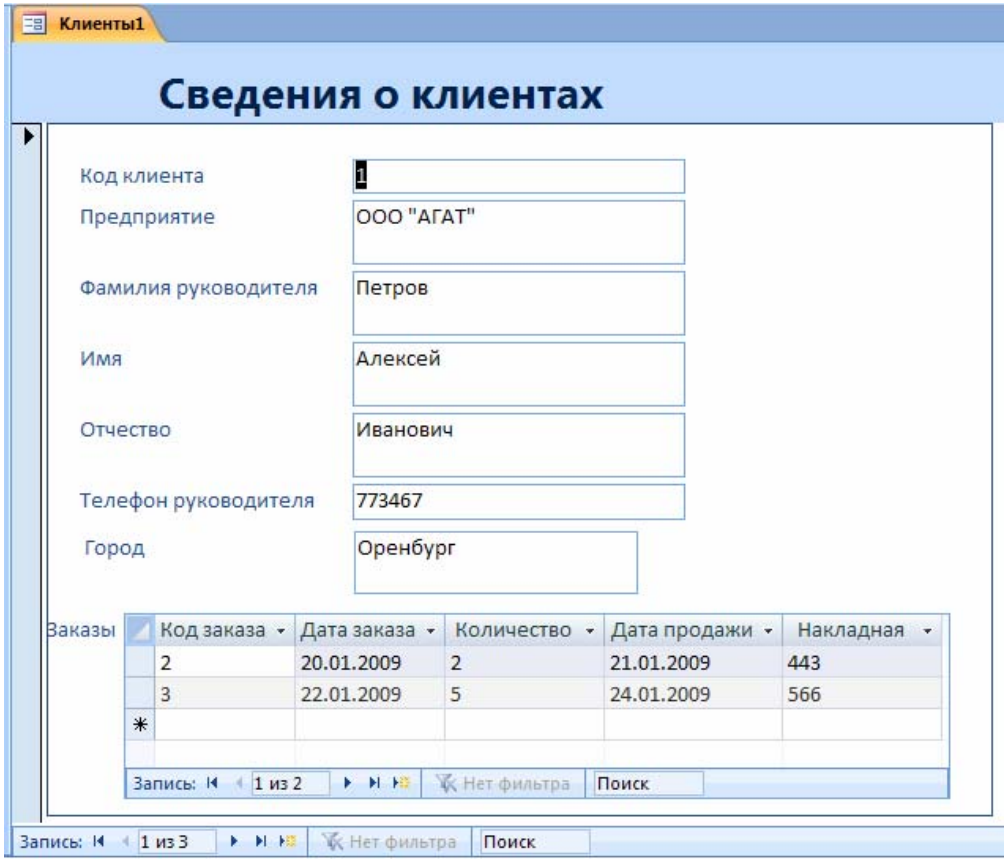



Рисунок 5.8 – Форма и подчиненная форма

## 5.2 Задание на выполнение работы

5.2.1 Запустить Access, создать новую базу данных «Продажи», сохранить ее на диске D:/ в папке, имеющей имя группы.

5.2.2 Создать таблицы «Клиенты», «Заказы», «Представитель», «Товары», входящие в базу данных «Продажи», схема данных которой приведена на рисунке 5.6.

5.2.3 Осуществить ввод данных в созданные таблицы.

5.2.4 Установить связи между таблицами в окне диалога «Схема данных», определить условия целостности данных.

5.2.5 Осуществить многотабличные запросы к связанным таблицам, задавая различные условия отбора; сохранить их.

5.2.6 Создать отчеты на основе запросов, выполненных в пункте 5.2.5.

5.2.7 Создать пользовательскую форму с подчиненной формой.

5.2.8 Выполнить нормализацию базы данных «Отношения между работодателями и подрядчиками», структура которой представлена в таблице 5.2.

5.2.9 Оформить отчет по лабораторной работе.

### 5.3 Содержание отчета

5.3.1 Название работы.

5.3.2 Цель работы.

5.3.3 Структура данных базы данных «Продажи» в третьей нормальной форме с указанием типа связей между таблицами.

5.3.4 Вид бланков запроса на языке QBE при различных заданных условиях отбора в многотабличных запросах, выполняемых в пункте 5.2.5.

5.3.5 Вид отчета, полученного в пункте 5.2.6.

5.3.6 Макет пользовательской формы с подчиненной формой, которая создавалась при выполнении пункта 5.2.7.

5.3.7 Структура базы данных «Отношения между работодателями и подрядчиками» после проведения нормализации.

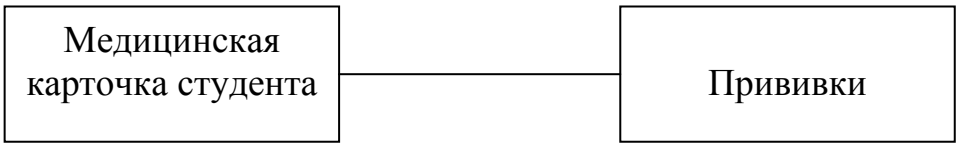
Таблица 5.2 – Отношения между работодателями и подрядчиками

| Наименование | Тип |
|---|---|
| 1 Код сотрудника | Числовой |
| 2 Имя | Текстовый |
| 3 Отчество | Текстовый |
| 4 Фамилия | Текстовый |
| 5 Дата рождения | Дата/время |
| 6 Адрес домашний | Текстовый |
| 7 Телефон домашний | Текстовый |
| 8 Должность | Текстовый |
| 9 Номер счета | Текстовый |
| 10 Дата счета | Дата/время |
| 11 Отработанные часы | Числовой |
| 12 Почасовая ставка | Денежный |
| 13 Содержание работы | Поле Memo |
| 14 Код работодателя | Числовой |
| 15 Адрес предприятия | Текстовый |
| 16 Наименование предприятия | Текстовый |
| 17 Руководитель | Текстовый |
| 18 Телефон предприятия | Текстовый |

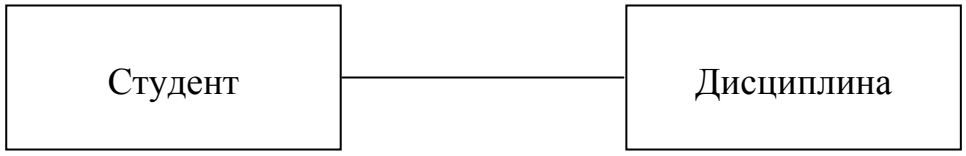
## 5.4 Тесты и контрольные вопросы

5.4.1 Строки таблицы соответствуют:
а) объекту;
б) атрибуту;
в) характеристике;
г) признаку.
5.4.2 Поставьте в соответствие схемы взаимосвязи таблиц и виды связей:

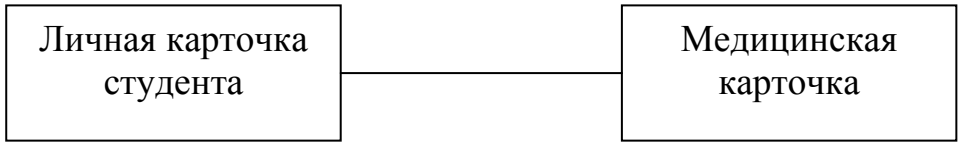
1)




2)




3)




1) M : M
2) 1 : 1
3) 1 : M

а) 1 - 1, 2 - 2, 3 – 3;
б) 1 - 2, 2 - 3, 3 – 1;
в) 1 - 3, 2 - 2, 3 – 1;
г) 1 - 3, 2 - 3, 3 – 2;
д) 1 - 3, 2 - 1, 3 – 2.
5.4.3 Связи между таблицами реляционной базы данных позволяют:
а) избежать дублирования информации;
б) определить местонахождение нужной таблицы;
в) производить сортировку таблицы;
г) распечатать данные таблицы;
д) удалить таблицы.
5.4.4 Какие связи не допускают реляционные многотабличные БД:
а) многие — к одному;
б) один — ко многим;
в) один — к одному;

г) многие – ко многим.

5.4.5 Между двумя реляционными таблицами могут быть сформированы связи, если они имеют:

а) одинаковое имя;

б) одинаковое количество столбцов;

в) одинаковое количество строк;

г) общее поле данных;

д) одинаковые записи.

5.4.6 Основная цель проектирования - разработать эффективную структуру данных, что позволяет:

а) обеспечить быстрый доступ к данным в таблице;

б) производить анализ средств автоматизации;

в) осуществить ограниченный доступ к информации;

г) нерационально использовать память.

5.4.7 Нормализация базы данных – это:

а) создание эффективной структуры данных;

б) арифметическая операция над данными;

в) обеспечение секретности данных;

г) выполнение запросов в базе данных;

д) обеспечение независимости данных.

5.4.8 Поставьте в соответствие нормальную форму и требование, предъявляемое к ней:

1) первая нормальная форма
2) вторая нормальная форма
3) третья нормальная форма

1) каждый атрибут таблицы, не являющийся ключевым, не транзитивно зависит от каждого возможного ключа этого отношения
2) поля должны быть неделимыми
3) поля должны зависеть от первичного ключа

а) 1 - 1, 2 - 2, 3 – 3;

б) 1 - 2, 2 - 3, 3 – 1;

в) 1 - 3, 2 - 2, 3 – 1;

г) 1 - 3, 2 - 3, 3 – 2;

д) 1 - 3, 2 - 1, 3 – 2.

5.4.9 К требованию к третьей нормальной форме таблицы относится:

а) поля должны быть неделимыми;

б) не должно быть повторяющихся записей;

в) каждый атрибут таблицы, не являющийся ключевым, не транзитивно зависит от каждого возможного ключа этого отношения;

г) таблица должна содержать, как минимум три ключевых поля.

5.4.10 Зависимость между полями A и C, при которой поле C функционально зависит от поля B, а поле B функционально зависит от поля A; при этом не существует функциональной зависимости поля A от поля B, называется:

а) полной функциональной зависимостью;

б) зависимостью соединения;

в) транзитивной зависимостью;

г) многозначной зависимостью.

5.4.11 Из чего состоит реляционная база данных?

5.4.12 Какие связи могут существовать между отдельными таблицами в базе данных?

5.4.13 Что такое «первичный ключ»?

5.4.14 Что означает отношение «один-к-одному»? Приведите пример.

5.4.15 Что означает отношение «многие-к-одному»? Приведите пример.

5.4.16 Что означает отношение «многие-ко-многим»? Приведите пример.

5.4.17 Каковы цели разработки эффективной структуры данных?

5.4.18 Что называется «нормализацией»?

5.4.19 Какие требования предъявляются к первой нормальной форме базы данных?

5.4.20 Какие требования предъявляются ко второй нормальной форме?

5.4.21 Какие требования предъявляются к третьей нормальной форме?

5.4.22 Что такое «транзитивная зависимость»?

5.4.23 Как в Access устанавливаются связи между таблицами?

5.4.24 Что можно осуществить в диалоговом окне «Схема данных»?

5.4.25 Какие действия можно выполнить в окне диалога «Связи»?

5.4.26 Как можно изменить существующую связь?

5.4.27 Что называют «условиями целостности данных»?

5.4.28 Какие ограничения действуют при определении условий целостности данных?

5.4.29 Для чего предназначены каскадные операции?

5.4.30 Как осуществляются многотабличные запросы?

5.5.31 Как формируются отчеты в реляционных базах данных?

5.4.32 Что необходимо задать при работе мастера форм для создания подчиненных форм?

5.4.33 Для чего создаются пользовательские формы с подчиненными формами?

5.4.34 Что содержит пользовательская форма с подчиненной формой?

5.4.35 Чем отличается пользовательская форма с подчиненными формами от обычной формы?

# 6 Создание запросов в Access с помощью SQL

Целью работы является ознакомление с синтаксисом структурированного языка запросов SQL в Microsoft Office Access 2007и создание запросов.

## 6.1 Общие положения

### 6.1.1 Структурированный язык запросов SQL

Рост количества данных, необходимость их хранить и обрабатывать привели к тому, что возникла потребность в создании стандартного языка баз данных, который мог бы функционировать в большом количестве разных видов компьютерных систем. Такой стандартный язык позволяет пользователям манипулировать данными независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ.

SQL (Structured Query Language) – это сокращенное название структурированного языка запросов, предоставляющего средства создания и обработки данных в реляционных базах данных (БД). Независимость от специфики компьютерных технологий, а также поддержка SQL лидерами промышленности в области технологии реляционных баз данных сделали его основным стандартным языком баз данных [4, 12].

Язык SQL является основой многих СУБД, так как он отвечает за физическое структурирование и запись данных на диск, а также за физическое чтение данных с диска и позволяет принимать SQL-запросы от других компонентов СУБД и пользовательских приложений. SQL является мощным инструментом, который обеспечивает пользователям, программам и вычислительным системам доступ к информации, содержащейся в реляционных БД.

Основные достоинства языка SQL заключаются в следующем:

а) стандартность языка SQL – его использование в программах стандартизировано международными организациями;

б) независимость от конкретных СУБД – все распространенные СУБД используют SQL, так как реляционную БД и программы, которые с ней работают, можно перенести с одной СУБД на другую с минимальными доработками;

в) возможность переноса с одной вычислительной системы на другую;

г) реляционная основа языка;

д) возможность создания интерактивных запросов – SQL обеспечивает пользователям немедленный доступ к данным, при этом в интерактивном режиме можно получить результат запроса за очень короткое время без написания сложной программы;

е) возможность программного доступа к БД;

ж) обеспечение различного представления данных – с помощью SQL можно предусмотреть такую структуру данных, что пользователи будут видеть различные представления данных;

и) возможность динамического изменения и расширения структуры БД; SQL обеспечивает гибкость с точки зрения приспособленности БД к изменяю-

щимся требованиям предметной области, не прерывая при этом работу приложения.

Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление, а также некоторых сопутствующих операций. SQL является непроцедурным языком и не содержит операторов управления, организации программ, ввода-вывода.

SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД (например, FoxPro СУБД Visual FoxPro, ObjectPASCAL СУБД Paradox, Visual Basic for Applications СУБД Access).

В современных СУБД с интерактивным интерфейсом можно создавать запросы, используя другие средства, например, QBE (язык запросов по образцу, позволяющий подготавливать запросы в наглядной форме). Однако применение SQL зачастую позволяет повысить эффективность обработки данных в базе. Например, при подготовке запроса в среде Access можно перейти из окна конструктора запросов в окно с эквивалентным оператором SQL. Подготовку нового запроса путем редактирования уже имеющегося в ряде случаев проще выполнить путем изменения оператора SQL. В различных СУБД состав операторов SQL может несколько отличаться.

### 6.1.2 Особенности запросов SQL

*Запросом SQL* называют запрос, создаваемый с помощью инструкции SQL. Примерами запросов SQL являются запросы на объединение, запросы к серверу, управляющие и подчиненные запросы.

*Запрос на объединение* – это такой запрос, в котором объединяются поля (столбцы) одной или нескольких таблиц или запросов в одно поле или столбец в результирующем наборе записей. Например, в пяти цехах предприятия установлено станочное оборудование. Создав запрос на объединение, можно объединить списки оборудования в результирующем наборе записей, а затем разработать запрос на создание таблицы, основанный на запросе на объединение.

*Запрос к серверу* - выполняет передачу команд SQL-серверу. Запросы к серверу позволяют непосредственно работать с таблицами на сервере вместо их присоединения. Результатом выполнения запроса к серверу может быть загрузка записей или изменение данных.

*Управляющий запрос* – создает или изменяет объекты базы данных, такие как Access или SQL Server.

*Подчиненный запрос* – состоит из инструкции SQL **SELECT**, находящейся внутри другого запроса на выборку или запроса на изменение. Эти инструкции вводятся в строку «Поле» бланка запроса для определения нового поля или в строку «Условие отбора» для определения условий отбора поля. Подчиненные запросы используются для выполнения следующих действий:

- проверки в подчиненном запросе существования некоторых результатов с помощью зарезервированных слов EXISTS или NOT EXISTS;

- поиск в главном запросе любых значений, которые равны, больше или меньше значений, возвращаемых в подчиненном запросе (с помощью зарезервированных слов ANY, IN или ALL);

- создание подчиненных запросов внутри подчиненных запросов (вложенных подчиненных запросов).

Язык SQL в Access может применяться при разработке экранных форм, отчетов, а также при создании макрокоманд и программ на Visual Basic for Application (VBA). В Access для создания запросов используется также язык QBE. Между языками QBE и SQL имеется тесная связь. Запросные таблицы (бланки, формы) на языке QBE, заполняемые пользователем, перед непосредственным выполнением преобразуются в выражения (или сообщения) SQL. То есть язык SQL является внутренним стандартом на выполнение запросов. Такой механизм имеет преимущество, поскольку позволяет внутри системы Access унифицировать подготовку запросов к выполнению на локальном и на удаленном компьютерах.

В Access запрос может находиться в одном из трех режимов (состояний): конструктора, SQL и таблицы. *Режим конструктора* применяют для разработки нового запроса с чистого листа (без использования мастеров или других средств) или для изменения макета существующего запроса. *Режим SQL* применяют для ввода или просмотра инструкций SQL. *Режим таблицы* применяют для работы с результатами выполнения запроса.

В режим таблицы запрос переходит при выборе нужного запроса в области переходов, для удобства работы предварительно сгруппировав все объекты Access по типам. Перейти в режим конструктора или режим SQL можно, выбрав соответствующий режим из раскрывающегося списка команды **Вид** с вкладки **Главная** группы «Представления».

Операторы языка SQL можно условно разделить на два подязыка:

- язык определения данных, к нему относятся операторы **CREATE TABLE** (оператор создания таблицы), **ALTER TABLE** (оператор изменения структуры таблицы), **DROP TABLE** (оператор удаления таблицы), **CREATE VIEW** (оператор создания представления) и другие;

- язык манипулирования данными, основным оператором которого является оператор **SELECT** (оператор выборки записей).

В таблице 6.1 приведены соглашения, используемые при описании синтаксиса SQL.

### 6.1.3 Синтаксис оператора SELECT

Оператор **SELECT** является ядром языка SQL. Он используется для отбора строк и столбцов из таблиц базы данных. При выполнении данного оператора Microsoft Jet находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке. Инструкция SELECT не изменяет данные в базе данных. Она содержит пять основных предложений.

В общем случае синтаксис можно представить в следующем виде:

**SELECT** *<предложение>*
**FROM** *<список таблиц>*
[**WHERE** *<предложение>*]
[**GROUP BY** *<предложение>*]
[**HAVING** *<предложение>*]
[**ORDER BY** *<предложение>*]

Обязательными являются предложения **SELECT** и **FROM**. Все компоненты команды **SELECT**, если они используются, должны быть записаны в том порядке, в котором они перечислены в формате команды.

Предложение **SELECT** определяет состав результирующего набора данных, эти поля могут принадлежать разным таблицам. Оно имеет следующий формат:

**SELECT** [DISTINCT| DISTINCTROW|ALL|TOP *n* [PERCENT]]
[ *|< *список выражений*> ]

Предикат DISTINCTROW позволяет отбирать строки из источников данных, включенных в *список таблиц*, с различающимися значениями первичных ключей в таблицах, из которых поступают столбцы в *список-полей*. Если указан предикат ALL, то выдаются все строки, в том числе повторяющиеся. Так как по умолчанию принимается значение ALL, то оно в явном виде может не указываться. Предикат DISTINCT требует, чтобы запрос возвратил только строки, отличающиеся от всех остальных.

Таблица 6.1 – Описание синтаксиса SQL

| Соглашение SQL | Толкование |
|---|---|
| ПРОПИСНЫЕ БУКВЫ | Прописными буквами набраны ключевые и зарезервированные слова, которые должны быть введены с точностью до регистра букв |
| *Курсив* | Слова, набранные курсивом, соответствуют переменным, которые задаются пользователем |
| Угловые скобки < > | В угловые скобки заключается обязательный элемент синтаксиса. Текст внутри угловых скобок характеризует элементы, однако не описывает его синтаксис. Угловые скобки не вводятся. |
| Квадратные скобки [ ] | В квадратные скобки заключаются один или несколько необязательных элементов, разделенных символом <<вертикальная черта>> (\|). Необходимо выбрать один, либо ни одного из перечисленных элементов. Квадратные скобки и вертикальная черта не вводятся. |
| Фигурные скобки { } | В фигурные скобки заключаются один или несколько элементов, разделенных символом <<вертикальная черта>> (\|). Необходимо выбрать один из перечисленных элементов. Фигурные скобки и вертикальная черта не вводятся. |
| Многоточие … | Многоточие показывает, что можно повторить некоторый элемент один или несколько раз. Если в описании вместе с многоточием присутствует запятая, то ее необходимо вводить между элементами. |

Чтобы результирующий набор содержал только первые *n* или первые *n* процентов записей, необходимо использовать «TOP n» или «TOP n PERCENT». Параметр должен быть целым числом, не превышающий 100, если используется ключевое слово PERCENT. Для каждой таблицы и запроса при желании можно определить альтернативное имя (псевдоним). Псевдоним используется при задании имен столбцов в списке полей, предложении **WHERE** и в подчиненных предложениях вместо полного имени таблицы, имени запроса или для ссылки на инструкцию выбора.

Знак * указывается, если в ответ выводятся все столбцы в том порядке, в котором они были заданы при описании таблицы.

В списке выражений перечисляются идентификаторы столбцов, которые будут выведены в ответ. В выражение можно включать имя столбца только той таблицы или запроса, которые указаны в предложении FROM. Если имя таблицы, встречается в нескольких таблицах или запросах данной инструкции, его необходимо полностью идентифицировать, указав также имя таблицы, запроса или псевдонима. Может быть использован следующий синтаксис для < *список выражений* > :

<*выражение*> [AS *имя – возвращаемого - столбца*]| *имя - таблицы*. * | *имя – запроса*. * |*псевдоним*. *

Выражение определяет некоторое значение в предикате или в списке полей инструкции **SELECT**. При построении выражения может быть задана только одна итоговая функция. В Access применяют следующие итоговые функции:

- AVG (возвращает среднее арифметическое значение выражения или заданного поля);
- COUNT (возвращает значение, равное числу строк в результирующей таблице);
- MAX (возвращает максимальное значение выражения или заданного поля);
- MIN (возвращает минимальное значение выражения или заданного поля);
- STDEV, STDEVP (возвращает корень квадратный из дисперсии значений выражений или заданного поля);
- SUM (возвращает сумму значений выражения и ли заданного поля);
- VAR (возвращает дисперсию значений выражения или заданного поля).

В выражении могут быть заданы различные формулы, по которым вычисляются значения поля в запросе. В арифметическом выражении должны использоваться поля, содержащие числовые значения. Перед составным выражением указывается функция CCur. Круглые скобки используются для уточнения порядка вычислений.

### 6.1.4 Предложение FROM

Предложение **FROM** задает таблицы или запросы, служащие источником данных для создаваемого запроса. Оно имеет синтаксис

**FROM** {*имя – таблицы* [[AS] *псевдоним* | *имя – запрос –на – выборку* [[**AS** ] *псевдоним*]| <*таблица - объединения*>}, …
где <*таблица - объединения*>:
({*имя – таблицы* [[AS] *псевдоним*]|
*имя – запроса- на- выборку* [[AS] *псевдоним*]|
<*таблица - объединения*>} {INNER | LEFT | RIGHT} YOIN
{*имя – таблицы* [[AS] *псевдоним*]|
*имя- запроса – на- выборку* [[AS] *псевдоним*]|
<*таблица - объединения*>} ON <*условие объединения*>).

В предложении **FROM** может быть указано только <*имя - таблицы*>, из которой выбираются записи.

Обширные возможности SQL во многом основаны на способности этого языка объединять информацию из нескольких таблиц или запросов и представлять результат в виде единого логического набора записей. В большинстве случаев Access позволяет обновлять набор записей запроса на объединение, как если бы это была отдельная базовая таблица.

Для задания типа объединения таблиц в логический набор записей, из которого будет выбираться необходимая информация, используется в предложении **FROM** операция YOIN. Можно включить в логический набор записей только соответствующие строки обоих таблиц (так называемое *внутреннее объединение – inner join*) или включить все строки одной из двух заданных таблиц, даже если соответствующие строки не найдены во второй таблице (*внешнее объединение – outer join*). Можно использовать вложенные операции YOIN, например, объединить с результатом объединения двух таблиц третью.

Используя операцию INNER YOIN, можно получить все сроки из обеих логических таблиц, удовлетворяющих условию объединения (*логическая таблица* – это любая таблица, запрос или таблица объединения). Операция LEFT YOIN возвращает все строки из первой логической таблицы, объединенные с теми строками из второй, для которых выполняется условие объединения. Если во второй логической таблице нет таких строк, Access возвращает значение Null в столбцах второй таблице. Аналогично, операция RIGHT YOIN возвращает все строки из второй логической таблицы, объединенные с теми строками из первой таблицы, для которых выполняется условие объединения.

Если в условии объединения используется только оператор *равно* (=), результат называется *объединением по равенству*. В бланке запроса можно задать лишь его. Запрос на объединение таблиц по неравенству (< , > ,< >, < = или > = ) можно создать только в режиме SQL.

### 6.1.5 Предложение WHERE

Предложение **WHERE** задает условие отбора в инструкции или предложении SQL. Инструкции **DELETE**, **SELECT** и **UPDATE** и подчиненный запрос, содержащие предложение WHERE, воздействует только на те строки, которые удовлетворяют условию отбора.

Синтаксис предложения:

**WHERE** < *условие отбора* >

Access применяет условие отбора к каждой строке логической таблицы, полученной в результате выполнения предшествующих предложений, и отвергает те, для которых условие отбора не принимает значения TRUE.

Если в условии отбора используется подчиненный запрос внутри предиката (подчиненный запрос в этом случае часто называют *внутренним запросом*), то Access сначала выполнит подчиненный запрос и только потом определяет значение предиката. Если подчиненный вопрос соглашается на таблицу или запрос, используемый во внешнем предложении **FROM** (в таком случае его обычно называют *связанным подчиненным запросом*), то Access выполняет подчиненный запрос для каждой обрабатываемой строки внешней таблицы. Если в подчиненном запросе нет ссылок на внешнюю таблицу, но Access выполняет его только один раз. Связанный подчиненный запрос может быть записан как запрос на объединение, который обычно более эффективен.

В предложении **WHERE** можно установить несколько условий, которым должны удовлетворять поля записей, в этом случае используются логические связи. Логические операторы в условии отбора выполняются в следующем порядке: NOT, AND, OR, XOR (исключение OR), EQV (равенство) и IMP (импликация). Для изменения порядка вычисления логических выражений можно использовать круглые скобки. В выражениях в условии отбора могут применяться итоговые функции AVG, COUNT, MAX, MIN и другие.

Например, в запросе «Товары с ценой выше средней» строится связанный подчиненный запрос. Имеется ссылка на таблицу «Товары», используемую во внешнем предложении FROM. В условии отбора применено предложение SELECT с итоговой функцией AVG, возвращающей среднее арифметическое значение заданного поля «Цена». Поэтому в предложении WHERE используется следующее выражение: WHERE ((( Товары. Цена) > (SELECT AVG ([Цена]) FROM Товары))).

В выражение условия отбора может входить предикат BETWEEN, который сравнивает значение с заданным диапазоном. Он имеет синтаксис

< *выражение* > [NOT] BETWEEN < *выражение* > AND < *выражение* >.

Типы данных выражений должны быть совместимы. Сравнение буквенно–цифровых литералов (строк) в Access производится без учета регистра.

Например, в запросе «Продажи по типам» для задания условия поиска заказов, размещенных в 2008 году используется следующее предложение WHERE:

```
WHERE (((Заказы. Дата размещения) Between #01/01/08# AND
 #31/12/08# ))
```

### 6.1.6 Предложение GROUP BY

Предложение **GROUP BY** в команде **SELECT** задает столбцы, используемые для формирования групп из выбранных строк. Строки каждой группы содержат одно и тоже значение заданного столбца (столбцов). В Access пред-

ложение используется для создания итоговых запросов. Оно имеет следующий синтаксис

**GROUP BY** *<имя – столбца, …>*

Имя столбца в предложении **GROUP BY** может быть именем произвольного столбца из любой таблицы, упомянутой в предложении **FROM**, - даже если этот столбец не содержится в списке полей инструкции **SELECT**. Если предложение **GROUP BY** расположено после предложения **WHERE**, Access создает группы из строк, выбранных после применения предложения **WHERE**. При включении предложения **GROUP BY** в инструкцию **SELECT** список полей должен состоять из итоговых функций SQL (AVG, COUNT, MAX, MIN, STDEV, STDEVP, SUM, VAR, VARP) или из имен столбцов, указанных в предложении **GROUP BY**.

Группировка может иметь несколько уровней. Например, в запросе «Продажи по типам» она выполняется по полям «Код типа», «Категория», «Марка».

### 6.1.7 Предложение ORDER BY

Предложение **ORDER BY** задает порядок расположения строк, возвращаемых инструкцией SELECT или INSERT. Оно имеет синтаксис

**ORDER BY** {*имя – столбца*| *номер столбца* [ ASC | DESC]}, …

Для задания столбца, по значениям которого упорядочиваются возвращаемые строки, можно использовать имя столбца или его относительный порядковый номер в наборе записей запроса (первый возвращаемый столбец имеет номер 1). В предложение **ORDER BY** можно указать несколько столбцов. Список сортируется сначала по значениям столбца, имя которого указано первым. Строки с равными значениями в этом столбце упорядочиваются по значениям столбца, имя которого находится на второй позиции в списке предложения **ORDER BY**. Для каждого столбца можно задать порядок сортировки по возрастанию (ASC) или по убыванию (DESC). Если порядок не указан, по умолчанию применяется сортировка по возрастанию. Использование предложения **ORDER BY** в инструкции SELECT - единственный список задания последовательности, в которой располагаются возвращаемые записи.

Например, в запросе «Десять самых дорогих товаров» необходимо выполнять сортировку по убыванию значений поля «Цена». Тогда предложение **ORDER BY** будет выглядеть следующим образом: «ORDER BY Товары. Цена Desc».

## 6.2 Задание на выполнение работы

6.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к лабораторной работе.

6.2.2 Создать таблицы «Поставщики», «Типы товаров», «Товар», «Склад» в соответствии с таблицами 6.2 - 6.5. В диалоговом окне «Схема данных» установить связи между таблицами.

6.2.3 Создать SQL – запросы, которые определяют:

- пять самых дешевых товаров;
- товары с ценой ниже средней;
- число товаров, поступивших из ТЦ «Джаз»;
- перечень имеющихся товаров на складе и их стоимость.

Формулировку созданных SQL - запросов записать в тетрадь.

### 6.3 Содержание отчёта

6.3.1 Название работы.

6.3.2 Цель работы.

6.3.3 Логическая схема базы данных «Поступление товаров».

6.3.4 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

6.3.5 Формулировки SQL-запросов, созданные по заданию 6.2.3.

Таблица 6.2 - Поставщики

| Код поставщика | Наименование | Адрес | Телефон | WWW- адрес |
|---|---|---|---|---|
| 1 | Компьютерный салон «Центр» | ул. Пролетарская, 42 | 774711 | http://www. kscenter.ru |
| 2 | Компьютерный магазин «Байт» | ул. Дзержинского, 20 | 369582 | http://www. mbyte.ru |
| 3 | ООО «Компания «Мехатроника» | ул. Гая, 5 | 780757 | http://www.Mtron.ru |
| 4 | Компьютерный салон «Глюк» | ул. Володарского, 10 | 777665 | http://www.Gluckol.net |
| 5 | ТЦ «Джаз» | ул. Володарского, 27 | 770268 | http://www.jazz-comtutor.nm.ru |
| 6 | ООО «Константа - Сервис» | пр-т Парковый, 46 | 724364 | http://www.Constanta.ru |

Таблица 6.3 - Типы товаров

| Код типа | Наименование | Производитель |
|---|---|---|
| 10 | Монитор | Samsung |
| 15 | Монитор | LCD Acer |
| 20 | Монитор | LCD LG |
| 25 | Принтер | Samsung |
| 30 | Принтер | HP |
| 35 | Процессор | Intel Celeron |
| 40 | Процессор | AMD |

Таблица 6.4 – Товары

| Код товара | Тип товара | Технические характеристики | Постав-щик | Цена |
|---|---|---|---|---|
| 100 | 10 | 17" 723N AKS <Silver> (LCD, 1280x1024) | 4 | 3880 |
| 102 | 20 | 17" L1734S-BN Flatron <Black> (LCD, 1280x1024) | 1 | 3807 |
| 104 | 30 | LaserJet P1005 <CB410A> A4 14с./мин 2Мб USB2.0 | 3 | 4199 |
| 108 | 40 | CPU AMD ATHLON-64 X2 4200+ (ADO4200) 1Мб/ 1000МГц Socket AM2 | 4 | 1739 |
| 110 | 35 | CPU Intel Celeron 430 1.8 ГГц/ 512K/ 800МГц 775-LGA | 5 | 1154 |
| 112 | 15 | 17" AL1716Fs (LCD, 1280x1024) | 5 | 3900 |
| 114 | 25 | ML-1640 (A4, 8Mb, лазерный, 16 с./мин, 1200dpi, USB 2.0) | 6 | 3570 |
| 116 | 35 | CPU Intel Celeron D 336 2.8 ГГц\ 256K\ 533МГц 775-LGA | 1 | 1910 |
| 118 | 10 | 19" 932B SBV <Black> (LCD, 1280x1024, +DVI) | 1 | 7870 |
| 120 | 15 | 19" MONITOR Acer V193bm <Black> (LCD, 1280x1024) | 1 | 4980 |
| 122 | 25 | SCX-4200 (18 с./мин, 8Mb, лазерн.принтер A4, копир, сканер, USB2.0) | 3 | 6147 |
| 124 | 30 | hp LaserJet P1505 <CB412A> A4 23стр/мин 2Мб USB2.0 | 4 | 6515 |
| 126 | 40 | CPU AMD Phenom X3 8450 BOX (HD8450) 1.5+2Мб/ 3600МГц Socket AM2+ | 2 | 3928 |
| 128 | 35 | CPU Intel Pentium 4 631 3.0 ГГц/ 2Мб/ 800МГц 775-LGA | 1 | 2144 |
| 130 | 35 | CPU Intel Core 2 Duo E6550 2.33 ГГц/ 4Мб/ 1333МГц 775-LGA | 2 | 5197 |
| 132 | 30 | LaserJet P2014 <CB450A> A4, 23 с./мин 32Mb USB2.0/LPT | 4 | 9625 |
| 136 | 15 | 20" V203Wb <Black> (LCD, Wide, 1680x1050) | 4 | 5380 |
| 138 | 20 | 19" L1942S-BF Flatron <Black> (LCD, 1280x1024) | 4 | 6460 |
| 140 | 10 | 19" 923NW NKS <Silver> (LCD, Wide, 1440x900) | 1 | 4720 |

Таблица 6.5 – Склад

| Код товара | Дата поступления | Количество |
|---|---|---|
| 102 | 12.01.09 | 5 |
| 130 | 17.01.09 | 10 |
| 108 | 17.01.09 | 12 |
| 108 | 18.01.09 | 8 |
| 104 | 20.01.09 | 5 |
| 110 | 25.01.09 | 3 |
| 124 | 26.01.09 | 4 |
| 112 | 27.01.09 | 20 |
| 140 | 12.02.09 | 15 |
| 128 | 15.02.09 | 10 |

## 6.4 Тесты и контрольные вопросы

6.4.1 Язык SQL – это:
а) структурированный язык отчетов;
б) система управления базой данных;
в) структурированный язык запросов;
г) язык иерархической базы данных;
д) структурированный язык форм.
6.4.2 Оператор **SELECT** относится к:
а) языку определения данных;
б) языку манипулирования данными;
в) языку описания данных;
г) языку ввода данных;
д) языку отчетов.
6.4.3 К достоинствам языка SQL не относится:
а) стандартность;
б) реляционная основа языка;
в) зависимость от СУБД;
г) возможность программного доступа;
д) возможность создания интерактивных запросов.
6.4.4 В угловые скобки, используемые при описании синтаксиса языка SQL, заключаются:
а) слова, соответствующие переменным, которые задаются пользователем;
б) обязательные элементы синтаксиса;
в) элементы, из которых выбирается один;
г) элементы, которые повторяются;
д) необязательные элементы.
6.4.5 В квадратные скобки, используемые при описании синтаксиса языка SQL, заключаются:
а) слова, соответствующие переменным, которые задаются пользователем;
б) обязательные элементы синтаксиса;
в) элементы, из которых выбирается один;
г) элементы, которые повторяются;
д) необязательные элементы.
6.4.6 В фигурные скобки, используемые при описании синтаксиса языка SQL, заключаются:
а) слова, соответствующие переменным, которые задаются пользователем;
б) обязательные элементы синтаксиса;
в) элементы, из которых выбирается один;
г) элементы, которые повторяются;
д) необязательные элементы.

6.4.7 Задание значения DESC в предложении **ORDER BY** оператора **SELECT** означает:

а) группировку по столбцам; б) сортировку по возрастанию;

в) выборку записей; г) сортировку по убыванию;

д) группировку по строкам.

6.4.8 Задание значения ASC в предложении **ORDER BY** оператора **SELECT** означает:

а) группировку по столбцам; б) сортировку по возрастанию;

в) выборку записей; г) сортировку по убыванию;

д) группировку по строкам.

6.4.9 Итоговая функция AVG при построении выражения в операторе **SELECT** возвращает:

а) значение, равное числу строк в результирующей таблице;

б) среднее арифметическое значение выражения или заданного поля;

в) максимальное значение выражения или заданного поля;

г) сумму значений выражения или заданного поля;

д) дисперсию значений выражения или заданного поля.

6.4.10 Итоговая функция COUNT при построении выражения в операторе **SELECT** возвращает:

а) значение, равное числу строк в результирующей таблице;

б) среднее арифметическое значение выражения или заданного поля;

в) максимальное значение выражения или заданного поля;

г) сумму значений выражения или заданного поля;

д) дисперсию значений выражения или заданного поля.

6.4.11 Что такое SQL?

6.4.12 В чем заключаются основные достоинства языка SQL?

6.4.13 Для чего предназначен SQL?

6.4.14 Что называется запросом SQL?

6.4.15 Приведите примеры запросов SQL, возможных в MS Access.

6.4.16 Как перейти в **Режим SQL**?

6.4.17 На какие два подязыка можно условно разделить операторы языка SQL?

6.4.18 Для чего предназначен оператор **SELECT**?

6.4.19 Что включает синтаксис оператора **SELECT**?

6.4.20 Какие итоговые функции применяются в запросах?

6.4.21 Что задает предложение **FROM**?

6.4.22 Какая операция в предложении **FROM** используется для задания типа объединения таблиц в логический набор записей?

6.4.23 Что задает предложение **WHERE**?

6.4.24 Какие логические операторы могут использоваться в предложении **WHERE**?

6.4.25 Что используется для изменения порядка вычисления логических выражений в предложении **WHERE**?

6.4.26 Что задает предложение **GROUP BY**?

6.4.27 Что задает предложение **ORDER BY**?

# 7 Разработка приложений в Microsoft Office Access 2007

Целью работы является создание пользовательского приложения по работе с базой данных в Microsoft Office Access 2007.

## 7.1 Общие положения

### 7.1.1 Access – средство быстрой разработки приложений

*Приложение* представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи [12]. В лабораторной работе рассматриваются вопросы разработки приложения, использующего базу данных в Microsoft Office Access 2007. *Приложение базы данных* - программа или комплекс программ, использующих базу данных и обеспечивающих автоматизацию обработки информации из некоторой предметной области. Приложение базы данных в Access содержит набор объектов, среди которых могут быть таблицы, запросы, формы, отчеты, макросы и модули кода, которые используются вместе и упрощают использование базы данных.

Access относится к средствам быстрой разработки приложений (RAD – Rapid Application Development). Можно выделить следующие отличительные черты таких средств разработки [29]:

- наличие объектно-ориентированного языка программирования, позволяющего эффективно использовать модульный принцип составления программ;
- использование визуальных средств разработки, представляющих возможность заменить написание программного кода рисованием пользовательского интерфейса и заданием необходимой функциональности диалоговыми средствами;
- поддержка стандартных протоколов обмена данными между приложениями, позволяющая разрабатывать многоуровневые приложения, не зависящие от источника данных.

Наряду с созданием обычных ACCDB-файлов приложений в Access 2007 (в более ранних версиях Access - MDB-файлов) имеется возможность создавать ACCDE-файлы приложений (в более ранних версиях Access - MDE-файлов), в которых хранятся базы данных, предназначенные «только для исполнения». Отличительной чертой этих файлов является то, что в соответствующем приложении нельзя использовать конструкторы для модификации приложения, а также все модули являются компилированными. Это позволяет защитить приложение от исправления и исследования, а также уменьшить размер файла базы данных.

Чтобы создать копию приложения с базой данных, предназначенную только для исполнения, нужно выполнить команду **Создать ACCDE**, находящуюся на **Ленте** на вкладке **Работа с базами данных** в группе «Работа с базами данных». Появится диалоговое окно «Сохранить как», в котором вводится имя создаваемого файла и папка его размещения, затем необходимо нажать

кнопку **Сохранить**. Access откомпилирует исходную базу и сохранит ее в новом файле с расширением ACCDE, после чего выполнит сжатие файла.

При открытии ACCDE-файла некоторой базы на **Ленте** на вкладке **Создание** станут недоступными (изменят свой цвет на серый): команды групп «Формы» и «Отчеты», команда **Конструктор запросов** в группе «Другие», а также нельзя открыть созданные формы и отчеты в режиме конструктора. Это означает, что нельзя изменить или создавать формы, отчеты и модули. Вместе с тем по-прежнему останутся доступными все средства просмотра, создания и модификации таблиц, запросов и макросов.

### 7.1.2 Использование диспетчера кнопочных форм

Меню в прикладной программе – это первое, что видит пользователь, решив запустить приложение.

Основное назначение меню заключается в том, чтобы дать возможность пользователю получить легкий доступ ко всем элементам прикладной программы. При разработке меню придерживаются следующих принципов:

- заголовок должен включать максимально ясную информацию о его назначении;
- структура меню должна соответствовать частоте выполнения действий, логической последовательности их выполнения или, в крайнем случае, хотя бы по алфавитному порядку;
- функционально связанные группы команд выделяют с помощью разделителей.

В Access создавать меню можно в виде кнопочной формы. Построение всех кнопочных форм, необходимых для управления сложным приложением, может оказаться довольно трудоёмким процессом. Специальная надстройка Access - диспетчер кнопочных форм - помогает выполнить эту работу. Эта надстройка применяет довольно сложную технику для управления всеми кнопочными формами с помощью одной формы и использует специальную таблицу - драйвер с именем «Элементы кнопочной формы», что позволяет определить любое число кнопочных форм и создать до восьми кнопок в каждой из них.

Чтобы запустить надстройку, необходимо выбрать на **Ленте** на вкладке **Работа с базами данных** в группе «Работа с базами данных» команду **Диспетчер кнопочных форм** (рисунок 7.1). Диспетчер кнопочных форм сначала проверит, есть ли в базе данных (БД) кнопочная форма и таблица **Элементы кнопочной формы**. В случае их отсутствия выводится окно сообщения «Не удается найти кнопочную форму в этой базе данных. Создать кнопочную форму?». Следует нажать кнопку **Да**. Построив скелет главной кнопочной формы и таблицу **Элементы кнопочной формы**, Диспетчер кнопочных форм выведет на экран своё основное окно «Диспетчер кнопочных форм». В этом окне можно создать дополнительные кнопочные формы, изменить созданные формы, удалить их, а также закрыть форму. Чтобы построить дополнительную кнопочную форму (в диспетчере она называется страницей), необходимо щёлкнуть на

кнопке **Создать**, ввести её имя в следующем окне диалога и щёлкнуть на кнопке **ОК** (рисунок 7.2).

После создания всех необходимых дополнительных кнопочных форм необходимо выбрать одну из них в основном окне диспетчера и щёлкнуть на кнопке **Изменить**. На экране появится окно, аналогичное тому, которое находится на рисунке 7.3. Здесь можно определить новый элемент кнопочной формы, отредактировать существующий или изменить порядок их расположения. Раскрывающийся список **Команда** позволяет назначить действие для создаваемого или изменяемого элемента: переход к другой кнопочной форме, открытие формы в режиме добавления или редактирования, открытие отчёта, изменение кнопочной формы, выход из приложения и другие. После выбора команды и при необходимости указания её аргумента диспетчер поместит в кнопочную форму кнопку, после щелчка на который будет выполняться заданная команда.

В главной кнопочной форме следует создать кнопки, открывающие другие формы, и включить кнопку для выхода из приложения. В каждой дополнительной кнопочной форме надо предусмотреть одну кнопку для возвращения на предыдущий уровень в иерархии кнопочных форм или для перехода в главную кнопочную форму.

После щелчка на кнопке **Закрыть** в основном окне диспетчера кнопочных форм Access создаст в текущей БД форму с именем **Кнопочная форма**. Её можно переименовать, изменив значение свойства «Подпись» на вкладке «Макет» окна свойств формы. Открыть «Окно свойств» формы можно, выбрав команду **Страница свойств** в группе «Сервис», доступную на **Ленте** в режиме конструктора формы (вкладка **Конструктор**).

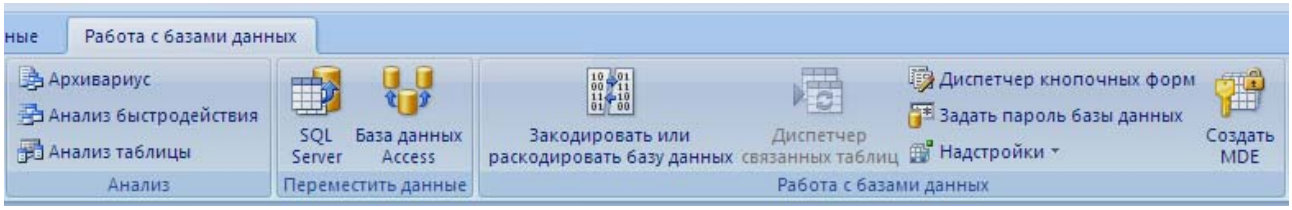



Рисунок 7.1 – Расположение **Диспетчера кнопочных форм** на **Ленте**

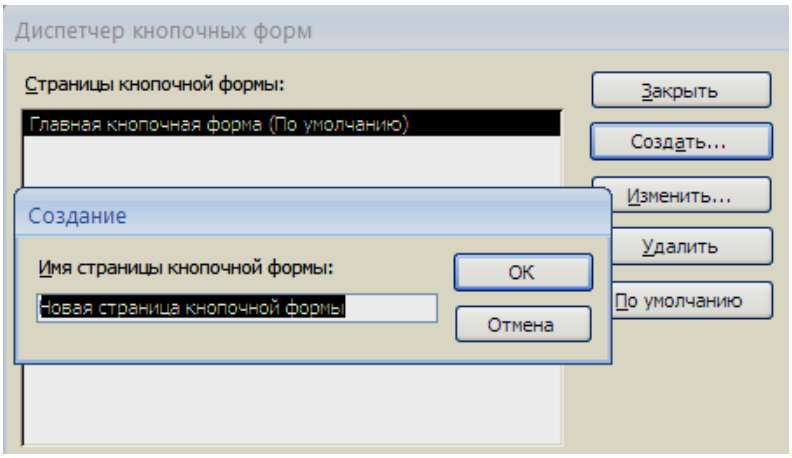



Рисунок 7.2 – Создание дополнительной кнопочной формы

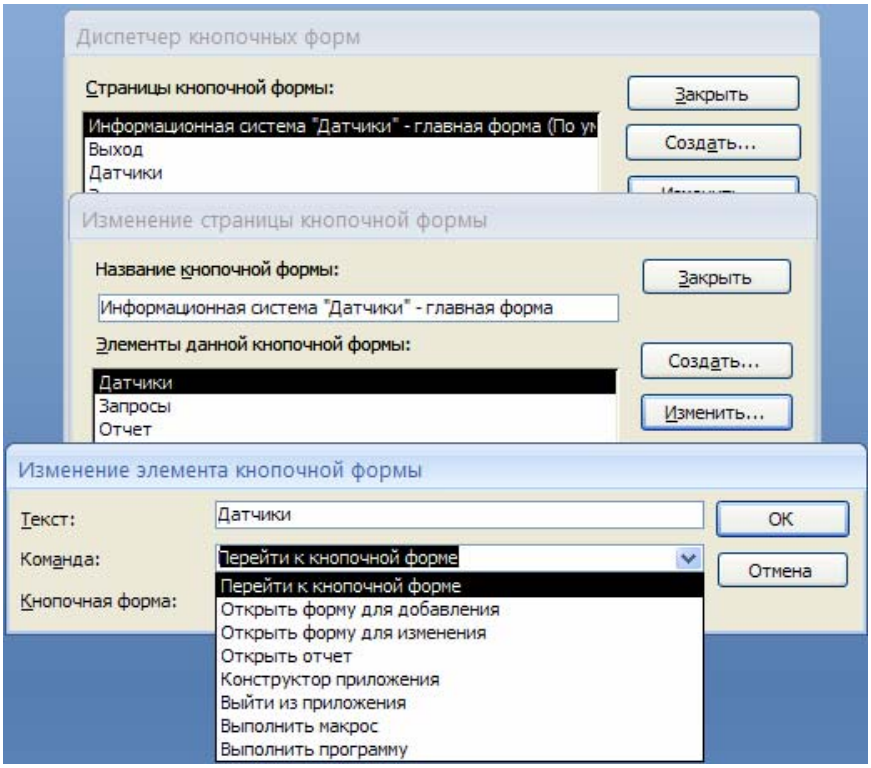



Рисунок 7.3 - Создание нового элемента кнопочной формы

### 7.1.3 Установка параметров запуска базы данных

После того, как построены все основные части приложения, можно задать режим автоматического запуска базы данных. Удобно для этих целей использовать параметры запуска и задать начальную форму приложения. Для этого необходимо нажать кнопку **Microsoft Office**  , а затем кнопку **Параметры Access**. В окне диалога «Параметры Access», щелкнув на кнопке **Текущая база данных**, возможно задание параметров текущей базы данных, как показано на рисунке 7.4.

В области «Параметры приложения» можно задать заголовок приложения и его значок. Можно указать только имя файла значка, если уверены, что при открытии базы данных он будет находиться в текущей папке. В противном случае надо задать полный путь к этому файлу. Поле «Форма просмотра» позволяет выбрать форму, которая будет выводиться на экран при открытии базы данных. Флажок **Строка состояния** снимается, если необходимо, чтобы при запуске приложения автоматически скрывалось окно строки состояния. Настройка удобного интерфейса пользователя может осуществляться также и другими параметрами в областях «Параметры приложения», «Переходы», «Параметры ленты и панелей инструментов», «Параметры автозамены имен» и «Параметры автофильтра».

Настройка внешнего вида таблиц в Access (цвет и шрифт по умолчанию, эффекты сетки и ячеек) осуществляется по нажатию кнопки **Таблица** в окне диалога «Параметры Access». Настройка параметров создания и изменения объектов баз данных – по нажатию кнопки **Конструкторы объектов**.

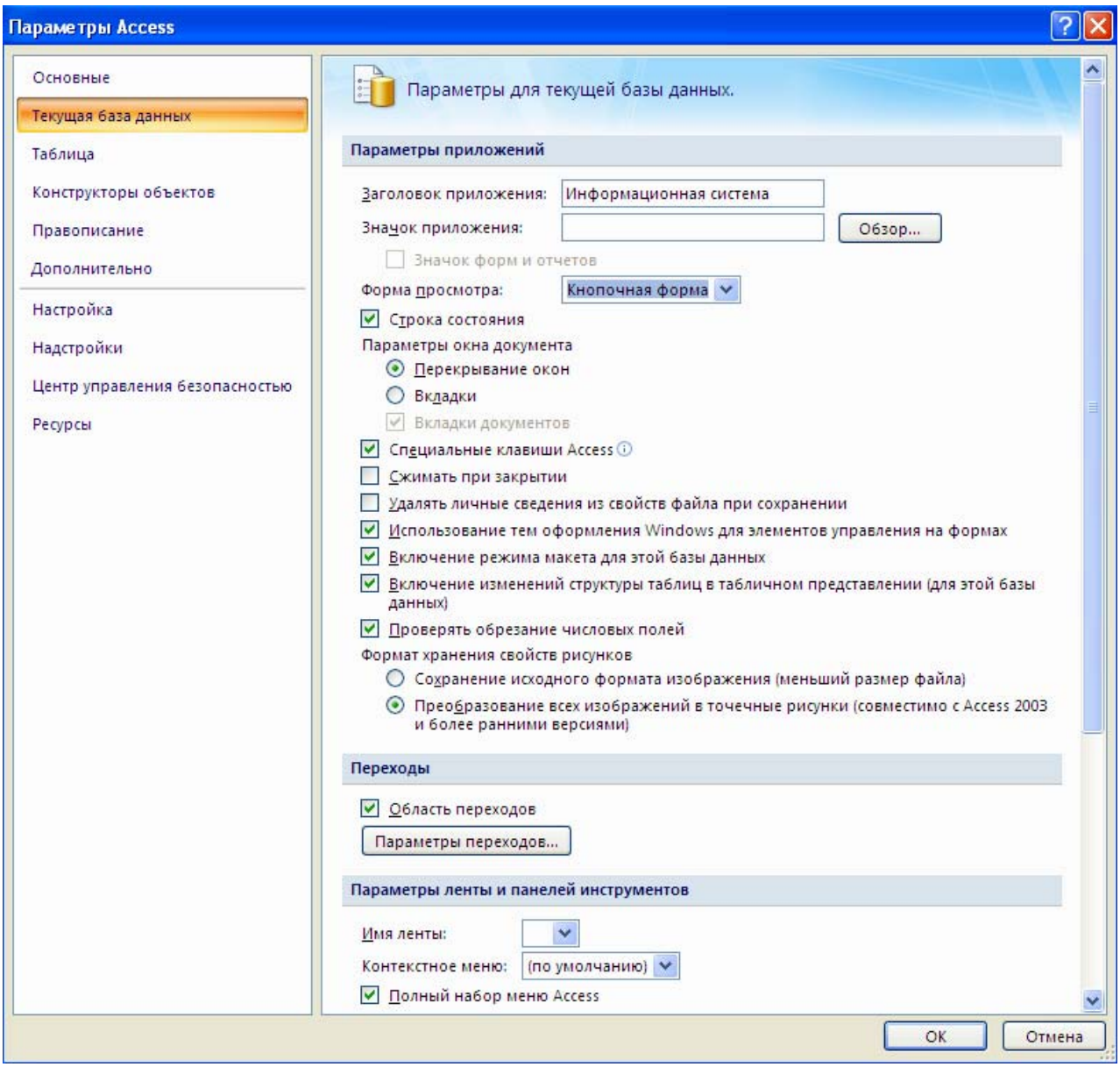



Рисунок 7.4 – Установка параметров базы данных

## 7.2 Задание на выполнение работы

7.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к данной лабораторной работе.

7.2.2 Запустить Access, создать новую базу данных «Датчики», сохранить ее в созданной папке.

7.2.3 Описать структуры таблиц «Датчики», «Типы датчиков», «Применение» в соответствии с рисунками 7.5 - 7.7.

7.2.4 Осуществить ввод данных в созданные таблицы. Значения данных приведены в таблице 7.1.

В лабораторной работе поле «Фото» в таблице «Датчики» является полем объекта OLE. Создать его содержимое удобно следующим образом: для описываемого объекта сделать поле текущим, затем из контекстного меню выпол-

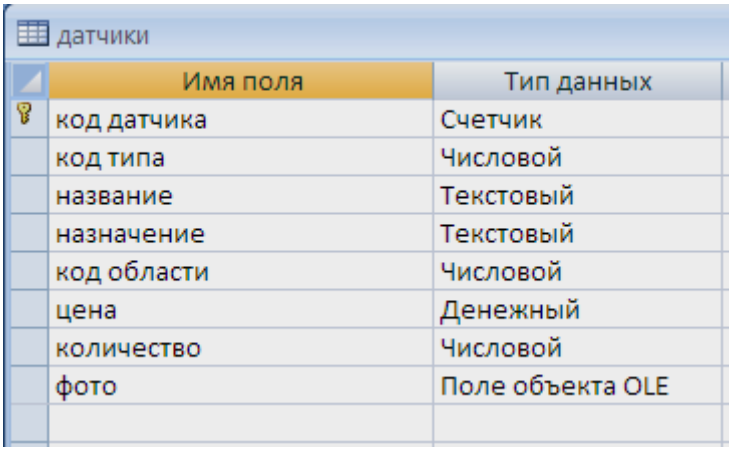



Рисунок 7.5 - Структура таблицы «Датчики»

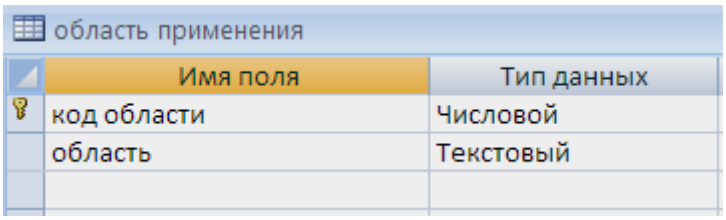



Рисунок 7.6 - Структура таблицы «Область применения»

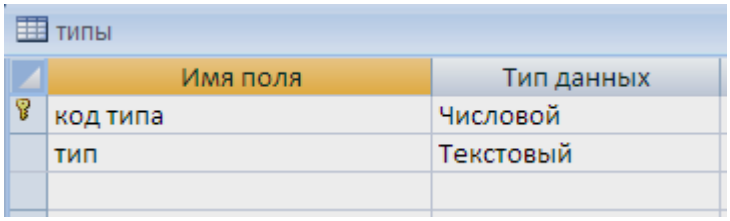



Рисунок 7.7 - Структура таблицы «Типы»

Таблица 7.1 – Датчики

Продолжение таблицы 7.1

| Название | Тип | Назначение | Область | Цена, р | Количество |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| Датчик бутылочный Люкс-110 | Радиочастотные | Включает автоматически источники света при вступлении в зону охвата | Торговля Охрана | 154 650 | 100 50 |
| Преобразователи давления АИР-20 | Давления | Преобразования значений давления в унифицированный токовый выходной сигнал | Промышленность Охрана | 2000 2400 | 35 10 |
| Беспроводной датчик движения Бирка | Движения | Включает автоматически источники света при вступлении в зону охвата | Торговля | 250 | 500 |
| Преобразователи давления Метран-20 | Давления | Преобразования значений давления в унифицированный токовый выходной сигнал | Промышленность | 1500 | 14 |
| Датчик измерительные FERRIT TAG АИР – 20/М2 | Радиочастотные | Противокражные | Торговля | 300 | 400 |
| Echotel 91s | Уровня | Ультразвуковые сигна- | Про- | 850 | 23 |

| | | | | | |
|---|---|---|---|---|---|
| | | лизаторы уровня в жидкостях | мышленность | | |
| MiniSightPlus | Температуры | Измерение температуры | Промышленность | 500 | 20 |
| Оптический датчик температуры | Температуры | Измерение температуры | Промышленность | 1350 | 12 |
| DewPro MMY 30 | Влажности | Измерение влажности | Промышленность | 1000 | 50 |

**объект…»**. Появляется диалоговое окно (рисунок 7.8). В нем предлагается выбрать тип объекта, а также способ его определения: создать объект с помощью соответствующей программы или вставить его готовым из файла. Независимо от способа определения объекта, существуют два варианта включения объекта в поле записи базы (задается с помощью флажка **Связь**): путем внедрения исходного объекта в базу данных или путем связывания (флажок **Связь** включают), когда устанавливается связь между отдельно хранящимся файлом объекта и записью базы данных.

7.2.5 Установить связи между таблицами в окне диалога «Схема данных» (рисунок 7.9). Для каждой связи в диалоговом окне «Изменение связей» установить:

- флажок напротив опции **обеспечение целостности данных**, который означает, что перед тем как занести данные в подчиненную таблицу, программа будет проверять их на соответствие главной;

- флажок напротив опции **каскадное обновление связанных полей**, который означает, что изменения в главной таблице автоматически будут влиять на подчиненную;

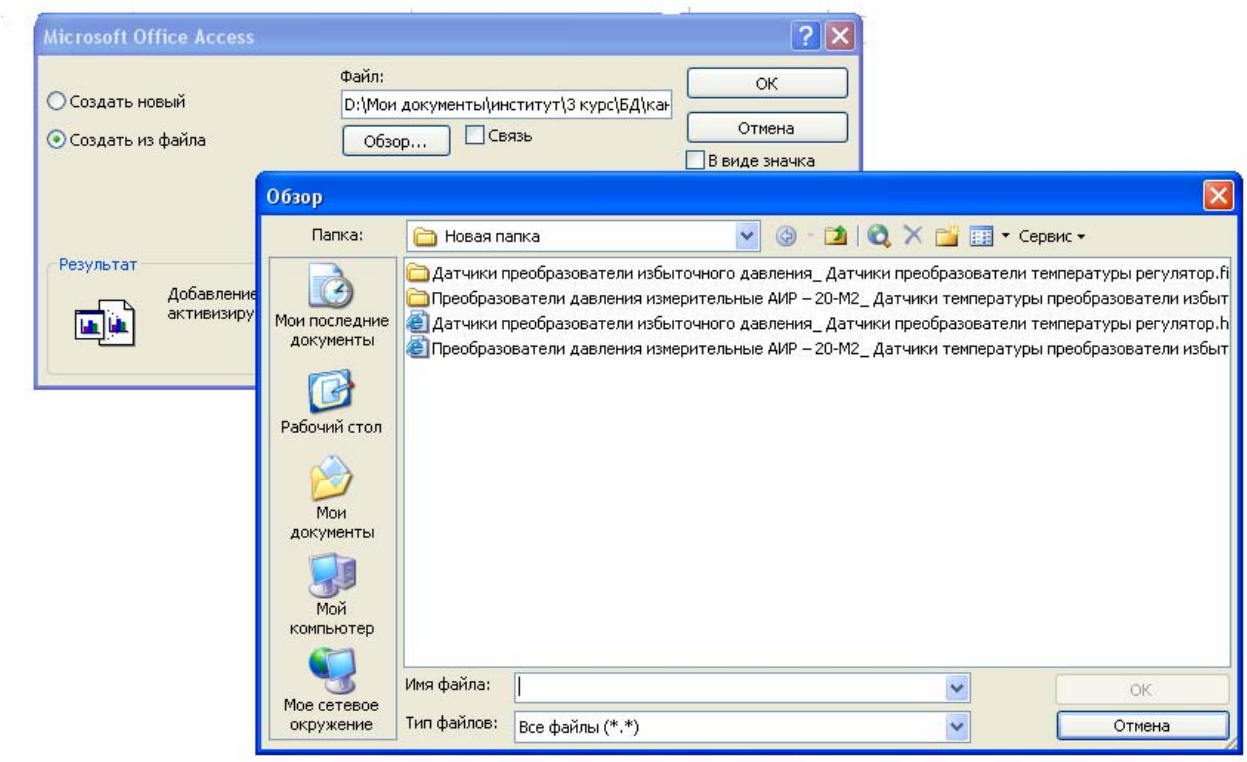



Рисунок 7.8 – Диалоговое окно вставки OLE-объектов

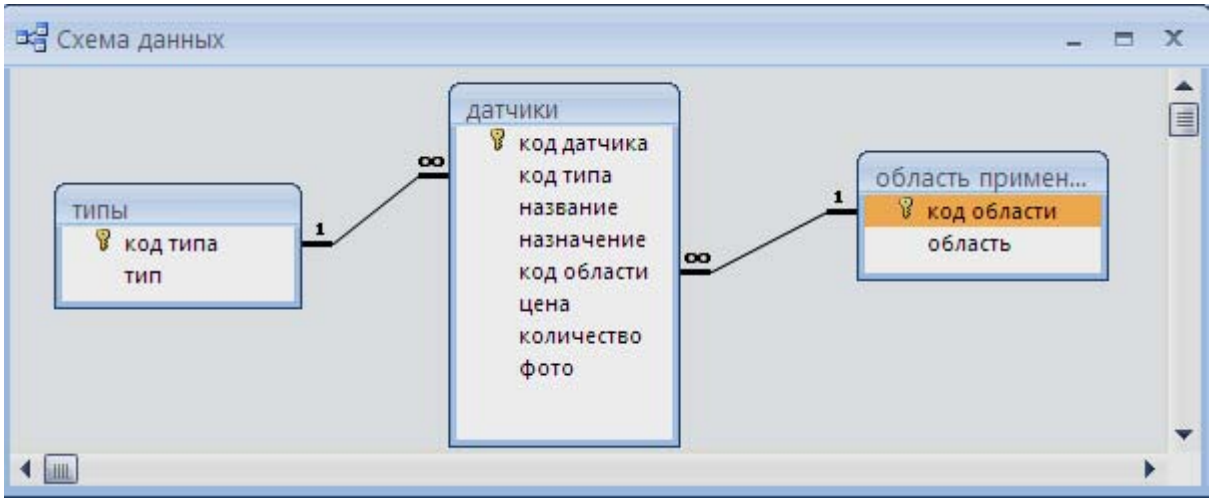



Рисунок 7.9 - Схема данных для таблиц «Типы», «Датчики» и «Область применения»

- флажок напротив опции **каскадное удаление связанных полей**, по которой поля, удаленные в главной таблице, будут удалены и в подчиненной.

7.2.6 Создать с помощью конструктора экранную форму «Датчики», в качестве источников для которой взять таблицы «Датчики», «Типы» и «Область применения». Пример экранной формы приведен на рисунке 7.10.

7.2.7 Создать с помощью **Конструктора** следующие запросы:

- по датчикам, применяемым в промышленности;
- по датчикам, применяемым в торговле;

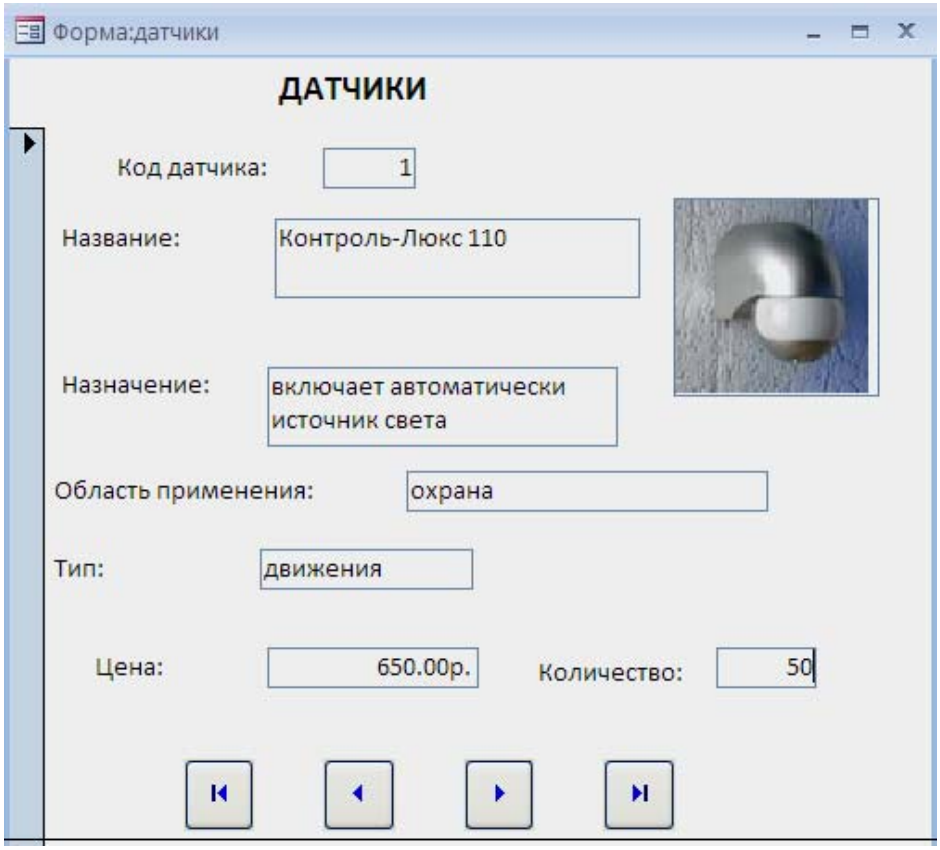



Рисунок 7.10 - Экранная форма «Датчики»

- по датчикам, применяемым в охране;
- по назначению (запрос с параметром).

Для запросов по различным областям применения используются все три таблицы, условие отбора записывается по полю «Область применения» из таблицы «Область применения». На экран выводятся значения всех полей, кроме поля «Область применения».

Access позволяет создавать запросы, в которых не определено, какие именно значения должны использоваться при выполнение запроса. Для этого включается в запрос параметр, и при каждом выполнение запроса Access будет запрашивать конкретное условие отбора. Чтобы определить параметр, вводится в строку **Условие отбора** вместо конкретного значения имя или фраза, заключенная в квадратные скобки (**[]**). То, что заключено внутри квадратных скобок, Access рассматривает как имя параметра. Оно выводится в окне диалога при выполнении запроса. При создании запроса по назначению в окне **Конструктора** необходимо сделать пометку вывода на экран из полей «Название», «Назначение», «Цена» из таблицы «Датчики». В строку **Условия отбора** для поля «Назначение» необходимо ввести **[Введите назначение]**.

7.2.8 По созданным запросам в пункте 7.2.7 создать экранные формы, включая все поля запросов.

7.2.9 Создать в **Мастере отчетов** отчет по датчикам на основании запроса «Датчики, применяемые в промыщленности». В отчет не включать поле «Фото», группировку осуществить по полю «Тип». Сортировку по возрастанию вы-

полнить по полю «Название». В режиме Конструктор, открыв созданный отчет, изменить название отчета, расположить отображаемые в отчете поля так, чтобы были видны все названия полей, и весь отчет помещался при просмотре на экране. Примерный вид созданного отчета показан на рисунке 7.11.

7.2.10 При помощи **Диспетчера кнопочных форм** создать главную и дополнительные кнопочные формы. На главной кнопочной форме «Информационная система» (рисунок 7.12) при нажатии на кнопку **Датчики** должна открываться в режиме редактирования форма «Датчики»; на кнопку **Запросы** открываться дополнительная кнопочная форма «Запросы», приводящая к открытию экранных форм, созданных в пункте 7.2.8; на кнопку **Отчеты** – дополнительная кнопочная форма «Отчеты», приводящая к открытию отчета по датчикам, применяемым в промышленности; на кнопку **Выход** – осуществляется выход из приложения.

7.2.11Отредактировать кнопочные формы в режиме **Конструктор**: внедрить рисунок, изменить свойства, раскрыв их командой **Страница свойств**, взятой на **Ленте** на вкладке **Конструктор** в группе «Сервис».

7.2.12 Задать параметры запуска приложения.

7.2.13 Создать ACCDE-файл, как описано в пункте 7.1.1.

7.2.14 Оформить отчет по лабораторной работе.

## 7.3 Содержание отчёта

7.3.1 Название работы.

7.3.2 Цель работы.

7.3.3 Структура таблиц, входящих в базу данных «Датчики».

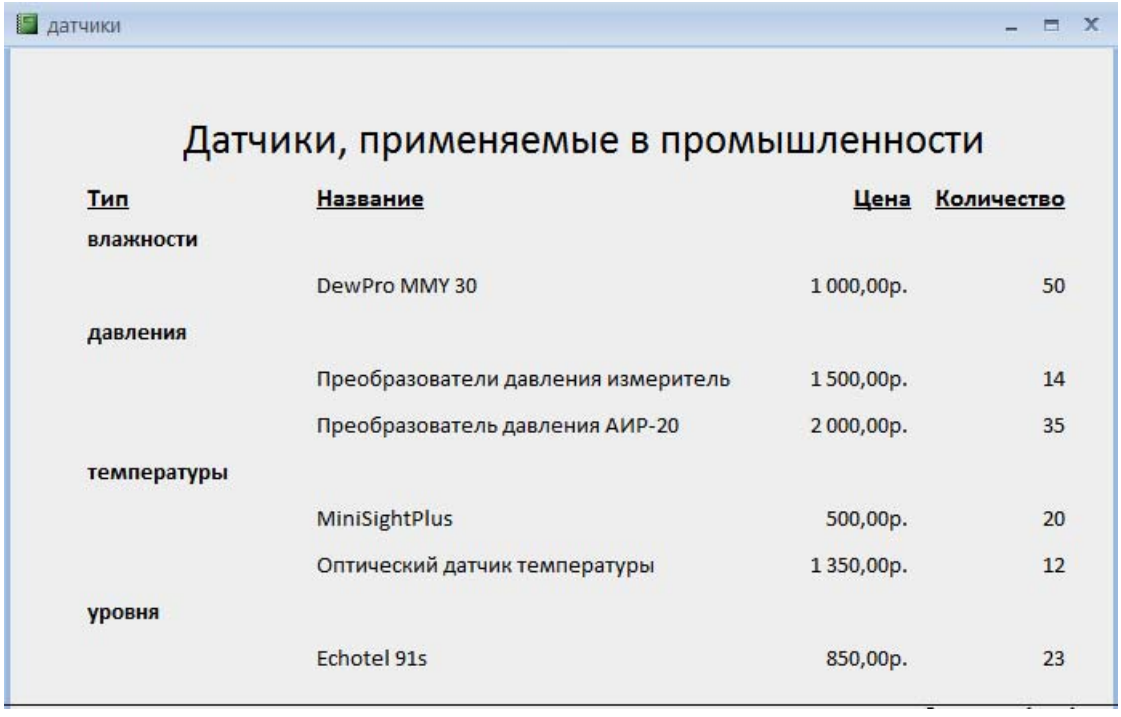



Рисунок 7.11 – Пример отчета

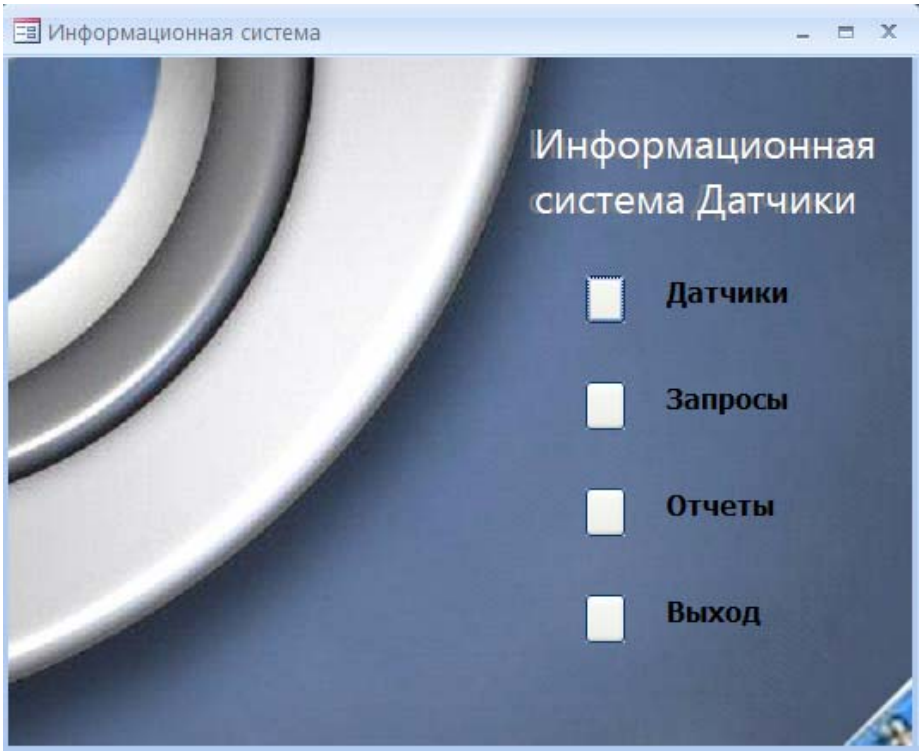



Рисунок 7.12 – Главная кнопочная форма

7.3.4 Перечень элементов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

7.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

7.3.6 Размеры созданных ACCDB- и ACCDE-файлов.

## 7.4 Тесты и контрольные вопросы

7.4.1 Приложение базы данных – это:

а) программа или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи;

б) специальным образом описанное требование, определяющее состав производимых над базой данных операций по выборке или модификации хранимых данных;

в) объект базы данных, основное назначение которого описание и вывод на печать документов на основе данных базы;

г) программа или комплекс программ, использующих базу данных и обеспечивающих автоматизацию обработки информации из некоторой предметной области;

д) объект базы данных, в котором разработчик размещает элементы управления, служащие для ввода, отображения и изменения данных в полях.

7.4.2 Что не относится к отличительным чертам средств быстрой разработки приложений:

а) наличие объектно-ориентированного языка программирования;

б) поддержка стандартных протоколов обмена данными между приложениями;

в) использование визуальных средств разработки;

г) обеспечение автоматизации обработки информации из некоторой предметной области?

7.4.3 На какой вкладке **Ленты** находится команда **Создать ACCDE**:

а) **Создание**;

б) **Работа с базами данных**;

в) **Главная**;

г) **Режим таблицы**;

д) **Конструктор**?

7.4.4 Особенностями ACCDE-файлов являются:

а) запрет на использование конструкторов форм и отчетов; больший, по сравнению с ACCDB-файлов, размер;

б) запрет на использование конструкторов форм и отчетов; меньший, по сравнению с ACCDB-файлов, размер;

в) возможность модификации форм и отчетов; больший, по сравнению с ACCDB-файлов, размер;

г) возможность модификации форм и отчетов; меньший, по сравнению с ACCDB-файлов, размер;

д) защита приложений от исправлений и исследований; больший, по сравнению с ACCDB-файлов, размер.

7.4.5 Основное назначение меню заключается в том, чтобы дать возможность пользователю получить легкий доступ:

а) ко всем формам прикладной программы;

б) ко всем формам и отчетам приложения;

в) ко всем элементам прикладной программы;

г) ко всем формам приложения.

7.4.6 На какой вкладке **Ленты** находится команда **Диспетчер кнопочных форм**:

а) **Главная**;

б) **Конструктор**;

в) **Создание**;

г) **Режим таблицы**;

д) **Работа с базами данных**?

7.4.7 Для чего используется драйвер с именем «Элементы кнопочной формы»:

а) для управления кнопочной формой открытия приложения;

б) для вызова кнопочной формы открытия приложения;

в) для управления всеми кнопочными формами;

г) для компилирования исходной базы данных;

д) для запуска приложения?

7.4.8 На какой вкладке **Ленты** находится команда **Страница свойств**:

а) **Конструктор**;

б) **Работа с базами данных**;

в) **Создание**;

г) **Режим таблицы**;

д) **Главная**?

7.4.9 Какого типа должно быть поле, значение которого представлено картинкой, созданной приложением Windows:

а) текстового;

б) поле объекта OLE;

в) гиперссылка;

г) поле Memo?

7.4.10 При задании запроса с параметром в строку **Условия отбора** выводимая на экран фраза должна быть заключена:

а) в квадратные скобки;

б) в фигурные скобки;

в) в угловые скобки;

г) в символы «вертикальная черта»;

д) в кавычки.

7.4.11 Что такое «приложение»?

7.4.12 Что такое «приложение базы данных»?

7.4.13 Что содержит приложение базы данных в Access?

7.4.14 Назовите отличительные черты средств быстрой разработки приложений?

7.4.15 Чем отличаются ACCDE- файлы, создаваемые в Access?

7.4.16 Как создать ACCDE-файл?

7.4.17 Для чего предназначено меню?

7.4.18 Каких принципов придерживаются при разработке меню?

7.4.19 В каком виде создается меню в лабораторной работе?

7.4.20 Для чего используется драйвер с именем «Элементы кнопочной формы»?

7.4.21 Как создать содержимое поля объекта OLE?

7.4.22 Для чего используется команда **Схема данных**?

7.4.23 Как создается схема данных?

7.4.24 Что можно делать над данными в запросах?

7.4.25 Как создается запрос с параметром?

7.4.26 Для каких целей используется команда **Диспетчер кнопочных форм**?

7.4.27 Что отражается в таблице «Элементы кнопочных форм»?

7.4.28 Что входит в параметры запуска базы данных?

7.4.29 Как изменить свойства формы?

7.4.30 Как изменить параметры приложения?

# 8 Формирование базы данных в системе Delphi

Целью работы является приобретение навыков создания базы данных в среде программирования Delphi и формирование запросов к ней на языке SQL.

## 8.1 Общие положения

### 8.1.1 Основные сведения о системе программирования Delphi

Система визуального программирования Delphi обладает большой популярностью среди широкого круга пользователей: от неспециалистов до системных программистов, занимающихся разработкой сложных приложений и информационных систем [26].

Система визуального программирования Delphi относится к RAD – системам (Rapid Application Development, быстрая разработка приложений), позволяющая быстро и эффективно разрабатывать приложения, включая и приложения для работы с базами данных. Кроме того, обладает практически всеми возможностями современных СУБД, таких как, например, Microsoft Access или Visual Fox Pro. Она имеет развитые возможности по созданию пользовательского интерфейса с помощью широкого набора инструментальных программных средств, позволяет визуально подготавливать запросы к базам данных, а также непосредственно писать запросы на языке SQL.

Система сочетает в себе особенности визуального и объектно-ориентированного программирования с дружественной программисту средой разработки, архитектура которой построена на компонентах. В Delphi отпадает необходимость программировать стандартные элементы управления Windows, такие как строки редактирования, кнопки и даже диалоговые окна. Все это уже имеется в виде готовых компонентов и шаблонов. С помощью Delphi можно настроить работу компонентов под нужды конкретного приложения для Windows. В системе имеются развитые средства отладки, облегчающие разработку приложений.

Среда разработки Delphi визуально реализуется несколькими одновременно раскрытыми на экране окнами, как показано на рисунке 8.1. Окна могут перемещаться по экрану, частично или полностью перекрывая друг друга. Каждое окно несет в себе определенную функциональность, то есть предназначено для решения определенных задач.

Основными являются четыре окна:

а) главное окно, осуществляющее основные функции управления проектом создаваемой программы (позиция 1);

б) окно Конструктора форм, обеспечивающее интерфейс пользователя для создаваемого приложения (позиция 2);

в) окно Инспектора объектов, предназначенное для изменения свойств объектов и управления событиями, на которые реагирует объект (позиция 3);

г) окно Редактора кода программы, содержащее исходный текст модуля разрабатываемого приложения, написанного на языке Object Pascal (позиция 4).

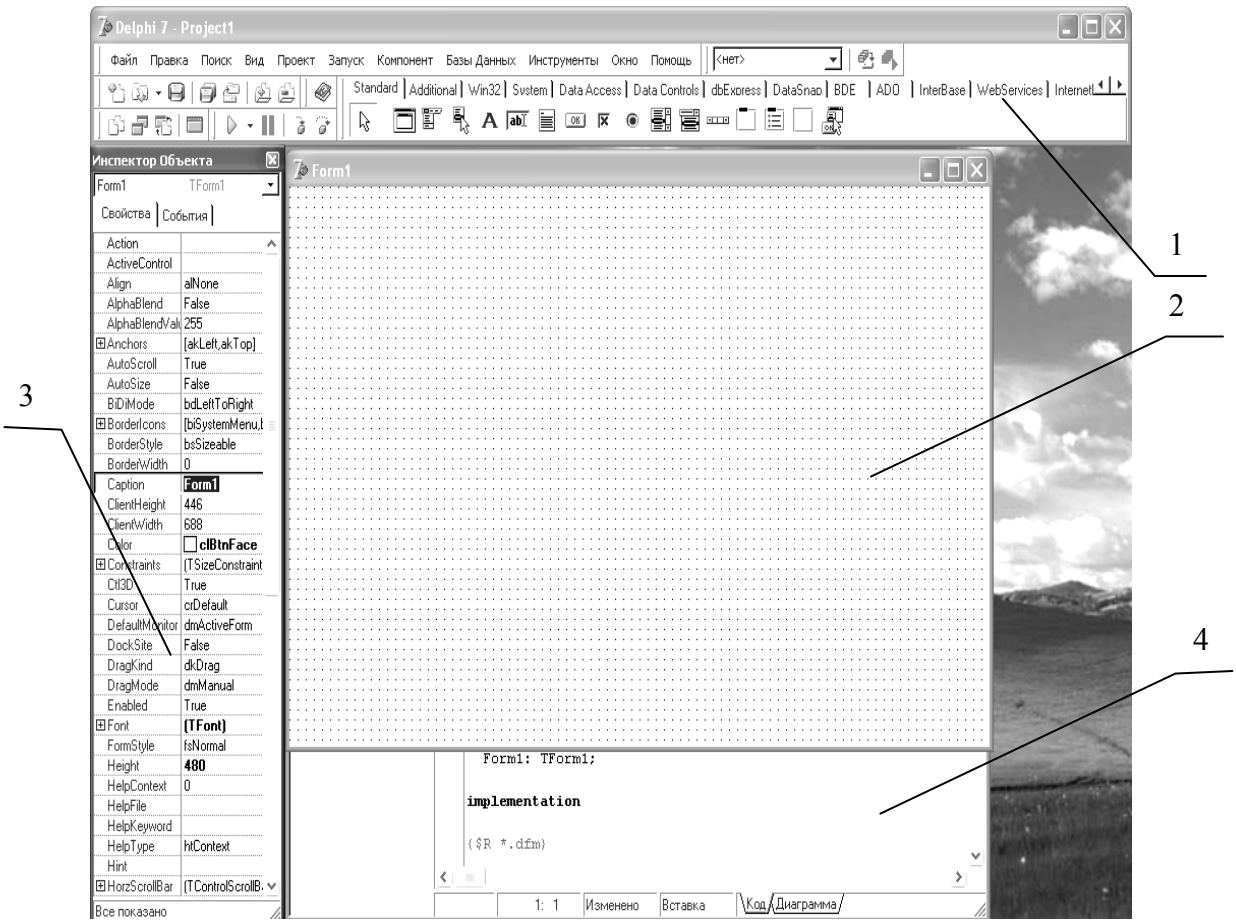



Рисунок 8.1 - IDE Borland Delphi 7 с новым проектом

На экране могут присутствовать и другие окна, отображаемые при вызове соответствующих средств. Окна Delphi (кроме главного окна) можно перемещать, убирать с экрана, а также изменять их размеры. Несмотря на наличие многих окон, Delphi является однодокументной средой, то есть позволяет одновременно работать только с одним приложением (проектом приложения). Название проекта приложения выводится в строке заголовка главного окна в верхней части экрана.

В лабораторной работе для создания файлов базы данных (БД) и вспомогательных файлов используется программа **Database Desktop**, которую необходимо загрузить из панели меню главного окна, выбрав опцию **Tools** (инструментарий). DataBase Desktop (DBD) работает как упрощенная версия Paradox и dBase for Windows. Программа предназначена для создания и редактирования таблиц, визуальных запросов и SQL-запросов, а также для выполнения действий с псевдонимами БД. Меню к панели DBD изменяются автоматически, в соответствии с текущим активным объектом.

Программа **Database Desktop,** окно которой показано на рисунке 8.2, имеет свое меню: **File**, **Edit**, **Tools**, **Window**, **Help**. При создании новой базы данных необходимо выбрать **File => New => Table**. В открывшемся окне **Create Table** возможно задание нужного типа формата таблиц **Table type**. Выбор нужного формата показан на рисунке 8.3. В Delphi поддерживается несколько ти-

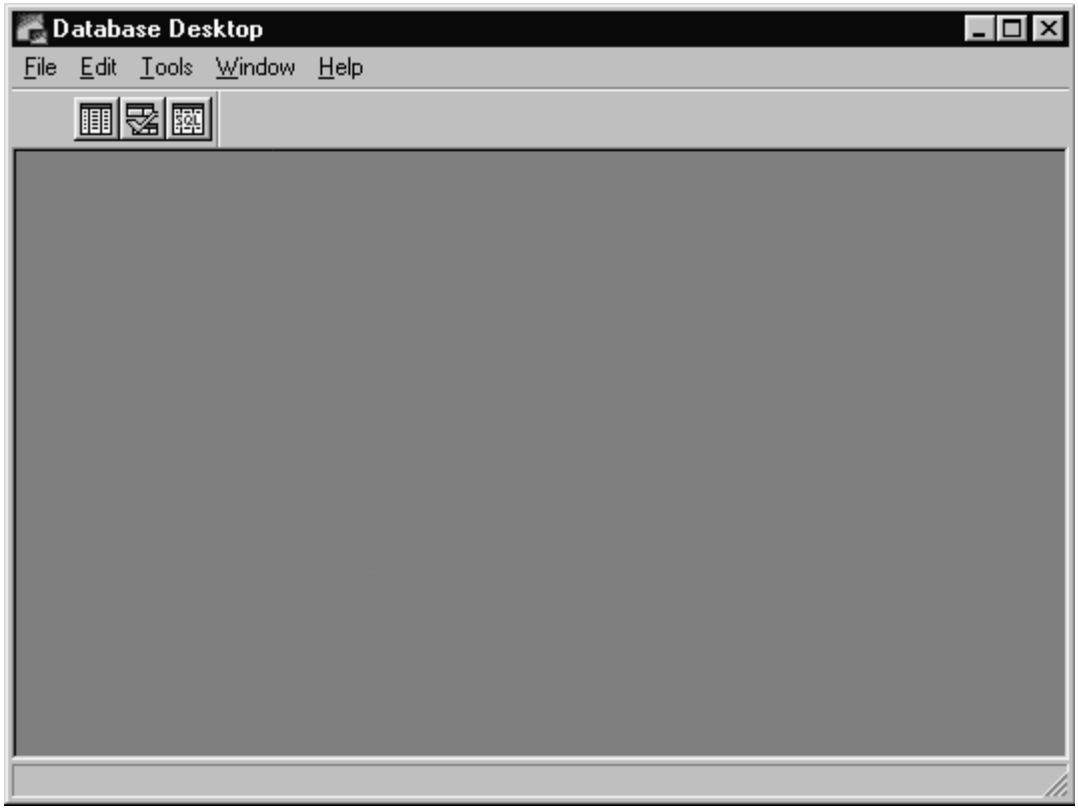



Рисунок 8.2 – Окно Database Desktop

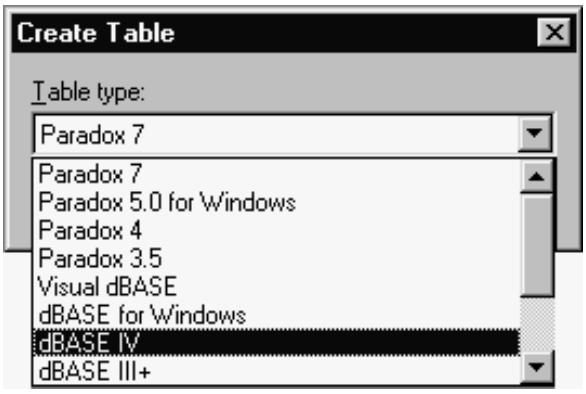



Рисунок 8.3 – Выбор формата таблицы

пов баз данных: Paradox 7, Paradox 5 for Windows, Paradox 4, Visual dBase, dBase IV, MS Access и другие.

При выполнении лабораторной работы рекомендуется выбрать тип dBase IV. Таблицы dBase являются одним из первых появившихся форматов таблиц для персональных компьютеров и поддерживаются многими системами, которые связаны с разработкой и обслуживанием приложений, работающих с базой данных [26]. Основные достоинства таблиц dBase:

- простота использования;
- совместимость с большим числом приложений;
- использование для хранения на дисках относительно мало физических файлов.

К недостаткам таблиц dBase относится то, что они не поддерживают автоматическое использование парольной защиты и контроль целостности связей, поэтому программист должен кодировать эти действия самостоятельно.

### 8.1.2 Описание структуры базы данных dBase IV

Описание структуры базы данных заключается в задании для каждого поля его характеристик. В формате данных dBase IV задаются следующие характеристики: имя **Field Name**, тип **Type**, длина **Size** и точность **Dec**. Каждое поле должно быть занесено в раздел **Field Roster** (Список полей) диалогового окна. Этот раздел появляется при работе команды **Create Table**, как показано на рисунке 8.4.

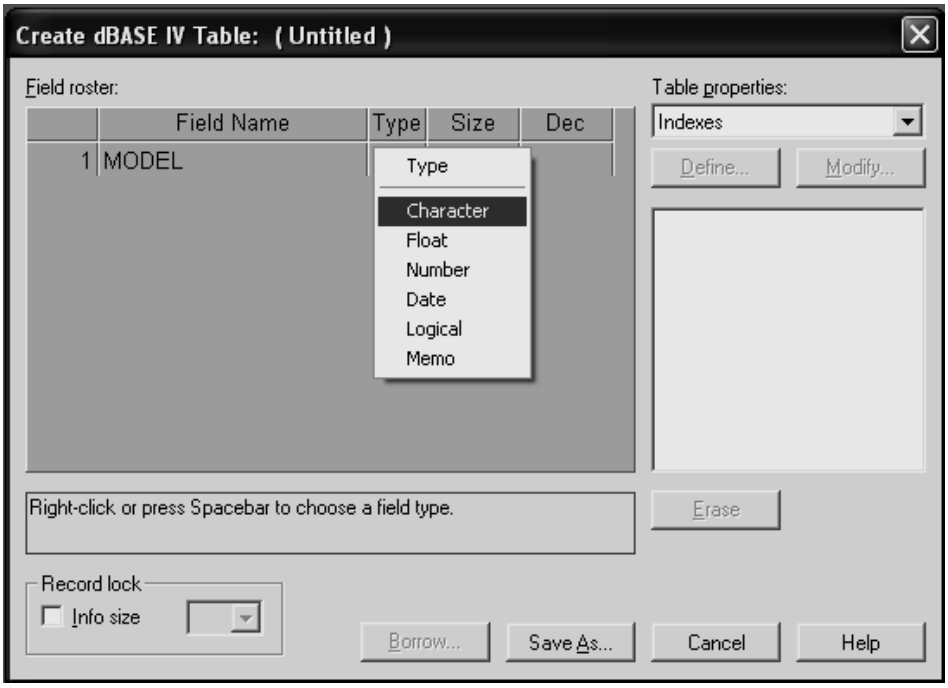



Рисунок 8.4 – Окно для описания структуры таблицы

Имя поля может быть длиной до 10 символов, должно начинаться с буквы и не может содержать пробелов. Допускается в имени поля использовать буквы, цифры и символ подчеркивания. Каждое имя поля должно быть уникальным. Ввод имени поля завершается нажатием клавиши **Enter**.

Типы полей зависят от выбранного формата таблицы. Например, в dBase используются следующие пять типов полей:

- символьный **Character** (предназначен для символьной информации );
- числовой с плавающей точкой **Float** (диапазон $-10^{308}$ … $10^{308}$);
- числовой с фиксированной точкой **Number** (содержит целые или дробные числа со знаком);
- дата **Date** (содержит данные в формате ММ/ДД/ГГ или ДД/ММ/ГГ);
- логический **Logical** (содержит значения «T» - истина или «F» - ложь);
- примечания **Memo** (поля для ввода текста переменной длины).

Тип поля можно задать, вводя первую букву ((C) - символьное, (N) - числовое, (D) - дата, (L) - логическое, (M) – поле примечаний) или из контекстного меню.

Длина поля **Size** должна быть задана для символьных и числовых полей и представляет собой максимальное число цифр и символов, которые предполагается занести в поле. Длина символьных полей - до 254 символов, числовых - до 19 разрядов, включая знак и десятичную точку. Поля типа даты, логические поля и поля примечаний имеют в системе заданную длину. Поля типа даты всегда имеют длину восемь символов, а логические поля - один символ. Полям примечаний автоматически присваивается длина 10 символов в файле базы данных, хотя содержимое этих полей может достигать 5000 знаков или больше.

Колонка «Точность» **Dec** заполняется только для числовых полей.

В записи допускается только до 128 полей. Максимальный размер записи - 4 Кб. Каждый символ занимает 1 байт.

Если допущены ошибки при определении структуры БД, то необходимо переместиться на соответствующую характеристику поля и произвести необходимые исправления.

Для записи созданной структуры в файл следует нажать кнопку **SaveAs**. После ее нажатия программа предлагает в диалоговом окне **Save Table As** задать таблице имя. В лабораторной работе рекомендуется в окне списка **Alias (**псевдоним) использовать имя **DBDEMOS**. Тип файла Delphi задается автоматически, в соответствии с выбранным форматом базы данных. Необходимо ввести имя создаваемой базы, например, **STANKI.DBF** и путь к нему в текстовом окне **File name** (Имя файла) и для сохранения таблицы щелкнуть мышью на **Save** (Сохранить).

### 8.1.3 Добавление данных

Формирование записей в соответствии с заданной структурой и добавление записей в файл базы данных может происходить двумя способами:

- из программы DataBase Desktop;
- из пользовательской программы, написанной на Delphi.

В лабораторной работе используется только первый способ. Прежде чем добавить запись, нужно открыть таблицу, используя команду **File => Open => Table** из главного меню. В появившемся диалоговом окне нужно задать псевдоним и имя файла. Таблица откроется для просмотра (рисунок 8.5). Для внесения изменений надо нажать кнопку  (Edit Data) или функциональную клавишу **F9**.

### 8.1.4 Модификация структуры файла базы данных

Модификация (изменение) структуры существующего файла базы данных состоит в добавлении/удалении полей или изменения характеристик поля (имени, длины, типа данных). При изменении структуры имя файла задавать не

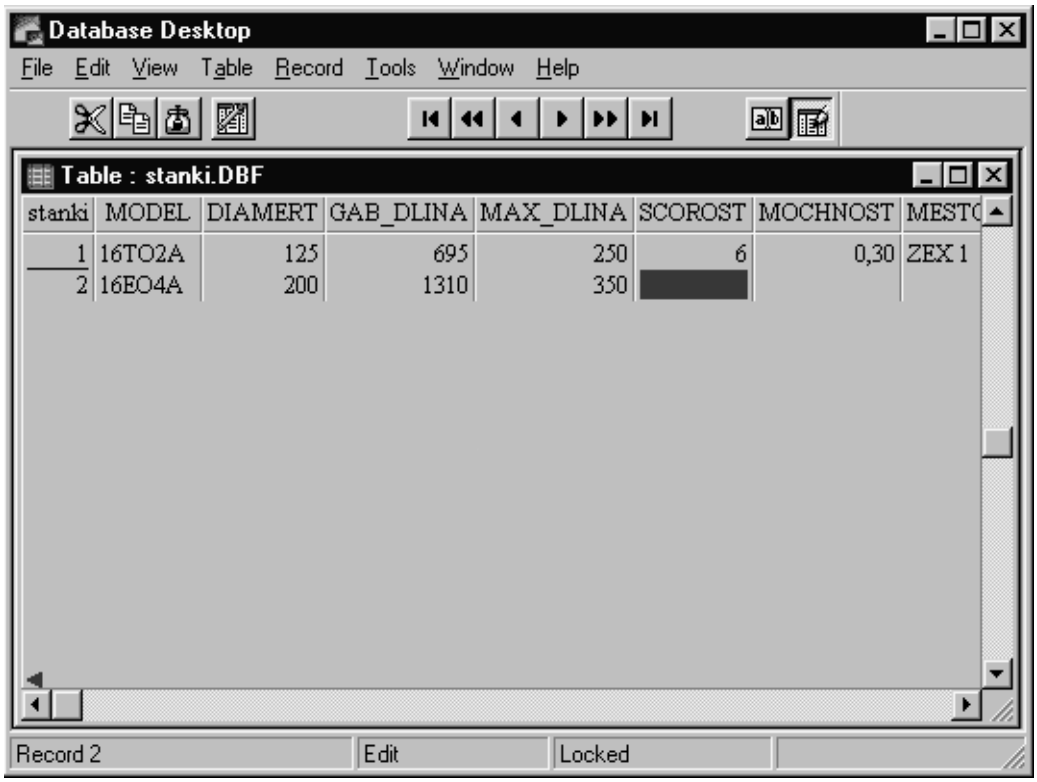



Рисунок 8.5 – Вид экрана при вводе данных

следует, поскольку в этом режиме можно работать только с активным файлом базы данных.

Войти в режим изменения структуры можно путем выбора и выполнения следующих опций в главном меню: **File => Open => Table**, затем нажать  (**Restructure**). После этого на экране появится описание структуры записи активного файла БД. Диалоговое окно Restructure аналогично диалоговым окнам создания соответствующих таблиц.

В режиме изменения структуры файла БД можно включать в состав базы новые поля. Для этого нужно высветить поле, перед которым будет вставлено новое поле, и нажать кнопку **Insert**. На экране в нужном месте появится пустая строка, в которую необходимо ввести характеристики нового поля. Для удаления поля необходимо высветить удаляемое поле и нажать **Ctrl+Delete**.

Когда обновление завершено, новую структуру нужно сохранить с помощью кнопки **Save**.

### 8.1.5 Изменение свойств таблиц

**DataBase Desktop** позволяет изменить некоторые свойства, влияющие на внешний вид таблиц. Например, можно установить размер строк и столбцов. Эти настройки не влияют на сохранность данных на диске и на программы Delphi, а используются только для удобства просмотра данных.

Для регулирования размера столбцов необходимо поместить указатель мыши на разделитель первого видимого столбца в таблице. Когда это произой-

дет, курсор превратится в направленную горизонтальную стрелку. Необходимо нажать кнопку мыши и переместить разделитель на новое место. Если сильно уменьшить столбец с текстовыми данными, то текст обрезается. Если уменьшить колонку с числовыми данными, то непомещающиеся в столбце числа отображаются набором звездочек. Для изменения размера строк таблицы необходимо поместить указатель мыши на разделитель первой видимой записи в первом видимом столбце в таблице. Обычно поле в этой позиции подчеркнуто. После этого курсор мыши превратится в двунаправленную вертикальную стрелку. Необходимо нажать кнопку мыши и переместить разделитель на новое место. Во время перемещения на месте разделителя появится серая горизонтальная линия. Не рекомендуется делать строки слишком узкими, чтобы не сделать таблицу нечитаемой.

Можно перемещать столбцы в таблице относительно друг друга. Для этого необходимо поместить указатель мыши на заголовок столбца, который надо переместить. Затем нажать и удерживать кнопку мыши. В результате разделители у краев столбца будут выделены, что сигнализирует о готовности к перемещению. После чего столбец переносится на новое место. При переносе столбец представляется в виде серого прямоугольника. Если столбец переносится за пределы видимой части таблицы, то таблица прокручивается вслед за перемещаемым столбцом. После переноса столбца на новое место нужно отпустить кнопку мыши.

Столбцы, не интересующие в данный момент, можно перенести в конец таблицы. Для этого необходимо активизировать нужный столбец, выделив в нем любую ячейку, и нажать **Ctrl+R**. Выбранный столбец окажется крайним справа.

### 8.1.6 Использование запросов SQL

После того как создана структура файла базы данных и введена информация в нее, обычно требуется отбирать и выводить на экран или печать те данные, которые отвечают поставленной задаче. Для этого формируются запросы на отбор необходимых данных.

Диалоговые запросы к базе данных лучше всего реализуются с помощью встроенного языка SQL (более подробно язык SQL рассматривался в лабораторной работе 6). Язык запросов SQL (Structured Query Language – Структурированный язык запросов) - близок к классу языков реляционного исчисления кортежей и применяется в основном в реляционных СУБД. Хотя название SQL предполагает, что это язык запросов, он включает в себя помимо средств запросов определение таблиц, обновление базы данных, определение представлений данных и привилегий доступа. Язык SQL включает в себя небольшое число операторов. В лабораторной работе рассматриваются операторы **CREATE** и **SELECT**.

Для создания запроса необходимо загрузить программу Database Desktop и выбрать меню **File => New => SQL file**. Появится окно для ввода SQL запросов. Далее нужно выбрать Alias: **SQL => Select Alias**... В появившемся диалого-

вом окне выбрать нужный Alias (например, DBDEMOS) и нажать кнопку **OK**.

Выполнение запроса осуществляется командой **SQL => RunSQL** или при нажатии клавиши **F8**.

Основным оператором языка SQL, выполняющим отбор информации из базы данных, является оператор **SELECT**, имеющий формат, в который входит несколько компонент. В лабораторной работе использовать следующий формат оператора:

**SELECT** < *предложение* >
**FROM** < *имя таблицы* >
[**WHERE** < *предложение* >]
[**ORDER BY** < *предложение* >]

Обязательными являются предложения **SELECT** и **FROM**. Все компоненты команды **SELECT**, если они используются, должны быть записаны в том порядке, в котором они перечислены в формате команды.

Предложение **SELECT** перечисляет столбцы, которые должны войти в результирующую таблицу. Это всегда столбцы некоторой реляционной таблицы. Для предложения SELECT может использоваться следующий формат:

**SELECT** [DISTINCT|ALL][*|< *список выражений* >]

Если указано DISTINCT (отличный от других), то в ответ выдаются только уникальные строки, если ALL - все строки, в том числе - повторяющиеся. Так как по умолчанию принимается значение ALL, то оно в явном виде может не указываться.

Знак * указывается, если в ответ выводятся все поля (столбцы) таблицы в том порядке, в котором они были заданы при описании таблицы. Использование знака * удобно при вводе интерактивных запросов, так как это сокращает длину запроса. В списке выражений обычно перечисляются идентификаторы столбцов, которые будут выведены в ответ.

В предложении **FROM** может указываться имя одной таблицы или список имен, если совместно обрабатывается несколько таблиц.

Например, синтаксис самого простого SQL запроса при отборе всей полей из таблицы STANKI выглядит следующим образом

**SELECT** *
**FROM** STANKI

При сохранении запроса для дальнейшего использования необходимо открыть меню **File** и выбрать команду **Save**. Затем вводится имя нового файла. Database Desktop сохранит запрос в файле с расширением .sql. Это обычный текстовый файл, который можно изменять в любом текстовом редакторе. Для открытия существующего файла с запросом используется команда **File => Open => SQL** и затем вводится имя запроса.

В результате выполнения описанного выше запроса на экране появится база данных, которая сохранялась в файле STANKI.DBF, со всеми полями.

Например, пусть имеется запрос

**SELECT** MODEL**,** MESTO
**FROM** STANKI

На экране будут видны только два поля из базы данных STANKI: модель

станка (MODEL) и место установки (MESTO).

Для поиска информации с заданными условиями, которым должны удовлетворять поля записей, используется предложение **WHERE** <*условие*>.

Например, необходимо составить список всех записей поля DIAM (Диаметр), у которых значение больше 13. Для оформления этого запроса в окне SQL Editor надо набрать следующее

**SELECT ***
**FROM** STANKI
**WHERE** DIAM > 13

Если требуется установить несколько условий, которым должны удовлетворять поля записей, то их необходимо связать одной из логических связок «AND» или «OR» («И» или «ИЛИ»).

Например, если необходимо получить информацию о станках, имеющих диаметр DIAM > 13 и установленных в цеху 1, то запрос выглядит следующим образом:

**SELECT ***
**FROM** STANKI
**WHERE** DIAM > 13 and MESTO =’ЦЕХ 1’

Информация, получаемая в результате реализации запроса, может быть упорядочена. Для этого используется фраза:

**ORDER BY** < *имя столбца* | *целое* > [ ASC|DESC ]

Вместо имени столбца в **ORDER BY** может записываться целое число, которое соответствует номеру столбца в таблице, считая слева направо. Параметр [ASC | DESC] означает вид сортировки (по возрастанию | убыванию соответственно). По умолчанию принимается значение ASC.

Создать базу данных можно при помощи команды **CREATE TABLE**. Её формат:

**CREATE TABLE** < *имя таблицы* > ( < *имя столбца* >
< *тип данных* > [< *имя столбца* >< *тип данных* >, ... ] )

Например, пусть необходимо создать таблицу под именем STUDENT, в которой имеется три поля. Первое – «FIO», содержащее фамилию, имя, отчество студента, имеет символьный тип. Второе – «DATA» (дата рождения), используемый тип - DATE. Третье поле «KURS» обозначает курс студента, оно может принимать целые положительные числа. В SQL запросе используется для создаваемой таблицы тип Paradox 7.

Запрос в этом случае после выбора команды **File => New => SQL file** в окне SQL Editor записывается в виде:

**CREATE** TABLE STUDENT
(FIO char, DATA date, KURS int )

Отправить указанный запрос можно, выполнив команду: **SQL => Run SQL**. Если затем сделать новый запрос

**SELECT ***
**FROM** STUDENT

то появится окно для ввода данных в базу STUDENT.

## 8.2 Задание на выполнение работы

8.2.1 Создать базу данных STANKI, используя формат dBASE IV. Параметры для разных типов станков приведены в таблицах 8.1 – 8.4. Номер таблицы при выполнении лабораторной работы задается преподавателем.

8.2.2 По заданию преподавателя изменить структуру данных в базе; вставить новое поле.

8.2.3 Выполнить SQL-запросы, задавая различные условия поиска.

8.2.4 Создать новую базу данных, содержащую четыре поля, используя SQL-запрос.

8.2.5 Оформить отчет по лабораторной работе.

## 8.3 Содержание отчета

8.3.1 Название лабораторной работы.

8.3.2 Цель работы.

8.3.3 Перечень используемых при выполнении лабораторной работы команд с указанием их назначения.

8.3.4 Структуры данных создаваемых таблиц.

Таблица 8.1 –Токарно-винторезные и токарные станки

| Модель станка | Наибольший диаметр обрабатываемой заготовки, мм | Габаритная длина (без ЧПУ), мм | Число скоростей шпинделя | Мощность электродвигателя, кВт | Место-установки станка | Дата ввода в эксплуатацию | Необходимость техобслуживания |
|---|---|---|---|---|---|---|---|
| 16ТО2А | 125 | 695 | 6 | 0,27 | цех 1 | 15.06.87 | ДА |
| 16БО4А | 200 | 1310 | 5 | 1,1 | цех 2 | 15.06.90 | НЕТ |
| 16БО5П | 250 | 1510 | 8 | 1,5 | цех 2 | 05.09.91 | НЕТ |
| 16Б16А | 320 | 2280 | 21 | 3,5 | цех 2 | 25.07.88 | ДА |
| 16В16Т1 | 320 | 3100 | 18 | 6,2 | цех 1 | 23.11.89 | ДА |
| 16Л2ОП | 400 | 2920 | 21 | 4,7 | цех 1 | 26.06.92 | НЕТ |

Таблица 8.2 – Токарно-карусельные станки

| Модель станка | Наибольший диаметр обрабатываемой заготовки, мм | Габаритная длина (без ЧПУ), мм | Чистота вращения планшайбы | Мощность электродвигателя, кВт | Место установки станка | Дата ввода в эксплуатацию | Необходимость техобслуживания |
|---|---|---|---|---|---|---|---|
| 1512 | 1250 | 5920 | 250 | 30 | цех 2 | 15.06.83 | ДА |
| 1А512МФ3 | 1450 | 6560 | 335 | 55 | цех 1 | 01.06.92 | НЕТ |
| 1516 | 1600 | 8615 | 280 | 30 | цех 2 | 05.09.93 | НЕТ |
| 1516Ф1 | 1600 | 5485 | 80 | 30 | цех 1 | 25.07.81 | ДА |
| 1А516МФ3 | 1800 | 8213 | 240 | 75 | цех 2 | 23.11.73 | НЕТ |
| 1525 | 2500 | 8615 | 200 | 40 | цех 1 | 26.06.73 | ДА |

Таблица 8.3 – Радиально-сверлильные станки

| Модель станка | Наибольший условный диаметр сверления в стали, мм | Габаритная длина (без ЧПУ), мм | Число подач шпинделя | Мощность электродвигателя, кВт | Место установки станка | Дата ввода в эксплуатацию | Необходимость техобслуживания |
|---|---|---|---|---|---|---|---|
| 2М55 | 50 | 2665 | 12 | 5,5 | цех 2 | 15.12.83 | ДА |
| 2554 | 50 | 2685 | 10 | 5,5 | цех 2 | 01.05.92 | НЕТ |
| 2Ш55 | 50 | 4280 | 8 | 4 | цех 2 | 05.01.93 | НЕТ |
| 2Р53 | 50 | 6685 | 12 | 5,5 | цех 1 | 25.03.81 | ДА |
| 2М57 | 75 | 3500 | 18 | 7,5 | цех 1 | 23.02.73 | НЕТ |
| 2М58-1 | 100 | 4850 | 18 | 13 | цех 1 | 26.06.73 | НЕТ |

Таблица 8.4 – Вертикально-сверлильные станки

| Модель станка | Наибольший условный диаметр сверления, мм | Наибольший ход шпинделя, мм | Число скоростей шпинделя | Мощность электродвигателя, кВт | Место установки станка | Дата ввода в эксплуатацию | Необходимость техобслуживания |
|---|---|---|---|---|---|---|---|
| 2Н106П | 6 | 125 | 7 | 0,4 | цех 1 | 15.02.75 | ДА |
| 2М112 | 12 | 190 | 5 | 0,6 | цех 2 | 27.02.75 | ДА |
| 2Н118 | 18 | 200 | 9 | 1,5 | цех 2 | 03.09.89 | НЕТ |
| 2Н125Л | 25 | 250 | 9 | 1,5 | цех 1 | 30.04.80 | ДА |
| 2Н125 | 25 | 250 | 12 | 2,2 | цех 1 | 25.10.91 | НЕТ |
| 2Н135 | 35 | 300 | 12 | 4,0 | цех 1 | 27.02.75 | ДА |

## 8.4 Тесты и контрольные вопросы

8.4.1 Какое окно Delphi осуществляет основные функции управления проектом создаваемой программы:

а) окно форм;

б) главное окно;

в) окно инспектора объектов;

г) окно кода программы?

8.4.2 Какое окно Delphi обеспечивает интерфейс пользователя для создаваемого приложения:

а) окно форм;

б) главное окно;

в) окно инспектора объектов;

г) окно кода программы?

8.4.3 Какое окно Delphi предназначено для изменения свойств объектов и управления событиями, на которые реагирует объект:

а) окно форм;

б) главное окно;

в) окно кода программы;

г) окно инспектора объектов?

8.4.4 Для чего предназначено окно форм:

а) для осуществления основных функций управления проектом создаваемой программы;

б) для обеспечения интерфейса пользователя для создаваемого приложения;

в) для изменения свойств объектов и управления событиями, на которые реагирует объект;

г) для редактирования кодов программ, написанных на языке Object Pascal?

8.4.5 Какая программа Delphi позволяет создавать файлы БД:

а) **BDE Administrator**;

б) **Datapump**;

в) **Image Editor**;

г) **DataBase Desktop**;

д) **SQL Explorer**?

8.4.6 Программа **DataBase Desktop** находится в опции:

а) **File**;

б) **Edit**;

в) **Tools**;

г) **View**;

д) **DataBase**.

8.4.7 В диалоговом окне «Create dBASE IV Table» описывается:

а) схема;

б) подсхема;

в) схема хранения;

г) внешняя схема;

д) внутренняя схема.

8.4.8 Какой тип полей не поддерживает dBASE IV:

а) Character; б) Float; в) Money;

г) Logical; д) Number?

8.4.9 Что не задается в диалоговом окне «Create dBASE IV Table»:

а) Fields Name; б) Type; в) Size;

г) Key; д) Dec?

8.4.10 Какой оператор языка SQL позволяет создавать таблицу:

а) **CREATE TABLE**; б) **SELECT**;

в) **ALTER TABLE**; г) **DROP TABLE**?

8.4.11 Для чего предназначена система программирования Delphi?

8.4.12 С помощью какой программы в Delphi создаются базы данных?

8.4.13 Какие форматы данных возможно использовать в Delphi при создании баз данных?

8.4.14 Какая модель данных используется в Delphi?

8.4.15 В чем заключаются достоинства таблиц dBase?

8.4.16 В чем заключаются недостатки таблиц dBase?

8.4.17 Что входит в описание структуры базы данных?

8.4.18 Как осуществляется модификация структуры файла базы данных?

8.4.19 Что такое SQL?

8.4.20 Для чего предназначена команда **SELECT**? Ее формат?

8.4.21 Для чего предназначена команда **CREATE TABLE**? Ее формат?

# 9 Разработка простейшего приложения базы данных в Delphi

Целью работы является разработка в системе программирования Delphi приложения, предназначенного для создания и обслуживания базы данных.

## 9.1 Общие положения

### 9.1.1 Создание таблиц базы данных

Одна из важнейших возможностей Delphi – это возможность создавать и работать с базами данных. Delphi позволяет разрабатывать приложения, предназначенные для создания и изменения электронных хранилищ информации баз данных. Простота и естественность языка, ориентация системы на разработку такого рода приложений, наконец, эффективность (большая производительность и относительно небольшие размеры) создаваемых с ее помощью программ сделали Delphi незаменимым средством разработки клиентских мест (программ для доступа к базе данных) [24].

При работе с таблицами локальных баз данных сама база данных размещается в каталоге на диске и хранится в виде набора файлов. Для хранения одной таблицы создается отдельный файл, такие же отдельные файлы создаются для хранения индексов таблицы и полей Memo.

Обращение к базе данных из утилит и программы осуществляется по псевдониму базы данных. Псевдоним должен быть зарегистрирован в файле конфигурации конкретного компьютера при помощи утилиты **BDE Administrator.** Эта утилита предназначена для установки псевдонимов (имен) баз данных, их параметров (формат даты и времени; форматы представления числовых значений), используемых языковых драйверов и так далее. Для присвоения псевдонима создаваемой базе данных необходимо выполнить следующие действия:

- запустить утилиту BDE Administrator из главного меню Windows командой **Пуск => Программы => Delphi => BDE Administrator**;
- выбрать в главном меню окна утилиты элемент **Object => New**;
- в появившемся окне оставить тип создаваемой базы данных без данных, без изменений (**STANDARD**), как показано на рисунке 9.1, и нажать кнопку **OK**;
- в левом поле окна администратора базы данных изменить строку с именем **STANDARD1** на новое имя, например, LR9;
- в правом поле, в котором указаны параметры базы данных, изменить параметр **PATH** , указывающий маршрут доступа к каталогу, в котором располагается база данных; для этого щелкнуть по полю **PATH** , нажать на появившуюся в правом углу поля кнопку  , выбрать каталог с папкой своей группы и папку с данной лабораторной работой 9 (например, D:\ 06_САП\ ЛР9) и нажать клавишу **OK** (рисунок 9.2);
- запомнить определение псевдонима, для этого в левом окне меню администратора базы данных щелкнуть по имени псевдонима правой кнопкой

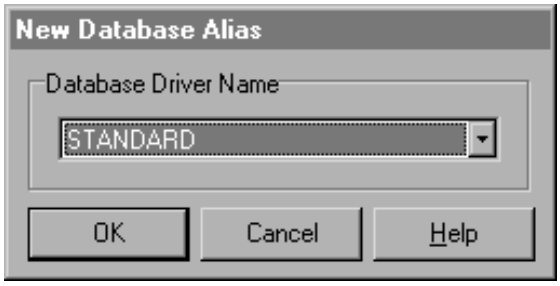



Рисунок 9.1 – Окно выбора драйвера базы данных

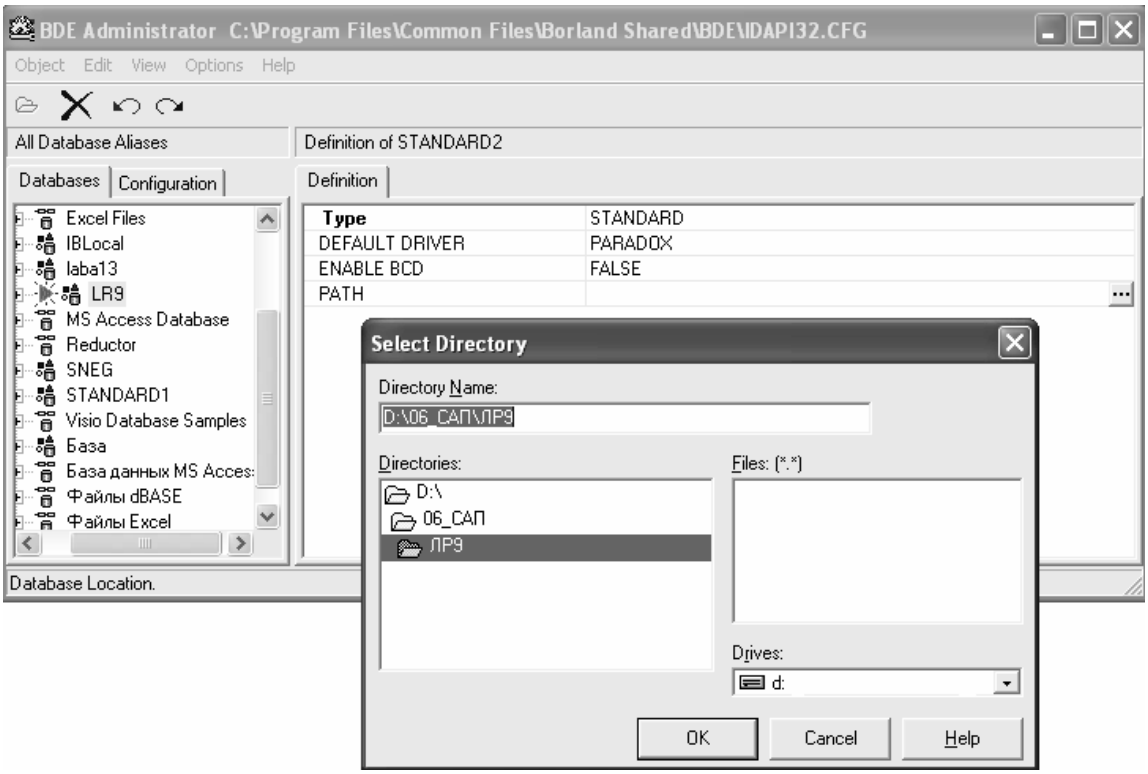



Рисунок 9.2 – Окно параметров псевдонима базы данных

мыши, в дополнительном меню выбрать опцию **Apply**, в появившемся диалоговом окне **Confirm** на вопрос « Save all edits to LR9?» («Запоминать все изменения для псевдонима LR9?») нажать кнопку **OK**;

- закрыть окно утилиты **BDE Administrator**.

Указанными действиями создание псевдонима завершено, и к нему можно обращаться из других утилит и приложений, но каталог, на который ссылается псевдоним базы данных, еще пуст.

Для создания таблиц базы данных необходимо запустить утилиту **Database Desktop (DBD)**, которая является средством для создания, изменения и просмотра базы данных. Запуск утилиты осуществляется с помощью главного меню Windows: **Пуск** => **Все программы** => **Borland Delphi 7** => **Database Desktop** или **Пуск** => **Все программы** => **Borland Delphi 7** => **Delphi 7** => **Tools** => **Database Desktop**.

После запуска утилиты **Database Desktop** необходимо установить псевдоним базы данных. Для этого нужно:

- выбрать команду **File => Working Directory**;
- в открывшемся окне «Set Working Directory» в выпадающем списке **Aliases** выбрать имя псевдонима, которое было ранее сохранено (например, LR9), в окне **Working Directory** задать имя каталога, в котором будет располагаться база данных;
- нажать кнопку **OK**.

Для создания таблицы базы данных нужно выбрать элемент главного меню **File => New => Table**. В появившемся окне **Create Table** выбирается тип создаваемой таблицы списка. В лабораторной работе рекомендуется создать таблицу типа Paradox 7, для этого надо установить курсор на этот тип и нажать кнопку **OK**. После этого появится окно определения структуры таблицы базы данных «Create Paradox 7 Table».

Таблицы Paradox являются достаточно развитыми и удобными для создания баз данных. Они обладают следующими достоинствами:

- большое количество типов полей для предоставления данных различных типов;
- поддержка целостности данных;
- организация проверки вводимых данных;
- поддержка парольной защиты таблиц.

Недостатком таблиц Paradox является наличие относительно большого количества типов файлов, требуемых для хранения содержащихся в таблице данных. При копировании или перемещении какой-либо таблицы из одного каталога в другой необходимо обеспечить копирование или перемещение всех файлов, относящихся к этой таблице.

Каждая строка таблицы соответствует полю. Назначение столбцов:

- имя поля **Fields Name**;
- тип поля **Type**;
- размер поля **Size** (только для строковых полей);
- определение первичного ключа **Key** (установка звездочки «*****» означает, что поле входит в состав первичного ключа; если в первичный ключ входит несколько полей, они должны определяться в той последовательности, в которой они присутствуют в первичном ключе).

Краткая характеристика типов полей для таблиц Paradox приведена в таблице 9.1 [26].

Определение полей, входящих в создаваемые таблицы, осуществляется аналогично предыдущей лабораторной работы 8. Для определения типа поля необходимо щелкнуть по столбцу **Type** и выбрать необходимый тип в контекстном меню из списка типов полей. При задании ключевого поля нажать любой символ на клавиатуре, после этого в столбце **Key** появляется звездочка. Повторное нажатие любого символа снимает отметку в столбце **Key**.

Для каждого поля возможно определение требования обязательного его заполнения значением. Для этого, переходя от поля к полю, необходимо включить переключатели **Required Field**. Также для каждого поля можно наложить

Таблица 9.1 – Характеристика типов полей для таблиц Paradox

| Тип | Обозначение | Хранимые значения |
|---|---|---|
| Alpha | A | Символьные значения длиной до 255 символов |
| Number | N | Числовые значения с плавающей точкой в диапазоне $-10^{307}$ … $+10^{308}$, точность до 15 значащих цифр |
| Money | $ | Предназначен для хранения денежных сумм, число знаков после занятой по умолчанию равно двум |
| Short | S | Целочисленные значения в диапазоне -32768 … 32767 |
| Long Integer | I | Целочисленные значения в диапазоне -2147483648 … 2147483647 |
| BCD | # | Числовые значения, в том числе и дробные, в двоично-десятичном формате |
| Date | D | Значение даты |
| Time | T | Значение времени |
| Memo | M | Строковые значения, длина не ограничена |
| Formatted Memo | F | Форматированный текст произвольной длины, в котором отдельные фрагменты текста могут использовать разные шрифты, цвет и стили, отличается от типа Memo тем, что строка может содержать форматированный текст |
| Timestamp | @ | Значение даты и времени |
| Graphic Fields | G | Графические изображения в форматах BMP, PCX, TIF, GIF, EPS, которые при хранении преобразуются к формату BMP; для хранения изображения используется файл с расширением mb |
| OLE | O | Данные в формате, который поддерживается технологией OLE |
| Logical | L | Логическое значение, допустимы значения True (истина) и False (ложь) |
| Autoincrement | + | Автоинкрементное поле; при добавлении к таблице новой записи в поле автоматически заносится значение, на единицу большее, чем в последней добавленной записи; значение поля доступно для чтения и обычно используется в качестве ключевого поля |
| Binary | B | Последовательность байтов, длина не ограничена; байты содержат произвольное двоичное значение |
| Bytes | Y | Последовательность байтов; длина не более 255 байтов |

следующие ограничения на значение поля:

- минимальное значение поля **Minimum value**;
- максимальное значение поля **Maximum value**;
- значение поля по умолчанию **Default value**;
- шаблон изображения поля **Picture**.

После определения всех полей необходимо сохранить таблицу на диске, для этого нажать кнопку **Save As** и в появившемся окне указать имя. Рекомендуется использовать латинские буквы при задании имени таблицы.

### 9.1.2 Определение ссылочной целостности между таблицами

*Целостность* – свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию [4, 12]. Нарушение целостности базы данных означает, что хранящаяся в ней информация становится недостоверной. СУБД обычно блокирует действия, которые нарушают ссылочную целостность. Нарушение хотя бы одной связи делает информацию в базе данных недостоверной.

Чтобы предотвратить потерю ссылочной целостности, используется механизм каскадных изменений. Он состоит в обеспечении следующих действий:

- при изменении поля связи в записи родительской таблицы следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;

- при удалении записи в родительской таблице следует удалить соответствующие записи в дочерней таблице.

Изменения или удаления в записях дочерней таблицы при одновременном изменении (удалении) записи родительской таблицы называются каскадными изменениями и каскадными удалениями.

Обычно для реализации ссылочной целостности в дочерней таблице создают внешний ключ, в который входят поля связи дочерней таблицы. Этот ключ является для дочерней таблицы первичным и поэтому по составу полей должен совпадать с первичным ключом родительской таблицы.

Для определения ссылочной целостности между таблицами выполняются следующие действия:

- открыть дочернюю таблицу (**File** => **Open** => **Table**);

- выбрать режим изменения структуры таблицы (**Table** => **Restructure**);

- в выпадающем списке **Table Properties** выбрать элемент **Referential Integrity**, нажать кнопку **Define**;

- в появившемся диалоговом окне (рисунок 9.3) в списке **Fields** показаны поля дочерней таблицы, а в списке **Tables** – имена всех таблиц, входящих в базу данных;

- в окне **Fields** из списка полей дочерней таблицы выбрать поле внешнего ключа (поставить курсор) и нажать кнопку с изображением стрелки вправо, теперь название этого поля будет записано в поле **Child Fields**;

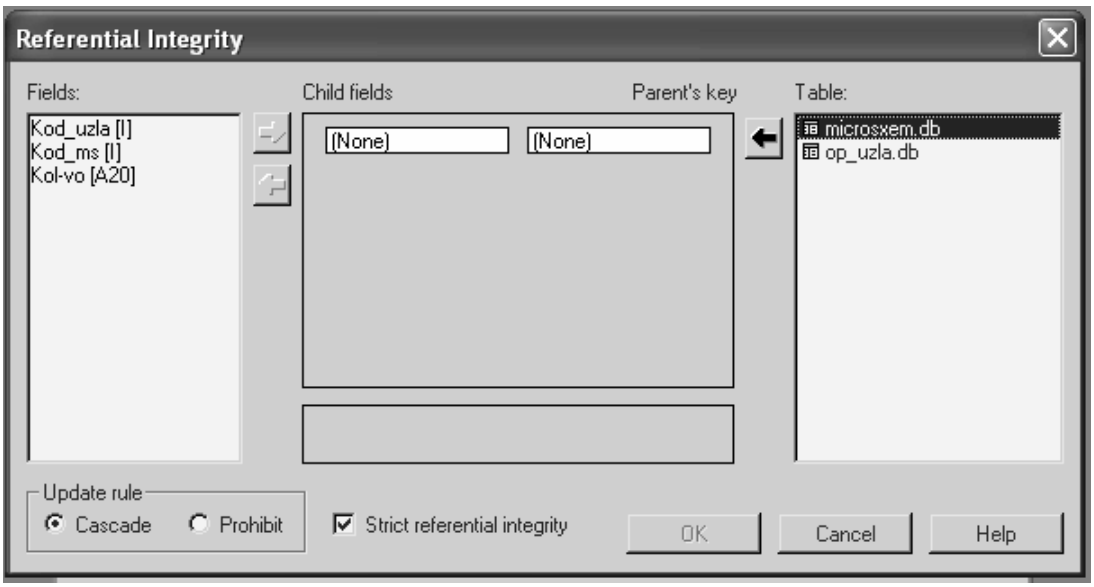



Рисунок 9.3 – Окно определения ссылочной целостности

- выбрать в списке **Tables** родительскую таблицу и нажать кнопку с изображением стрелки влево; в результате в поле **Parents Key** (ключ родительской таблицы) будет показано поле из первичного ключа родительской таблицы.

Переключатели **Update rules** определяют вид каскадных воздействий на дочернюю таблицу при изменении значения поля связи в родительской таблице или при удалении в ней записи:

а) **Cascade** – разрешены каскадные изменения и удаления подчиненных записей в дочерней таблице;

б) **Prohibit** – запрещены изменения полей связи или удаление записи в родительской таблице, если для данной записи есть связанные записи в дочерней таблице.

В лабораторной работе рекомендуется оставить разрешение каскадных изменений.

После определения ссылочной целостности между таблицами необходимо нажать **OK**. Утилита **Database Desktop** (DBD) запросит имя ссылочной целостности (в Paradox ссылочные целостности именуются). Необходимо ввести имя, например «< имя родительской таблицы >_< имя дочерней таблицы >_ Integrity», и нажать кнопку **OK**. Теперь имя созданной ссылочной целостности будет помещено в список. Затем необходимо сохранить изменения в дочерней таблице (кнопка **Save**), выйти из режима реструктуризации (кнопка **Cancel**) и закрыть окно **Database Desktop**.

### 9.1.3 Создание простейшего приложения

Система программирования Delphi является средством быстрой разработки приложений. Она сочетает в себе особенности визуального и объектно-ориентированного программирования с дружественной программисту средой разработки, архитектура которой построена на компонентах. Интегрированная среда разработки Delphi представляет собой многооконную систему, ее вид

(интерфейс) может различаться в зависимости от настроек. После загрузки интерфейс Delphi первоначально имеет четыре окна: главное окно (**Delphi 7 – Project1**); окно Инспектора объектов (**Object Inspector**), отражающее свойства и события объектов событий для текущей формы; окно Конструктора формы (**Form1**), в котором выполняется проектирование формы; окно Редактора кода (**Unit1.pas**), содержащее исходный текст модуля разрабатываемого приложения (рисунок 8.1).

В главном окне отображаются:

- главное меню, содержащее обширный набор команд для доступа к функциям Delphi;

- панели инструментов, которые находятся под главным меню в левой части главного окна и содержат 15 кнопок для вывода часто используемых команд главного меню;

- палитра компонентов, находящаяся под главным меню в правой части главного окна и содержащая множество компонентов, размещаемых в создаваемых формах.

Компоненты являются своего рода строительными блоками, из которых конструируются формы приложения. Все компоненты делятся на группы, каждая из которых в Палитре компонентов располагается на отдельной вкладке (странице), а сами компоненты представлены соответствующими значками (пиктограммами). Нужная вкладка выбирается щелчком левой кнопки мыши на ее ярлычке. Поместить выбранный компонент на форму очень просто – надо сделать щелчок мышью в нужном месте формы.

Компоненты, используемые для создания приложений БД, делятся на:

- невизуальные, предназначенные для организации доступа к данным, содержащимся в таблицах; они представляют собой промежуточное звено между данными таблиц БД и визуальными компонентами;

- визуальные, используемые для создания интерфейсной части приложения; с их помощью пользователь может выполнять такие операции с таблицами БД, как просмотр или редактирование данных.

В лабораторной работе используются следующие страницы палитры компонентов:

**- Standard** (Стандартная);

**- Data Access** (Доступ к данным);

**- Data Control** (Управление данными);

**- BDE** (Управление данными с использованием механизма BDE).

На странице **Standard** используются такие визуальные компоненты:

- метка **Label** (  ), применяемая для размещения текста, который не изменяется пользователем;

- компонент редактирования **Edit** (  ) , используемое для ввода пользователем однострочных текстов, для отображения текста;

- командная кнопка **Button** (  ), используемая для создания кнопок, которыми пользователь выбирает команды в приложении.

На странице **BDE** используются невизуальные компоненты:

- набор данных **Table** (  ) для установления связи приложения с таблицей базы данных;

- набор данных **Query** (  ) для построения и выполнения SQL – запросов к удаленным SQL – серверам или локальным базам данных;

На странице **Data Access** используется невизуальный компонент: источник данных **DataSource** (  ) для соединения компонентов типа Table или Query с компонентами, отображающими данные.

На странице **Data Controls** находятся такие визуальные компоненты, как:

- сетка **DBGrid** (  ) для создания ориентированных на данные сеток (таблиц) с отображением данных в строках и столбцах;

- поле редактирования **DBEdit** (  ), представляющий собой ориентированный на данные вариант компонента Edit.

При запуске Delphi по умолчанию создает пустую форму для нового приложения. Создадим приложение для ввода данных в родительскую таблицу.

В палитре компонентов на странице **BDE** выбрать мышью компонент **Table** (  ), щелкнуть на нем мышью и затем щелкнуть мышью на форме. После этого изображение компонента останется в форме.

Для созданного компонента необходимо задать свойства (Properties) в Инспекторе Объектов (Object Inspector):

- установить в свойство **DatabaseName** при помощи выпадающего меню или вручную значение псевдонима базы данных, который был задан ранее (пункт 9.1.1);

- установить в свойство **TableName** имя родительской таблицы базы данных;

- для свойства **Active** установить значение True.

В этот момент произойдет реальное связывание компонента **Table** (по умолчанию имеет имя Table1) с таблицей.

В качестве связующего звена между невизуальными компонентами (в данном случае Table1) и визуальными компонентами используют компонент **DataSource** (  ), который часто называют источниками данных. Необходимо расположить на форме этот компонент, выбрав его в палитре компонентов на странице **Data Access**, и установить в свойство **DataSet** (имя набора данных) компонента значение Table1, выбрав его из выпадающего списка.

Для создания ориентированной на данные таблицы расположить в форме компонент **DBGrid**, взяв его из палитры компонентов на странице **Data Controls**. В свойство **DataSource** устанавливается значение DataSource1 (это имя, присвоенное по умолчанию созданному перед этим компоненту **DataSource**), выбрав его из выпадающего списка.

Вид разрабатываемой формы представлен на рисунке 9.4.

После создания формы необходимо сохранить форму и проект на диске. Для этого выбрать в главном меню команду **File => Save All** и ввести имена сохраняемых модуля и проекта.

Созданную форму можно использовать для ввода данных, предварительно запустив программу командой **Run => Run**. В результате выполнится компиляция программы (*компиляция* - преобразование исходного текста програм-

мы в последовательность исполняемых машинных команд [12]) и сформируется файл <имя проекта>.exe.

Работающая программа откроет непосредственный доступ к данным в таблице. Для добавления записи нужно нажать на клавиатуре клавишу **Insert** или, находясь на последней записи набора данных, клавишу смещения курсора вниз. Таблица автоматически перейдет в режим добавления новой записи. После ввода значений в поля записи запомнить запись в наборе данных можно, перейдя на другую запись при помощи клавиш управления курсором. Отказаться от запоминания записи можно, нажав кнопку **Esc**. Для изменения записи следует переместить указатель текущей записи в нужное место и изменить значения там, где это необходимо. Набор данных автоматически перейдет в режим редактирования. Для удаления записи следует установить на нее указатель текущей записи и нажать **Ctrl+Del**.

На рисунке 9.5 показан вид окна программы в момент добавления в таблицу новой записи.

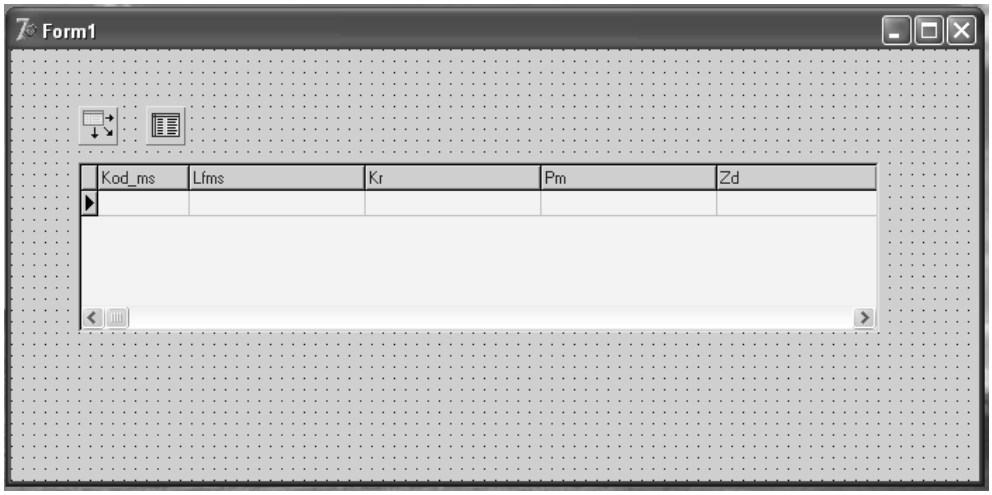



Рисунок 9.4 – Вид формы на этапе разработки

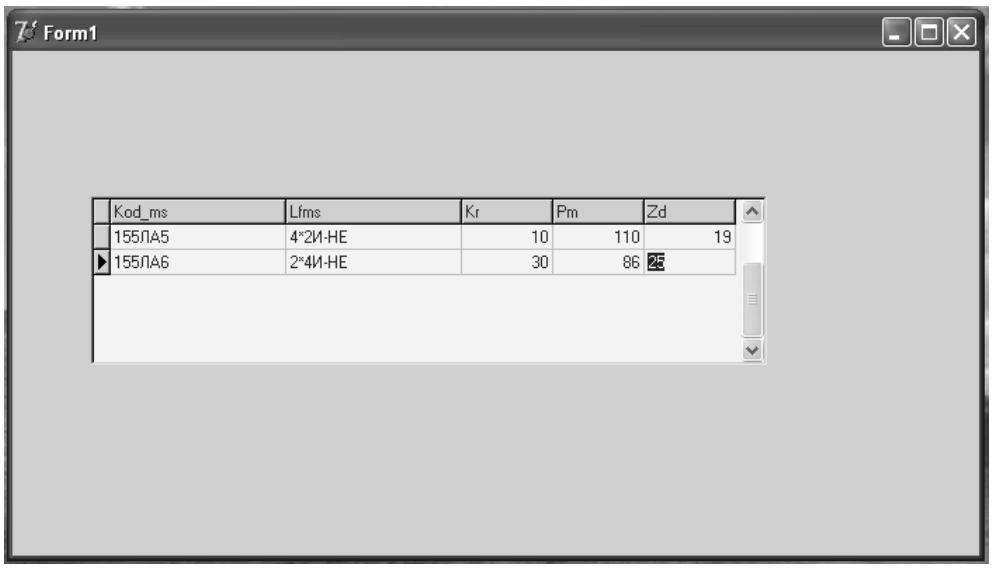



Рисунок 9.5 - Добавление новой записи в таблицу

### 9.1.4 Создание приложения для работы с двумя таблицами

В одной форме можно связывать два набора данных (главный и подчиненный) так, чтобы в подчиненном наборе всегда показывались записи, соответствующие текущей записи в главном наборе.

Для перехода от работающей программы к форме, необходимо выполнить команду **File => Open =>** <имя модуля >.pas. В результате откроется главная форма. Для создания на ней подчиненной необходимо:

- поместить на форму компонент **Table** (с именем Table2), настроить на работу с дочерней таблицей (в свойство **DatabaseName** задать псевдоним базы данных, в свойство **TableName** имя дочерней таблицы базы данных, в свойство **Active** – значение True);
- поместить на форму компонент **DataSource** (с именем DataSource2), установить в свойство **DataSet** значение Table2;
- разместить на форме компонент **DBGrid** (с именем DBGrid2);
- установить в его свойство **DataSource** значение DataSource2.

Чтобы содержимое подчиненного набора соответствовало выбору записи в главном наборе, дочернюю (подчиненную) таблицу нужно связать с родительской (главной). Для этого выполняются следующие действия:

- активизировать компонент Table2;
- в окне Инспектора Объектов раскрыть список выбора в свойстве **MasterSource**, выбрать единственное имеющееся в нем значение DataSource1;
- щелкнуть по правой части строки **MasterFields**, по появившейся в ней кнопке с тремя точками, чтобы раскрыть окно редактора связей;
- раскрыть список **Available Indexes** в верхней части окна и выбрать индекс, соответствующий внешнему ключу дочерней таблицы, в результате в окошке **Detail Fields** появится имя этого поля связи, щелкнуть по нему;
- выбрать это же поле в окошке **Master Fields**, затем щелкнуть по нему;
- нажать кнопку **Add**.

Связь между таблицами будет установлена, в окошке **Joined Fields** появится схема связи по имени поля (рисунок 9.6). Закрыть окно редактора связей можно кнопкой **OK**.

В созданном приложении можно задать имена таблиц и названия столбцов, используя русские наименования.

Для создания имени таблицы используется компонент **Label** ( A ). Необходимо поместить над компонентами **DBGrid** взятый из палитры компонентов на странице **Standart** компонент **Label** и установить в его свойство **Caption** значение, соответствующее русскому названию таблицы.

Для изменения названий столбцов необходимо выполнить следующие действия:

- щелкнуть по компоненту **DBGrid** правой клавишей мыши и выбрать в появившемся меню элемент **Columns Editor**;
- в появившемся окне редакторе столбцов компонента (рисунок 9.7а) для изменения характеристики столбцов (перейти от неявно определяемым столб-

цам к явно определяемым) щелкнуть по кнопке **Add All Fields** ( ) на инструментальной панели окна редактора (рисунок 9.7б);

- щелкнуть по имени редактируемого столбца в списке полей;
- раскрыть в окне Инспектора Объектов список свойства **Title** (для этого нужно либо дважды щелкнуть по имени свойства мышью, либо щелкнуть по нему правой кнопкой и выбрать **Expand**);
- в элементе **Caption** изменить заголовок столбца на русское наименование.

После изменения заголовков необходимо закрыть редактор столбцов кнопкой закрытия ( ), а затем сохранить все сделанные изменения. Если теперь запустить приложение командой **Run**, то ввод и изменение данных в дочерней таблице можно производить с помощью компонента DBGrid2 (рисунок 9.8). При этом перемещение указателя в главной таблице приводит к автоматической смене информации в отображаемых данных дочерней таблицы.

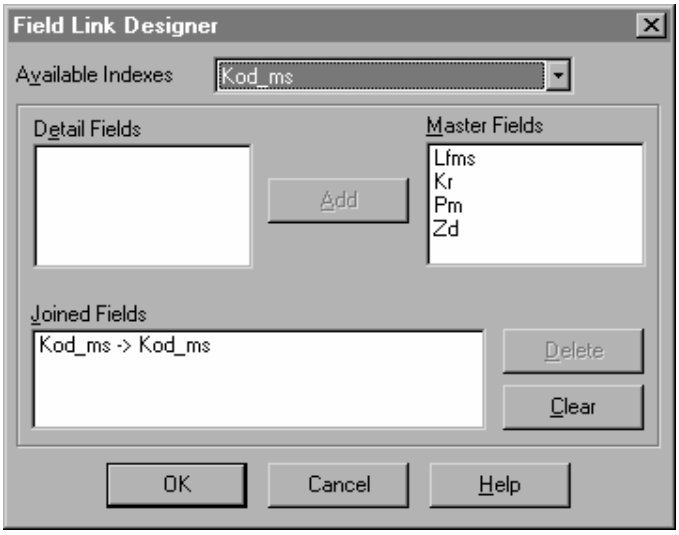
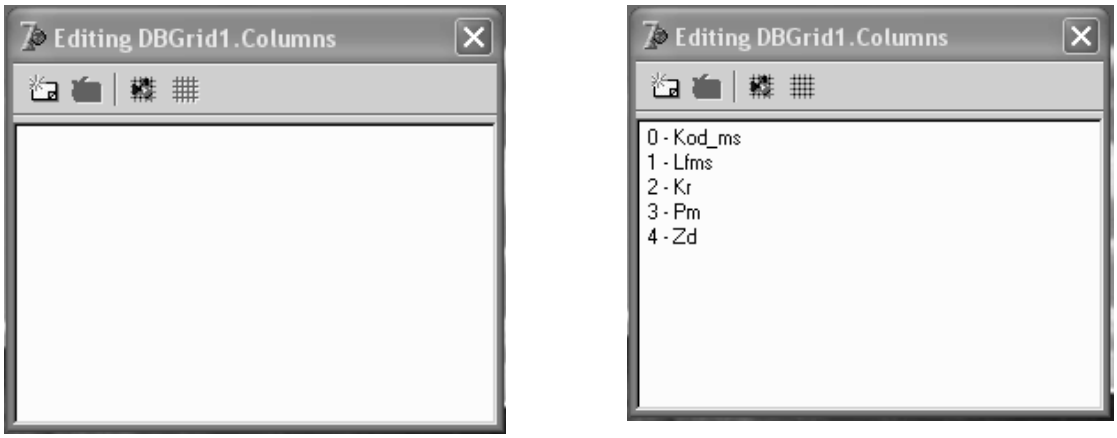



Рисунок 9.6 - Окно редактора связей




а б

а - пустой список столбцов; б - заполненный список

Рисунок 9.7 - Окно редактора столбцов

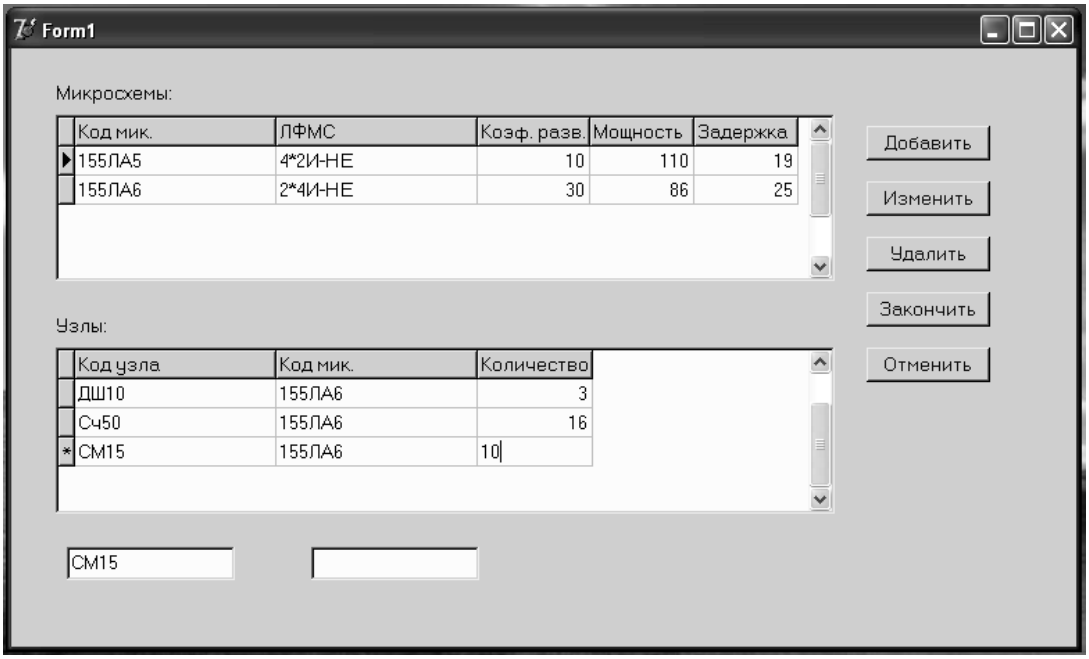



Рисунок 9.8 - Работа формы с главными и подчиненными наборами данных

### 9.1.5 Определение визуальных компонентов для работы с полями

Для улучшения интерфейса программы на форме можно добавить кнопки (визуальные компоненты), с помощью которых возможно добавление, изменение, удаление, запоминание данных. Для этих целей используется компонент **Button**, взятый со страницы **Standard** палитры компонентов.

Поместим на форму компонент **Button** для добавления данных. Для этого компонента необходимо установить свойство **Name** равное InsertButton, свойство **Caption** – «Добавить». Перейдя на обработчик событий **On Click** двойным нажатием на компонент, написать следующий обработчик события:

```
«procedure TForm1.InsertButtonClick(Sender: TObject);
  begin
  if Table2.State = DsBrowse then
     Table2.Insert;
  end;».
```

Примечание – В данной и последующих лабораторных работах при задании программного кода вводить кавычки не нужно.

Если набор данных Table2 находится в режиме просмотра **DsBrowse**, метод **Insert** переводит его в состояние добавления записи **DsInsert**.

Для редактирования данных на форму добавляется компонент **Button** с именем EditButton (свойство **Name**), надписью на кнопке «Изменить» (свойство **Caption**), обработчиком события:

```
«procedure TForm1.EditButtonClick(Sender: TObject);
  begin
  if Table2.State=DsBrowse then
     Table2.Edit;
  end;».
```

Метод **Edit** переводит набор данных Table2 в состояние добавления записи **DsEdit**.

Для удаления записей помещается на форму компонент **Button** с именем DeleteButton (свойство **Name**), надписью на кнопке «Удалить» (свойство **Caption**) и обработчиком события:

```
«procedure TForm1.DeleteButtonClick(Sender: TObject);
  begin
  if Table2.State=DsBrowse then
  if MessageDlg('Подтвердите удаление записи',MtConfirmation, [MBYes, MBNo],0) = MRYes then
  Table2.Delete;
  end;».
```

Если набор данных Table2 находится в режиме просмотра записей **DsBrowse**, вызывается диалоговое окно **MessageDlg** с предложением подтвердить удаление записи. Если пользователь нажимает кнопку **Yes**, текущая запись в наборе данных Table2 удаляется методом **Delete**.

Для создания визуального компонента, позволяющего запоминать записи, необходимо:

- поместить на форме компонент **Button**;
- установить в свойство **Name** значение PostButton;
- установить в свойство **Caption** значение «Запомнить»;
- написать обработчик событий:

```
«procedure TForm1.PostButtonClick(Sender: TObject);
  begin
  if Table2.State in [DsInsert, DsEdit] then
     Table2.Post;
  end;».
```

Если набор данных находится в режиме добавления новой записи или редактирования, вызывается метод **Post** набора данных **Table2**, который запоминает текущее состояние записи в таблице базы данных. После запоминания набор данных автоматически переводится в режим просмотра **DsBrowse**.

Отмену последних произведенных действий можно осуществить с помощью кнопки, создать которую можно следующим образом:

- поместить на форму визуальный компонент **Button**;
- установить в свойство **Name** значение CancelButton;
- установить в свойство **Caption** значение «Отменить»;
- написать обработчик событий:

```
«procedure TForm1.CancelButtonClick(Sender: TObject);
  begin
  if Table2.State in [DsInsert, DsEdit] then
     Table2.Cancel;
  end;».
```

Если набор данных находится в режиме добавления новой записи или редактирования, вызывается метод **Cancel**, который отменяет сделанные изменения и переводит набор данных в режим просмотра **DsBrowse**.

На форму можно поместить компоненты **DBEdit1** и **DBEdit2** для отображения значений полей «Код узла» и «Код микросхемы». Для связи компонентов с полями необходимо поместить в их свойства **DataSource** значения соответственно DataSource1и DataSource2, а в свойства **DataField -** имена полей. Если после сохранения модуля и проекта запустить программу, то при добавлении новой записи или при корректировке существующей в поле можно заносить значения, используя эти компоненты.

### 9.2.6 Формирование набора данных из нескольких таблиц

Delphi позволяет создавать набор данных из нескольких таблиц с помощью невизуального компонента **Query** (  ) на странице палитры компонента **BDE**. Этот компонент реализует набор данных, источником для которого являются одна или несколько таблиц базы данных, структура записи и состав которого определяется SQL –запросом.

В качестве примера рассмотрим создание следующего приложения. Пусть набор данных собирается из двух таблиц: «Микросхемы» (microsxem.db) и «Описание узла» (op_uzla.db). При этом соединяются записи, имеющие одинаковое значение поля «Код микросхемы» (Kod_ms). В создаваемый набор данных должны входить поля «Код узла» (Kod_uzla), «Код микросхемы» (Kod_ms), «Количество» (Kol-vo) из таблицы «Описание узлов», из таблицы «Микросхемы» поля «Логическая функция микросхемы» (Lfm) и вычисляемое поле «Потребляемая мощность узла» (Power) со значением [«Количество» * «Потребляемая мощность»].

Для создания приложения необходимо выполнить следующие действия:

- открыть пустую форму **Form2**, выполнив команду главного меню **File => New Application**;

- расположить на форме компонент **Query,** взяв его со страницы **BDE** палитры компонентов, установить в свойство **Query1.DatabaseName** значение псевдонима базы данных (например, LR9);

- расположить на форме компонент **DataSource**, связать этот компонент с компонентом **Query1**, установив в свойство **DataSet** значение Query1;
- расположить на форме компонент **DBGrid** со страницы **DataControls**, связать этот компонент с DataSource1, установив в свойство **DataSource** значение DataSource1;
- активизировать компонент **Query1**, раскрыть редактор свойства **SQL**;
- ввести текст SQL – запроса

```
«SELECT O.Kod_uzla, O.Kod_ms, O.Kol-vo, M.Lfms, (M.PM*O.Kol-vo) As Power
  FROM microsxem M,op_uzla O
  WHERE M.Kod_ms=O.Kod_ms
  ORDER BY O.Kod_uzla, O.Kod_ms»;
```

- закрыть редактор кнопкой **OK**;
- установить в свойство **Query1.Active** значение True;
- сохранить созданный модуль и проект.

Запустив программу (**Run**), на экране появится набор данных, аналогично показанному на рисунке 9.9. В таблице изменить каким-либо образом отображаемые в DBGrid1 данные нельзя, так как они получены в результате выполнения оператора **SELECT** языка SQL, который объединил данные из двух таблиц.

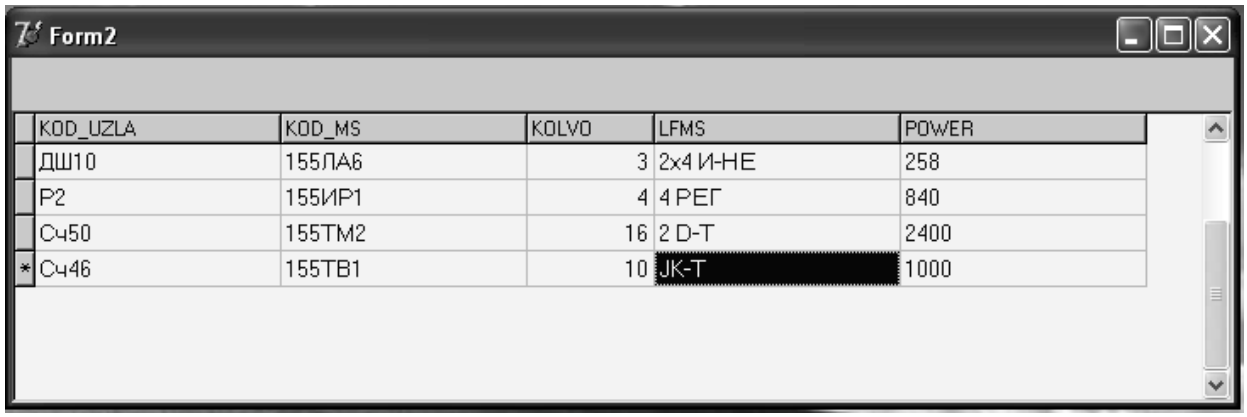



Рисунок 9.9 - Создание данных из разных таблиц в одном наборе данных

### 9.2 Задание на выполнение работы

9.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к лабораторной работе 9. При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его определение в созданном каталоге.

9.2.2 Запустить утилиту **Database Desktop** и установить умалчиваемый псевдоним.

9.2.3 Определить структуры таблиц «Микросхемы» и «Описание узла», содержание которых приведены в таблицах 9.2 и 9.3, сохранить соответствующие им файлы, как описано в пункте 9.1.1. Тип создаваемых таблиц - Paradox 7.

9.2.4 Определить ссылочную целостность между таблицами, описание определения которой приведено в пункте 9.1.3.

9.2.5 Создать простейшее приложение для добавления записей в таблицу «Микросхемы», запустить программу и ввести записи.

9.2.6 Создать приложение для работы с двумя таблицами, где таблица «Микросхемы» – родительская, а «Описание узла» – дочерняя; ввести заголовки этих таблиц, изменить названия столбцов на русские наименования.

9.2.7 Запустить приложение, созданное в пункте 9.2.6 и ввести данные в подчиненную форму.

9.2.8 Создать кнопки «Добавить», «Изменить», «Удалить», «Запомнить», «Отменить» для работы с полями при помощи визуальных компонентов **Button**. Протестировать созданную программу.

9.2.9 Сформировать набор данных из двух таблиц «Микросхемы» и «Описание узла», как описано в подразделе 9.1.6, запустить программу.

9.2.10 Оформить отчет по лабораторной работе.

Таблица 9.2 – Содержание таблицы «Микросхемы»

| Код типа микросхемы | Логическая функция микросхемы | Коэффициент разветвления | Потребляемая мощность | Задержка |
|---|---|---|---|---|
| 155ЛА3 | 4х2 И-НЕ | 10 | 110 | 19 |
| 155ЛА6 | 2х4 И-НЕ | 30 | 86 | 25 |
| 155ТМ2 | 2 D-Т | 10 | 150 | 55 |
| 155ТВ1 | JK-Т | 10 | 100 | 55 |
| 155ИР1 | 4 РЕГ | 10 | 410 | 35 |

Таблица 9.3 – Содержание таблицы «Описание узла»

| Код узла | Код типа микросхемы | Количество |
|---|---|---|
| Р2 | 155ЛА3 | 16 |
| Р2 | 155ИР1 | 4 |
| ДШ10 | 155ЛА3 | 2 |
| ДШ10 | 155ЛА6 | 3 |
| Сч50 | 155ТМ2 | 16 |
| Сч50 | 155ЛА6 | 16 |
| Сч46 | 155ТВ1 | 10 |
| Сч46 | 155ЛА3 | 4 |
| Р3 | 155ИР1 | 8 |
| Р3 | 155ЛА3 | 8 |
| СМ5 | 155ТМ2 | 8 |
| СМ5 | 155ЛА6 | 10 |
| СМ15 | 155ТМ3 | 8 |
| СМ15 | 155ЛА3 | 6 |
| Р1 | 155ТВ1 | 16 |

## 9.3 Содержание отчета

9.3.1 Название работы.
9.3.2 Цель работы.
9.3.3 Структуры таблиц «Микросхемы» и «Описание узла».
9.3.4 Макет формы с главным и подчиненным наборами данных и визуальными компонентами.
9.3.5 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

## 9.4 Тесты и контрольные вопросы

9.4.1 Какая программа Delphi позволяет зарегистрировать псевдоним БД:
а) **BDE Administrator**; б) **Datapump**; в) **Image Editor**;

г) **DataBase Desktop**; д) **SQL Explorer**?

9.4.2 Какой тип полей не поддерживает Paradox 7:

а) **Alpha**; б) **Float**; в) **Money**;

г) **Logical**; д) **Long Integer**?

9.4.3 Что не задается в диалоговом окне «Create Paradox 7 Table»:

а) **Fields Name**; б) **Type**; в) **Size**;

г) **Key** д) **Dec**?

9.4.4 Целостность – это:

а) свойство, характеризующее сущность;

б) средство ускорения операции поиска записей в таблице;

в) свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию;

г) свойство программы, позволяющее проверять правильность записей базы данных.

9.4.5 Во внешний ключ входят:

а) поля дочерней таблицы;

б) поля связи дочерней таблицы;

в) поля связи родительской таблицы;

г) поля родительской таблицы.

9.4.6 Компонент, предназначенный для установления связи приложения с таблицей базы данных,:

а)  б)  в)  г)  д) 

9.4.7 Какой компонент является невизуальным:

а) **DataSource**; б) **Label**; в) **Button**;

г) **DBEdit**; д) **DBGrid**?

9.4.8 Компиляция – это:

а) непосредственное исполнение текста исходной программы в ходе просмотра ее текста;

б) создание специальных копий базы данных, с которыми пользователи могут работать одновременно на разных рабочих станциях;

в) преобразование исходного текста программы в последовательность исполняемых машинных команд;

г) последовательность операций над базой данных, отслеживаемая системой управления базами данных от начала до завершения как единое целое;

д) преобразование исполняемых машинных команд программы в исходный текст.

9.4.9 Какой компонент является визуальным:

а) **DataSource**;

б) **DBGrid**;

в) **Table**;

д) **Query**?

9.4.10 Какой компонент можно использовать для задания имени таблицы:

а) **DataSource**;

б) **Label**;

в) **Button**;

г) **DBEdit**;

д) **DBGrid**?

9.4.11 Для чего предназначена утилита **BDE Administrator**?

9.4.12 Как создать псевдоним базы данных?

9.4.13 Для чего предназначена утилита **Database Desktop**?

9.4.14 Как установить умалчиваемый псевдоним базы данных?

9.4.15 Что можно задать в окне **Create Table**?

9.4.16 Что задается в окне определения структуры таблицы базы данных?

9.4.17 Какие типы данных используются для таблиц Paradox?

9.4.18 Для каких полей в столбце **Key** помещается звездочка?

9.4.19 Как запомнить структуру таблицы?

9.4.20 Что отображается в главном окне?

9.4.21 Какие страницы содержит палитра компонентов?

9.4.22 Для каких целей определяется ссылочная целостность между таблицами?

9.4.23 Как определяется ссылочная целостность между таблицами?

9.4.24 Какие окна содержит Delphi?

9.4.25 Какие визуальные компоненты со страницы **Standard** использовались при выполнении лабораторной работы?

9.4.26 Что означает «визуальные» и «невизуальные» компоненты?

9.4.27 Какие невизуальные компоненты использовались со страницы **Data Access**?

9.4.28 Какие свойства компонента **Table** задаются для связи приложения с таблицей данных?

9.4.29 Какой компонент используется для создания ориентированной на данные таблицы?

9.4.30 Для чего используется компонент **DataSource**?

9.4.31 Как запустить программу?

9.4.32 Для каких целей создается форма с подчиненной формой?

9.4.33 Какие действия выполняются для связи подчиненной формы с главной?

9.4.34 С помощью какого компонента можно задать имя таблицы?

9.4.35 Каким образом можно изменить названия столбцов на русские наименования?

9.4.36 Для чего предназначен компонент **Button**?

9.4.37 Что необходимо задать для компонента **Button**, чтобы он действовал как кнопка?

9.4.38 Какие действия необходимо выполнить для создания набора данных из нескольких таблиц с помощью компонента **Query**?

9.4.39 Объясните содержание текста SQL – запроса.

9.4.40 Как в SQL – запросе создать вычисляемое поле?

# 10 Управление и фильтрация в базе данных в Delphi

Целью работы является знакомство с различными видами представления меню, создание пользовательского приложения по работе с базой данных, которое позволяет управлять данными и осуществлять фильтрацию записей.

## 10.1 Общие положения

### 10.1.1 Создание меню

Разработка приложения в Delphi осуществляется в окне Конструктора формы **Form1**, которое наряду с главным окном Delphi – **Project1** и окном инспектора объектов появляется после загрузки Delphi командой **Пуск** => **Программы** => **Borland Delphi** => **Delphi.**

При первой загрузке Delphi по умолчанию создает «пустой» файл проекта. Для создания собственного проекта необходимо выполнить команду **File => New => Application**. В результате создания нового проекта на экране появится окно Форм с заголовком Form1 и окно редактора кода с модулем Unit1.pas и файлом проекта Project1.dpr, эти окна можно размещать так, как удобно пользователю. В одно и то же время может быть открыт только один проект, поэтому перед сохранением нового проекта Delphi запрашивает, нужно ли сохранить текущее приложение.

Форма является основой, на которой размещаются необходимые компоненты, используемые приложением для выполнения возложенных на него функций и задач. Создание приложений в Delphi включает следующие этапы:

а) создание визуального интерфейса приложения, для этого выбираются необходимые компоненты из Палитры компонентов и размещаются на форме;

б) установка при помощи окна **Object Inspector** свойств формы и элементов управления (Инспектора объектов), что необходимо для придания объектам определенного вида и законов их поведения;

в) связывание с компонентами кода на языке Object Pascal для обработки событий, возникающих при использовании мыши, клавиатуры, системных событий и тому подобное.

Создание приложений при помощи визуальных компонентов следует широко применяемому в различных программных продуктах принципу WYSIWYG (What You See Is What You Get – что видишь, то и получишь) и поэтому то, что появляется на экране во время разработки, будет соответствовать тому, что будет наблюдаться во время выполнения приложения.

Простейшим компонентом для обработки текста является **Label** (Метка), доступным на странице **Standard** ( A ). Компонент представляет собой статический текст и применяется для идентификации других объектов приложения, он может помочь пользователю сориентироваться и обеспечит его необходимой информацией.

Пусть необходимо создать форму в виде, представленном на рисунке 10.1. Для создания заголовка формы помещается компонент **Label**, для того необходимо выбрать его на странице **Standard**, щелкнув на нем левой кнопкой

мыши. Затем передвинуть указатель к тому месту, где надо разместить объект. В окне Инспектора объектов необходимо задать свойства компонента. Для любого элемента управления значение свойства **Caption** всегда соответствует надписи на самом элементе управления. Поэтому для задания заголовка формы с помощью компонента **Label** установим значение его свойства **Caption** – «Выбор группы станков». При запуске приложения введенное значение будет той же надписью, которое отобразится на элементе управления.

Для работы многих приложений требуется выбор одного из вариантов, перечень которых задается в виде списка. Создание такого режима можно организовать с помощью визуального компонента **RadioButton** (радиокнопка), расположенного на странице **Standard** ( ). Компоненты **RadioButton**, собранные в одну группу, могут применяться для создания списка. В один момент может быть выбран только один из группы переключателей. Набор альтернатив, из которых выбирается одна, реализуется требуемым количеством радиокнопок, размещенных в одном контейнере (форме, панели и тому подобное).

Для реализации формы «Выбор группы станков» (рисунок 10.1) на форме располагается пять компонентов **RadioButton**. Каждому компоненту необходимо установить в окне Инспектора объектов свое значение свойства **Caption**: кнопке RadioButton1 - «Токарные»; RadioButton2 - «Сверлильные»; RadioButton3 - «Шлифовальные»; RadioButton4 - «Фрезерные»; RadioButton5 - «Выход».

Пусть только выбор варианта «Токарные» из предложенного списка приведет к дальнейшей работе приложения, остальные группы станков недоступны, а выбор «Выход» позволит закрыть работу приложения. Для перехода к следующему шагу работы проекта необходимо на создаваемой форме разместить компонент **Button** (Кнопка) OK со страницы **Standart** . Для компонента **Button** в свойство **Caption** вводится значение «Далее».

Поскольку нажатие кнопки должно привести к определенным полезным действиям, необходимо задать набор событий, которые этот объект может об-

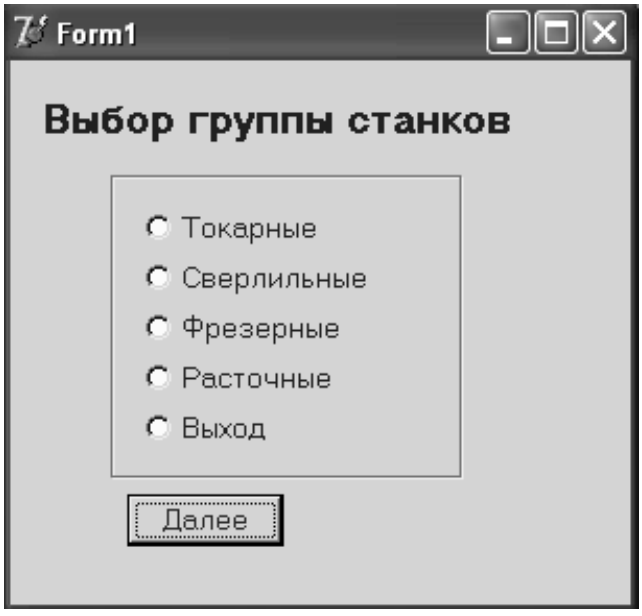



Рисунок 10.1 - Форма «Выбор группы станков»

рабатывать. Для генерации блока кода обработчика события надо сделать двойной щелчок мышью на объекте. Курсор окажется в окне Редактора кода. Курсор будет находиться внутри пустой процедуры **Button1Click**, в которой между операторами цикла **begin…end** нужно ввести необходимый программный код. Процедура для кнопки **Button** должна выглядеть следующим образом:
«procedure TForm1.Button1Click(Sender: TObject);
begin
if RadioButton1.Checked then form2.showmodal;
if RadioButton2.Checked then showmessage('Данная группа станков недоступна');
if RadioButton3.Checked then showmessage('Данная группа станков недоступна');
if RadioButton4.Checked then showmessage('Данная группа станков недоступна');
if RadioButton5.Checked then form1.Close;
end;».

Примечание – Кавычки при вводе программного кода не вводятся.

Если был выбран вариант «Токарные» в форме «Выбор группы станков», то на экране должна появиться следующая форма **Form2**, в которой предлагается выбрать тип станка (рисунок 10.2). Для открытия новой формы выполняется команда **File** => **New** => **Form.** Эта команда создаст новую форму и добавит ее к текущему проекту.

Создание главного и выпадающего меню возможно с помощью Конструктора меню, который позволяет также проверить структуру меню во время разработки приложения. Необходимо выполнить следующие этапы:

а) поместить невизуальный компонент **MainMenu** (главное меню)  со страницы **Standard** на форму;

б) сделать двойной щелчок мышью на компоненте меню, в результате будет выделен первый пункт меню, который еще предстоит определить;

в) ввести имя меню верхнего уровня (например, «Типы станков») и нажать клавишу **Enter**, в результате имя пункта меню верхнего уровня появится на панели меню приложения;

г) заполнить подпункты первого пункта нужными именами;

д) заполнять следующие пункты и подпункты до тех пор, пока не будет создана задуманная структура.

При создании меню появляются пустые графы ниже последнего подпункта в выпадающем меню и вправо от последнего пункта в окне меню. На них щелкают, когда требуется добавить новые пункты. Подразделы выпадающего меню можно отделять разделительной линией между ними, для этого используют вместо подписи несколько символов минус (-). Назначать имя разделителю не обязательно. Он полезен для визуального разбиения пунктов меню на отдельные группы. Его применение сделает интерфейс приложения более удобным и дружественным пользователю.

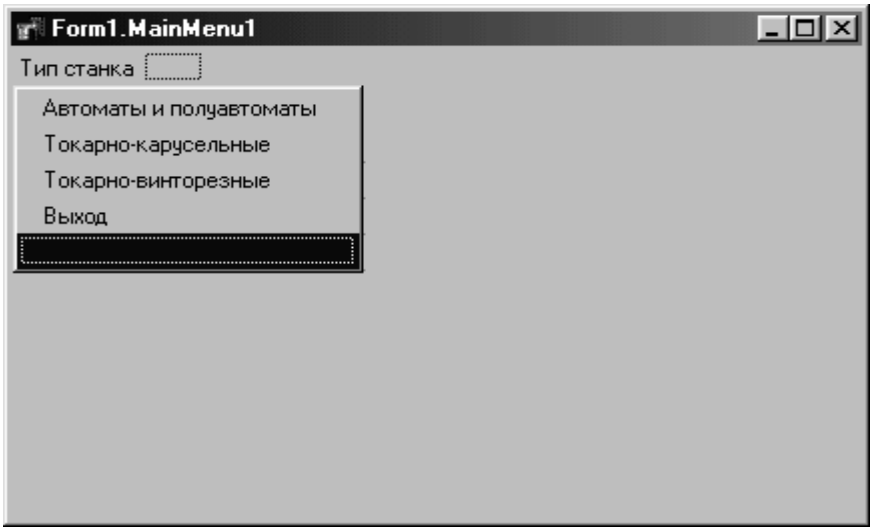



Рисунок 10.2 - Конструктор меню «Выбор типа станка»

Во время разработки приложения или во время его выполнения некоторые пункты меню можно сделать недоступными. Деактивация пунктов меню выполняется следующим образом:

- выбрать редактируемый пункт (или подпункт меню);
- дважды нажать на нем мышью;
- в Инспекторе объектов установить свойству **Enabled** значение False.

В результате выполненных действий в недоступных пунктах текст отобразится серым цветом.

Закончив построение меню и щелкнув на кнопке закрытия окна конструктора меню, можно просмотреть пункты меню, как в обычном меню, выбирая их или проводя над ними мышь с нажатой кнопкой.

Чтобы добавить код для любого из пунктов, необходимо дважды щелкнуть на нужном пункте меню, в результате откроется участок кода, соответствующий процедуре обработки события выбора этого пункта.

В качестве примера ниже приведены процедуры, обеспечивающие в форме, приведенной на рисунке 10.2, при выборе подпункта «Автоматы и полуавтоматы» открытие формы **Form3**, а при выборе подпункта «Выход» закрытие формы **Form2** .

Процедура для подпункта «Автоматы и полуавтоматы»:

«procedure TForm2.N2Click(Sender: TObject);
 begin
  Form3.showmodal;
 end;
Для подпункта «Выход»:
procedure TForm2.N5Click(Sender: TObject);
begin
  form2.Close;
end;».

### 10.1.2 Мастер форм баз данных

Для работы с таблицами баз данных при проектировании удобно использовать программу **Database Desktop**, которая позволяет выполнять следующие действия:

- создание таблицы;
- изменение структуры;
- редактирование записей;
- создание, редактирование и выполнение визуальных запросов и SQL – запросов.

Запуск утилиты осуществляется с помощью главного меню Windows **Пуск** => **Программы** => **Delphi** => **Database Desktop** или **Программы** => **Borland Delphi** => **Delphi** => **Tools** => **Database Desktop.**

Для создания таблицы базы данных нужно выбрать элемент главного меню программы **File** => **New** => **Table**. В появившемся окне **Create Table** выбирается тип создаваемой таблицы списка, а затем задается структура данных, то есть для каждого поля таблицы типа Paradox 7 задается имя **Field Name**, тип **Type**, размер **Size**, определяется первичный ключ **Key**.

После определения всех полей необходимо сохранить таблицу на диске, нажав кнопки **Save As** и задав в появившемся окне имя создаваемой таблицы.

Простым и быстрым способом создания форм для работы с базами данных является использование Мастера форм баз данных. С помощью мастера можно автоматически создавать формы от начала до конца, не помещая на форму какие-либо компоненты. Мастер является программой, которая на основе ряда заданных вопросов и предложенных вариантов ответов создаст основу, с которой можно начинать разработку сложной формы или проекта. Это может значительно сэкономить время.

Мастер форм берет на себя задачи соединения компонентов формы с компонентами таблиц и запросов, а также определяет последовательности активизации элементов управления.

Запуск мастера форм баз данных осуществляется выбором в главном меню Delphi команды **File => New => Other…**, в диалоговом окне **New Items** выбирается закладка **Business,** на которой запускается команда Мастера форм базы данных **DataBase Form Wizard**. На экран будет выведено диалоговое окно, представленное на рисунке 10.3.

На первом этапе Мастер предлагает выбрать вид и способ формирования таблицы. В разделе **Form Options** выбирается либо простая форма с одним набором данных **Create a single form,** либо форма «главная/детали» **Create a master/detail form** с зависимостью между таблицами «один-ко-многим».

В **DataSet Options** первая страница мастера форм позволяет выбирать следующие значения:

- **Create a form using objects** (создание формы с использованием объекта TTable , то есть пользователь явно указывает название используемой БД);
- **Create a form using TQuery objects** (создание формы, в которой обращение к БД будет осуществляться с помощью SQL запроса).

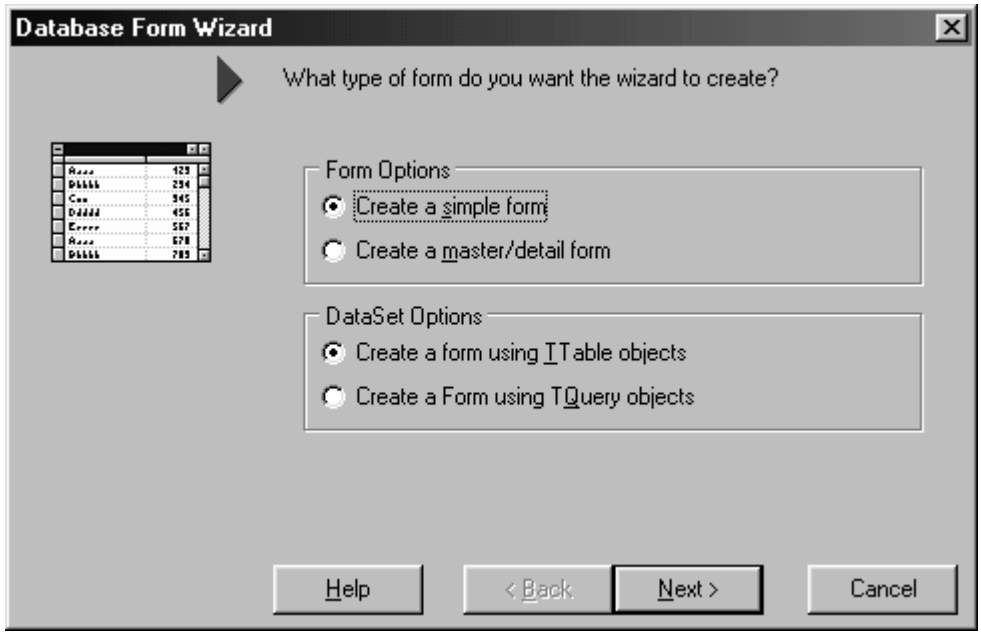



Рисунок 10.3 - Первое диалоговое окно Мастера форм баз данных

Для перехода к следующей странице необходимо щелкнуть на кнопке **Next>**. На следующем этапе пользователь выбирает имя таблицы, из которой будут получены данные (в случае создания связанных таблиц сначала выбирается имя главной таблицы).

В окне **Directories** определяется путь к файлу базы данных. Комбинированный список **Drive or Alias name** позволяет выбрать имя таблицы точно так же, как и при установке свойства **Database Name** компонента **DataSet** на этапе проектирования. Пример вида окна выбора имени таблицы представлен на рисунке 10.4. После выбора имени таблицы для перехода к следующей странице необходимо щелкнуть на кнопке **Next>**.

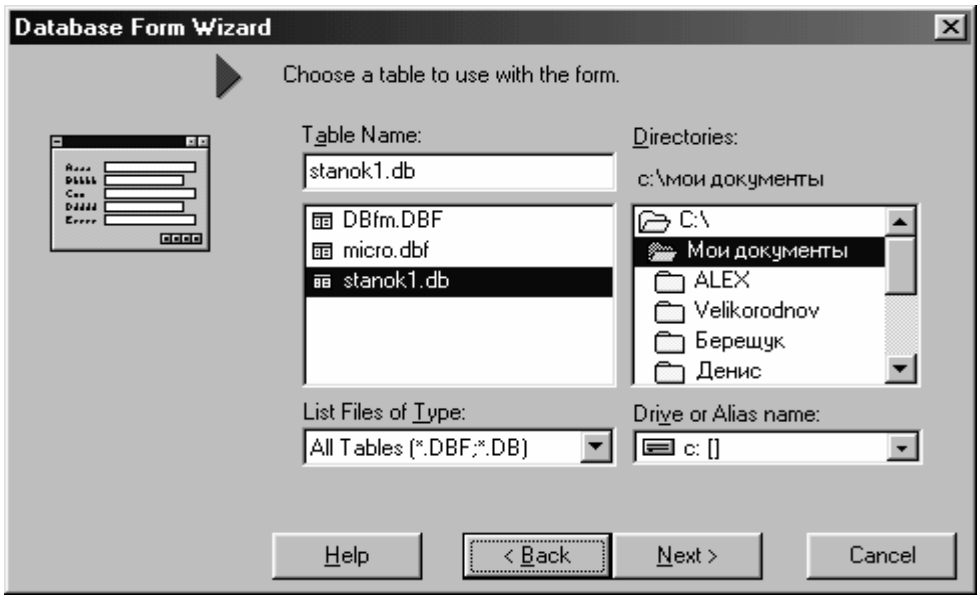



Рисунок 10.4 - Окно выбора имени таблицы

На третьей странице Мастера предлагается выбрать поля таблицы, которые должны быть включены в форму. Список **Available Fields** показывает поля, которые содержатся в выбранной таблице. Список **Ordered Selected Fields** содержит поля, которые надо отображать в создаваемой форме. Между двумя окнами списков расположены четыре кнопки для добавления или удаления полей. Чтобы добавить поле, необходимо щелкнуть на его имени в списке **Available Fields**, а затем на кнопке >. Чтобы добавить все поля за один раз, необходимо щелкнуть на кнопке >>. Удаление одного поля или группы выделенных полей осуществляется по кнопке <, всех полей за один раз <<. Включив нужные поля в список **Ordered Selected Fields**, можно изменить их порядок путем перетаскивания или щелкая на кнопках со стрелками под списком. На рисунке 10.5 показана третья страница мастера во время добавления полей. Для перехода к следующей странице нажать кнопку **Next>**.

На четвертой странице Мастера выбирается, как будут располагаться данные из каждого поля на форме. Возможны три варианта расположения: горизонтальное **Horizontally**, вертикальное **Vertically**, в виде таблицы **In a grid**. Выбор варианта расположения осуществляется нажатием соответствующей кнопки, при этом слева предлагается картинка с изображением.

На пятой странице выбирается, как будут размещены подписи к каждому полю относительно компонентов (Where do you want the labels pleased?). Метки можно разместить слева **Left** или сверху **Top** от компонентов.

На последней странице Мастера форм (рисунок 10.6) предлагается сделать два выбора:

- первый из них представлен флажком, расположенным в верхней части страницы, при установке флажка создаваемая форма должна быть главной формой приложения;

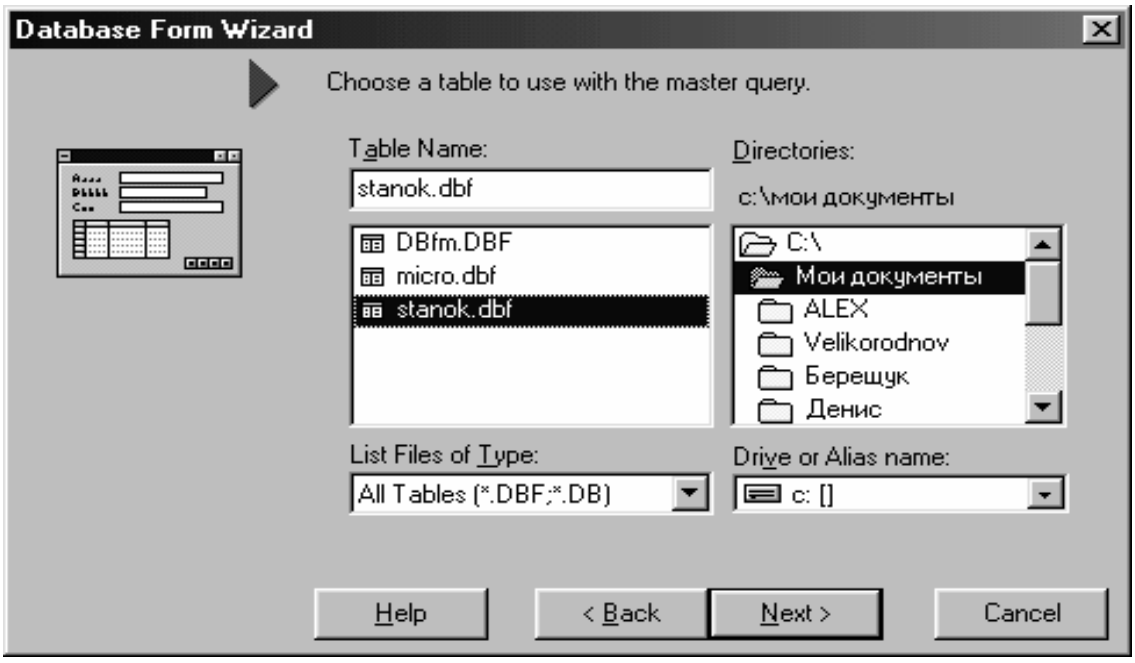



Рисунок 10.5 - Третья страница Мастера форм во время добавления полей

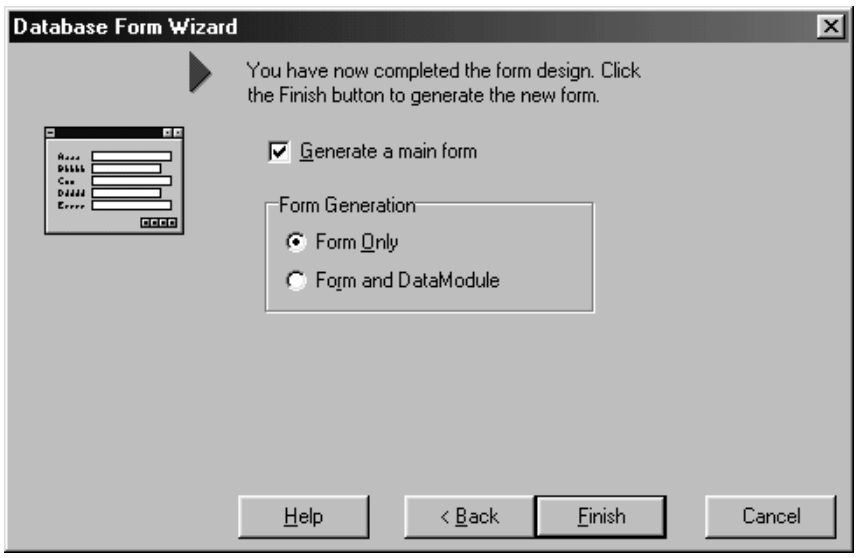



Рисунок 10.6 - Последняя страница Мастера форм баз данных

- второй **Form Generation** выбирает создание только формы или формы вместе с модулем данных; если выбрать первую опцию (**Form Only**), то форма будет содержать все компоненты данных, компонент **DataSet** (**Table** или **Query**) и компонент **DataSource**; если выбрать вторую опцию (**Form and DataModule**), то компоненты **DataSet** и **DataSource** будут удалены из формы и помещены в отдельный модуль данных.

Для создания формы необходимо щелкнуть на кнопке **Finish**. Завершенная форма будет выглядеть примерно так, как показано на рисунке 10.7. Форма содержит компоненты данных для каждого выбранного поля, а также метку для каждого компонента. В верхней части формы находится навигатор **DBNavigator** (  ), с помощью которого возможен переход от записи к записи.

Для удобства работы с формой можно изменить латинские названия столбцов на русские (как это выполнялось в предыдущей работе 9).

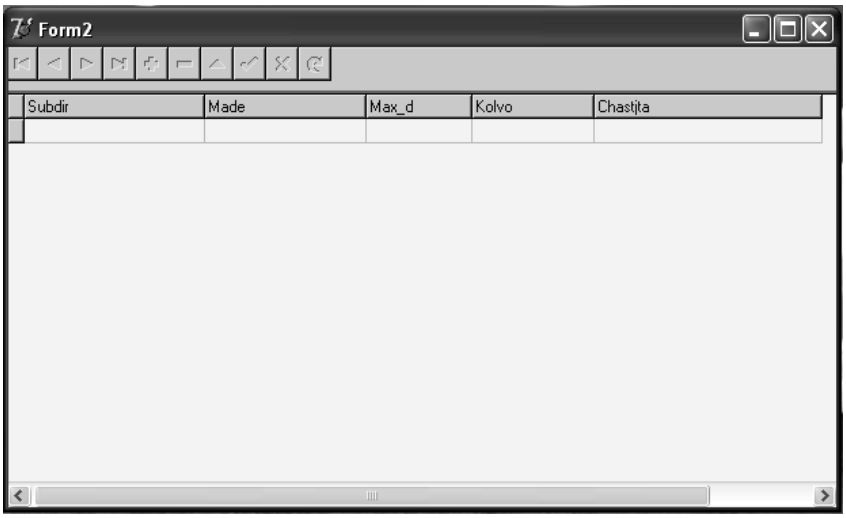



Рисунок 10.7 – Вид формы после запуска приложения

### 10.1.3 Создание кнопок управления для перемещения и редактирования

Перемещение по набору данных заключается в управлении указателем текущей записи (курсором). Этот указатель определяет запись, с которой будут выполняться такие операции, как редактирование или удаление.

Перед перемещением указателя текущей записи набор данных автоматически переводится в режим просмотра. Если текущая запись находилась в режиме редактирования или вставки, то перед перемещением указателя сделанные в записи изменения вступят в силу.

При создании формы с помощью Мастера формы баз данных вверху формы автоматически размещается навигатор, который представляет собой небольшую группу кнопок, связанных с одним набором данных. Навигатор **DBNavigator** (  ) является визуальным компонентом, расположенным на странице **Data Controls**. Каждая кнопка навигатора выполняет определенную функцию базы данных, описанную в таблице 10.1. Для связи навигатора с набором данных для свойства **DataSource** указывается соответствующее значение компонента **DataSource**.

Состав видимых кнопок определяет свойство **VisibleButtons**. Устанавливая для каждой из кнопок значение True или False в свойстве **VisibleButtons**, программист может формировать внешний вид навигатора. Подсказку для каждой из кнопок можно установить с помощью свойства **Hint** типа TString. Для отображения подсказок необходимо установить значение True для свойства **ShowHint**. Вызвав строковый редактор **String List Editor** нажатием на кнопку  в свойстве **Hint**, можно ввести русские названия соответствующих кнопок.

Таблица 10.1 – Кнопки навигатора базы данных

| Кнопка | Назначение | Метод |
|---|---|---|
|  | Переход на первую запись | First |
|  | Переход на предыдущую запись | Prior |
|  | Переход на следующую запись | Next |
|  | Переход на последнюю запись | Last |
|  | Вставка новой записи | Insert |
|  | Удаление текущей записи | Delete |
|  | Редактирование текущей записи | Edit |
|  | Утверждение результата изменения записи | Post |
|  | Отмена изменения в текущей записи | Cancel |
|  | Обновление информации в наборе данных | Refresh |

Реализация функций, выполняемых кнопками навигатора, возможна с помощью кнопок **Button**. В качестве примера рассмотрим создание двух групп кнопок «Управление» и «Редактирование».

Объединение группы связанных органов управления осуществляется визуальными компонентами **GroupBox**  (групповое окно), расположенными на странице **Standard**. Поместив на форме компонент **GroupBox**, необходимо убедиться, что на панели достаточно места для объединяемых компонентов. При недостатке места, активизировав компонент, можно изменить его размеры. Свойство **Caption** позволяет установить подпись. Так, пусть в расматриваемом примере на форму **Form3** необходимо поместить два компонента **GroupBox**, как показано на рисунке 10.8. Размеры их устанавливаются таким образом, чтобы внутри обрамления в виде прямоугольного участка на форме поместилось по пять компонентов **Button**, находящихся на странице **Standard**. Поскольку нажатие на каждую из кнопок должно приводить к стандартным процедурам навигатора базы данных, то в программном коде необходимо вызвать процедуры по присвоенному ей имени в соответствии с таблицей 10.1. Программные коды для кнопок управления имеют следующий вид:

```
«procedure TForm1.FirstButtonClick(Sender: TObject);
   begin
     Table1.First;
   end;
   procedure TForm1.NextButtonClick(Sender: TObject);
   begin
      Table1.Next;
   end;
   procedure TForm1.PriorButtonClick(Sender: TObject);
   begin
      Table1.Prior;
   end;
   procedure TForm1.LastButtonClick(Sender: TObject);
     begin
       Table1.Last;
   end;
   procedure TForm1.RecnoButtonClick(Sender: TObject);
    begin
     Table1.RecNo=StrToInt(Edit2.Text);
   end;».
```

Справа от кнопки **RecnoButton** («Перейти на»), по которой осуществляется переход на запись с указываемым номером, необходимо поместить визуальный компонент (однострочный редактор) **Edit**  , взятый со страницы **Data Controls**. Этот компонент используется для ввода и отображения текста. В Инспекторе свойств объекта задать свойству **Text** значение по умолчанию, равное «1».

Для кнопок редактирования вводятся следующие коды.

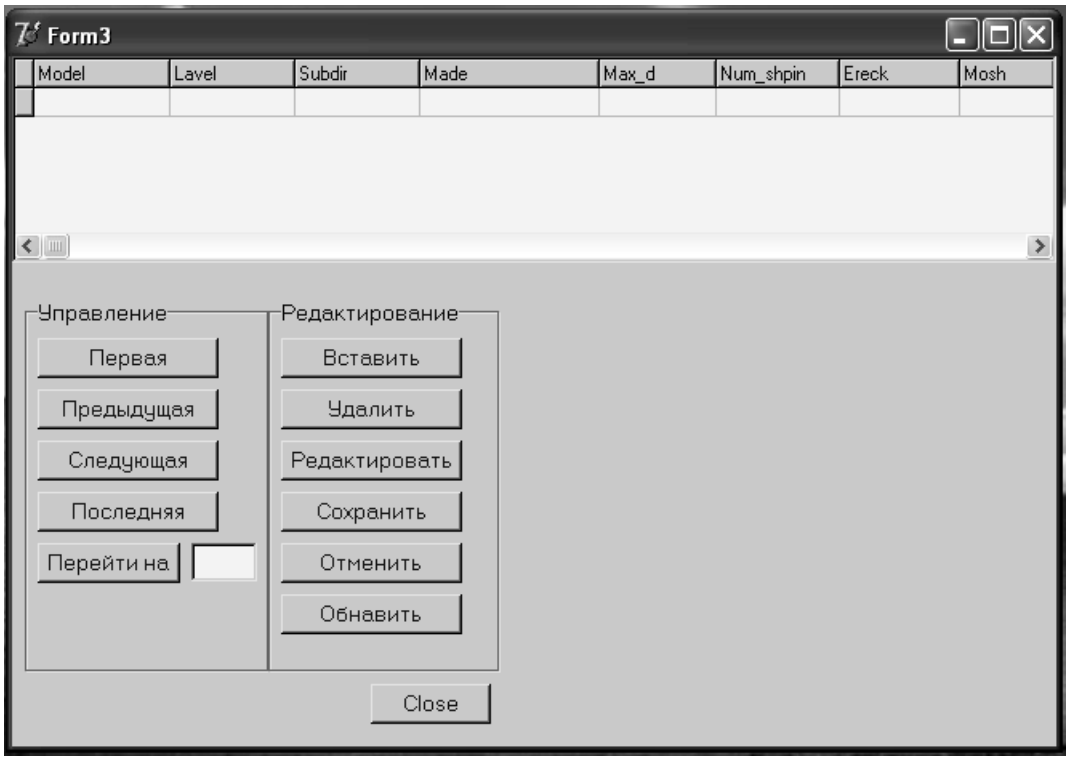



Рисунок 10.8 - Форма с кнопками

Для кнопки **Button6** «Вставить»:

«procedure Tform3.Button6Click(Sender: TObject);
begin
Table1.Insert;
end;»,

Для кнопки **Button7** «Удалить»:

«procedure Tform3.Button7Click(Sender: TObject);
begin
Table1.Delete;
end;».

Для кнопки **Button8** «Сохранить»:

«procedure Tform3.Button8Click(Sender: TObject);
begin
Table1.Post;
end;».

Для кнопки **Button9** «Отменить»:

«procedure Tform3.Button9Click(Sender: TObject);
begin
Table1.Cancel;
end;».

Для кнопки **Button10** «Обновить»:

«procedure Tform3.Button10Click(Sender: TObject);
begin
Table1.Refresh;
end;».

Для каждого компонента **Button** необходимо задать соответствующие значения в свойстве **Caption**.

### 10.1.4 Фильтрация записей

*Фильтрация* - это задание ограничений для записей, которые должны войти в набор данных. Состав записей в наборе данных в определенный момент времени зависит от устанавливаемых ограничений, в том числе и фильтрации. Delphi предоставляет возможность выполнения двух следующих вариантов фильтрации записей наборов данных:

- по выражению;
- по диапазону.

#### 10.1.4.1 Фильтрация по выражению

При фильтрации по выражению набор данных ограничивается записями, удовлетворяющими фильтру, задающему условия отбора записей. Поскольку в процессе отбора просматриваются все записи таблицы, фильтрация по выражению эффективна при небольшом количестве записей.

Для определения выражения фильтра используется свойство **Filter** типа String. Выражение фильтра представляет собой конструкцию, в состав которой могут входить следующие элементы:

- имена полей таблиц;
- литералы;
- операции сравнения;
- арифметические операции;
- логические операции;
- круглые и квадратные скобки.

Если имя поля содержит пробелы, то его заключают в квадратные скобки, в противном случае квадратные скобки необязательны. Литерал представляет собой значение, заданное явно (например, число, строка или символ).

Операции сравнения представляют собой обычные для языка Pascal отношения < , > , = , <= , >= и <> . Арифметическими являются операции + , - , * и / (сложение, вычитание, умножение, деление). В качестве логических операций можно использовать AND, OR и NOT(логическое умножение, сложение и отрицание соответственно). Круглые скобки применяются для изменения порядка выполнения арифметических и логических операций.

Для активизации и деактивации фильтра используется свойство **Filtered** типа Boolean. По умолчанию это свойство имеет значение False (фильтрация не происходит). При установке свойству **Filtered** значения True фильтрация включается, и в набор данных отбираются те данные, которые удовлетворяют фильтру, записанному в свойстве **Filter**. Если выражение фильтра не задано, то в набор данных попадают все записи.

Параметры фильтрации можно задать с помощью свойств **FilterOptions** типа TFilterOptions. Это свойство принадлежит к множественному типу и может принимать комбинации двух значений:

- **foCaseIntensitive** – регистр букв не учитывается. Например, при наличии этого значения для выражения фильтра Avtom='Автомат' в поле Avtom, например, могут содержаться слова «Автомат», «АВТОМАТ» или «автомат», которые будут восприняты как одинаковые.

- **foNoPartialCompare** – выполняется проверка на полное соответствие содержимого поля и значения, заданного для поиска. Обычно применяется для строк. Если известны только первые символы или символ строки, то нужно указать их в выражении фильтра, заменив остальные символы символом «*» и выключив значение **FoNoPartialCompare**.

По умолчанию все фильтры выключены, и свойство **FilterOptions** имеет значение, равное «[ ]».

### 10.1.4.2 Фильтрация по диапазону

При фильтрации по диапазону в набор данных будут включены те записи, значение полей которых соответствуют заданному диапазону. Таким образом, условием фильтрации является выражение вида «**значение > нижней границы AND значение < верхней границы**», в котором вместо операций сравнения < и > могут указываться операции <= и >=. Такая фильтрация применяется к наборам данных **Table**.

Достоинством фильтрации по диапазону является высокая скорость обработки записей. В отличие от фильтрации по выражению, когда последовательно просматриваются все записи таблицы, фильтрация по диапазону ведётся индексно-последовательным методом, поэтому этот способ фильтрации применим только для индексированных полей. Индекс поля, диапазон которого задан в качестве критерия для отбора записей, должен быть установлен как текущий с помощью свойства **IndexName** или **IndexFieldNames**.

В лабораторной работе индексированные поля не создавались, поэтому фильтрация по диапазону более подробно рассматриваться не будет.

### 10.1.4.3 Пример создания фильтрации по выражению

Рассмотрим в качестве примера обработчики событий формы, использующей фильтрацию записей набора данных по выражению. Вид формы приведён на рисунке 10.9.

Для задания выражений фильтра используются редакторы **Edit**. Компонент **Edit2** предназначен для задания выражения при фильтрации по подгруппе станка, компоненты **Edit3** и **Edit4** задают соответственно минимальную и максимальную границу значения поля «Максимальный диаметр заготовки».

На форму помещаются две кнопки **Button**. При нажатии кнопки **Button11** с заголовком «Фильтровать» фильтр активизируется путём присваивания значения True свойству **Filtered** набора данных. При активизации фильтра проис-

ходит отбор записей, которые удовлетворяют заданному в выражениях условию. При нажатии кнопки **Button12** с заголовком «Отменить» показываются все записи, фильтр при этом отключается.

Ниже приведены программные коды (обработчики событий) модуля формы **Form3**, предназначенные для фильтрации:

«procedure TForm3.Button11Click(Sender:TObject);
```
begin
if RadioButton1.Checked then
 begin
    Table1.FilterOptions:=[foCaseInsensitive];
    Table1.Filter:='Podgrup='"+edit2.text+'*"';
    Table1.Filtered:=True;
    beep;
  end;
if RadioButton2.Checked then
 begin
    Table1.Filter:='Max_d>='+edit3.text+'AND Max_d<='+edit4.text;
    Table1.Filtered:=True;
    beep;
  end;
end;
procedure TForm3.Button12Click(Sender:TObject);
begin
  Table1.Filtered:=false;
end;».
```

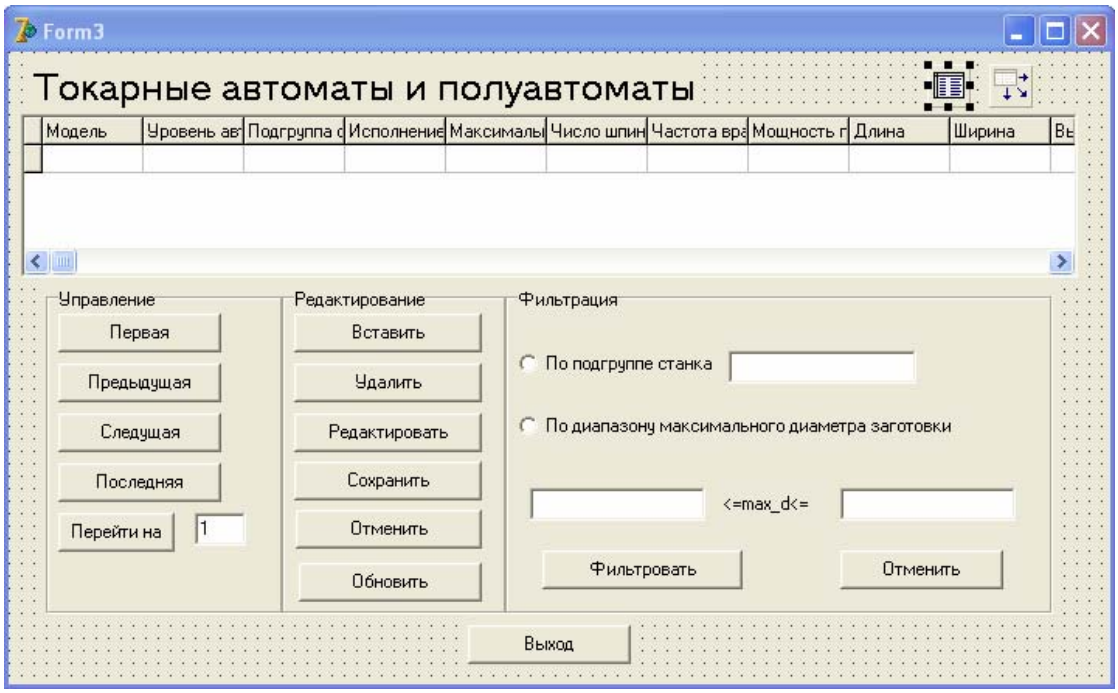



Рисунок 10.9 - Форма «Токарные автоматы и полуавтоматы» с фильтрацией по выражению

### 10.1.5 Редактор полей

По умолчанию для каждого физического поля при открытии набора данных создается автоматически объект типа **TField**, и в наборе данных доступны все поля. Эти поля являются динамическими. Для создания статических полей используется специальный редактор полей. В случае, если хотя бы одно поле набора данных является статическим, то динамические поля создаваться не будут. Таким образом, в наборе данных будут доступны только статические поля, а все остальные считаются отсутствующими. Определить или отменить состав статических полей можно с помощью редактора полей на этапе разработки приложения.

Для запуска редактора полей (рисунок 10.10) следует дважды щелкнуть на компоненте **Table** или вызвать для этого компонента контекстное меню и выбрать пункт **Field Editor...** (Редактор полей).


Рисунок 10.10 - Редактор полей

В заголовке редактора полей выводится составное имя набора данных пример, **Form2.Table1**. Для перемещения по полям используются четыре кнопки редактора или мышь. Большую часть редактора занимает список статических полей, при этом поля перечисляются в порядке их создания, этот порядок может отличаться от порядка полей в таблице БД.

Первоначально список статических полей пуст. Это означает, что поля набора данных являются динамическими. С помощью редактора разработчик может:

- создать новое статическое поле;
- удалить статическое поле;
- изменить порядок статических полей;
- создать новое поле.

Кроме того, для любого выбранного в редакторе статического поля **Инспектора объектов** возможно задание или изменение свойств поля и определение обработчиков его событий. Подобные действия оказываются возможными в

связи с тем, что существующим статическим полям объекты типа **TField** доступны на этапе разработки приложения.

Для создания статического поля следует вызвать контекстное меню редактора полей и выбрать пункт **Add Fields** (Добавить поля), в результате появляется диалоговое окно добавления новых полей (рисунок 10.11). В окне **Aviable fields** (Доступные поля) содержатся все те поля набора данных, которые еще не являются статическими. После выбора одного или нескольких полей и нажатия кнопки **ОК** эти поля добавляются в состав полей набора данных. Добавленное статическое поле является полем данных и связано с конкретным физическим полем таблицы БД.

Для добавления в список всех физических полей таблицы (для набора данных Table) нужно выбрать в контекстном меню редактора полей пункт **Add all Fields** (Добавить все поля).

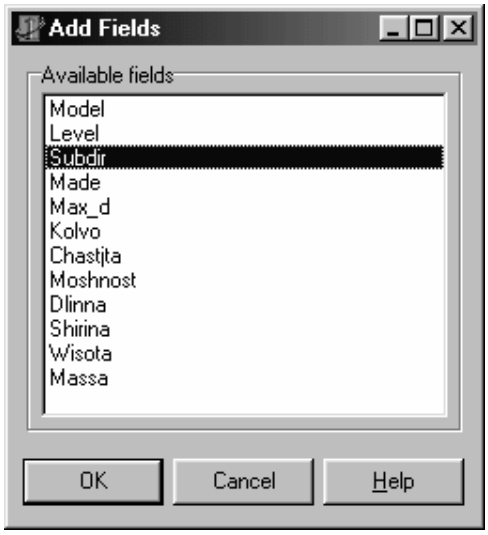



Рисунок 10.11 - Добавление новых статических полей

При каждом открытии (в том числе при разработке приложения) набора данных выполняется проверка возможности создания экземпляров класса **TField** для статических полей. Если поле вследствие какой-нибудь ошибки является недопустимым и создание объекта типа **TField** невозможно, то генерируется исключительная ситуация, а набор данных закрывается.

Для удаления статического поля нужно выбрать пункт **Delete** (Удалить) контекстного меню или выделить в списке поле и нажать клавишу **Delete**. После удаления статического поля оно становится недоступным для операций в программе, однако в случае необходимости это поле можно снова сделать статическим, добавив его в список редактора полей. При этом следует иметь в виду, что все свойства этого поля устанавливаются заново, а все сделанные ранее изменения теряются.

Перечень, в котором поля перечисляются в списке редактора, задает порядок этих полей. По умолчанию порядок полей соответствует порядку физических полей в таблицах БД. Можно изменить порядок полей, перемещая их в

списке с помощью мыши или комбинаций клавиш **Ctrl+Page Up** и **Ctrl+Page Down**.

Можно определить три типа статических полей:

- поле данных, заменяющее соответствующее физическое поле таблицы;
- вычисляемое поле, значение которого рассчитывается в обработчике события **onCalcFields** во время выполнения приложения;
- поле выбора, значение которого можно выбирать из списка, формируемого на основе заданных критериев и правил.

Для создания нового поля нужно выбрать в контекстном меню редактора полей пункт **New Field...** (Новое поле). В результате появляется диалоговое окно **New Field** создания нового статического поля (рисунок 10.12).

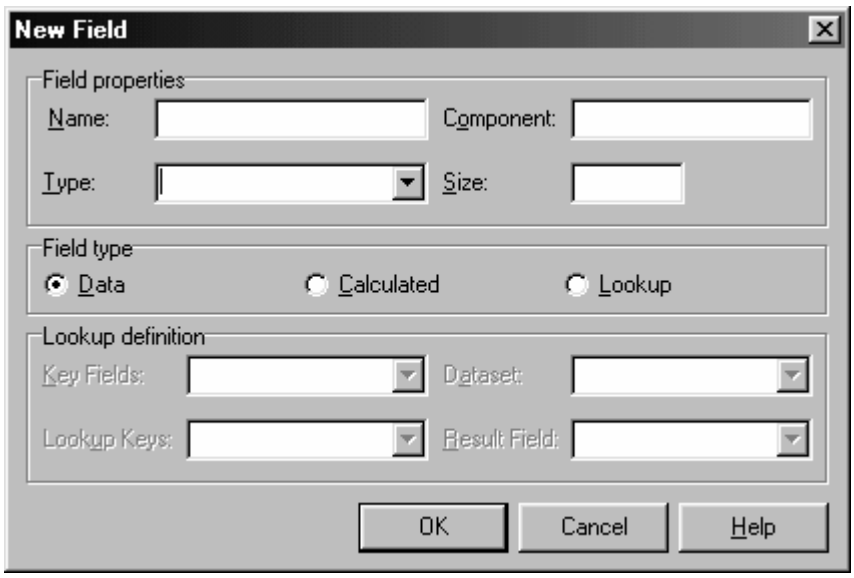



Рисунок 10.12 - Окно создания статического поля

Для задания общих свойств (параметров) нового поля используется группа управляющих элементов **Field properties** (Свойства поля). В поле **Name** (Имя) задается значение свойства **FieldName**, то есть имя поля, а в поле **Component** – значение свойства Name, то есть название компонента (объекта типа **TField**). При программировании обычно используется имя поля. Значение поля **Component** формируется Delphi автоматически, и попытка изменить его не приводит к желаемому результату. В полях **Type** и **Size** указывается тип и размер поля. Тип должен быть задан в обязательном порядке, необходимость задания размера поля зависит от типа.

Тип нового поля выбирается с помощью группы переключателей **Field type** (Тип поля) из следующих вариантов:

- **Data** (поле данных);
- **Calculated** (вычисляемое поле);
- **Lookup** (поле выбора).

В группе **Lookup definition** для поля выбора устанавливаются такие параметры, как набор данных и поля связи, а также поля для формирования списка выбора и результата.

Для создания вычисляемого поля нужно выполнить следующее:

- в окне создания нового поля задать имя и тип поля, а также выбрать переключатель **Calculated;**

- для набора данных, содержащего поле, подготовить код обработчика события **OnCalcFields** типа TDataSetNotifyEvent, в котором этому полю присваивается требуемое значение. Для расчета значения вычисляемого поля можно использовать значения других полей, а также переменные и константы программы.

Событие **onCalcFields** предназначено для определения значений всех вычисляемых полей набора данных. Оно генерируется при считывании записи из таблицы, а также при изменении значений невычисляемых полей, если свойству **AutoCalcFields** типа Boolean установлено значение по умолчанию. На время выполнения обработчика события **onCalcFields** набор данных переводится в режим расчета вычисляемых полей, а затем возвращается в предыдущий режим.

Например, для создания вычисляемого поля «Площадь» Ploshad, равного произведению длины на ширину, выбирается компонент **Table1**. В инспекторе объектов на вкладке **Events** два раза щелкнуть мышкой на компоненте списка событий **onCalcFields** и ввести следующий программный код:

«procedure TForm3.Table1CalcFields(DataSet: TDataSet);
begin
Table1Ploshad.AsInteger:=Table1.fieldbyname('dlina').asinteger*Table1.field byname('shirina').asinteger;
end;».

Пусть необходимо ограничить диапазон вводимых значений для поля «Число шпинделей» Number_shin, число шпинделей станка может принимать значение от 1 до 8. Для этого активизировать компонент **Table1** и вызвать редактор полей. Выделить статический столбец Number_shin, и в Инспекторе объектов установить свойству **MaxValue** значение, равное «8», а свойству **MinValue** - значение «1».

### 10.1.6 Сетка

Для вывода содержимого набора данных в табличном виде удобно использовать сетку, представленную в Delphi компонентом **DBGrid**. Внешний вид сетки соответствует внутренней структуре таблицы БД и набора данных, при этом строке сетки соответствует запись, а столбцу - поле.

С помощью сетки пользователь управляет набором данных, поля которого отображаются в ней. Для перемещения по записям и их просмотра используются полосы прокрутки и клавиши управления курсором. Для перехода в режим редактирования поля текущей записи достаточно нажать какую-либо из алфавитно-цифровых клавиш, когда курсор установлен на это поле. Переход в режим вставки новой записи выполняется при нажатии клавиши **Insert**. Вставка записи появляется в том месте, где находится указатель текущей записи. Изменения, сделанные при редактировании или добавлении записи, можно при-

нять, нажав клавишу **Enter** или перейдя к другой записи; отменить изменение, нажав клавишу **Esc**. Для удаления записи следует нажать клавиши **Ctrl+Delete**.

Отдельный столбец **Column** сетки представляет собой объект типа Tcolumn. По умолчанию для каждого поля набора данных, связанного с компонентом **DBGrid**, автоматически создается отдельный столбец и в сетке доступны все столбцы. Такие столбцы являются динамическими. Для создания статических столбцов используется специальный редактор столбцов. Если хотя бы один столбец сетки является статическим, то динамические столбцы уже не создаются ни для одного другого поля набора данных. Причем в наборе данных доступны статические столбцы, а остальные столбцы считаются отсутствующими. Определить или отменить состав статических столбцов можно с помощью Редактора столбцов на этапе разработки приложения.

Взаимодействие между динамическими и статическими столбцами, а также с Редактором столбцов аналогично взаимодействию между динамическими и статическими полями набора данных и Редактором полей.

Характеристики и поведение сетки и ее отдельных столбцов во многом определяются полями набора данных (а также соответствующими объектами типа **TField**), для которых создаются объекты **TColumn**.

Функционирование динамических столбцов зависит от свойств объекта поля - при изменении свойств объекта типа **TField** изменяются свойства объекта типа **TColumn**. К примеру, динамический столбец получает от поля название и ширину.

К достоинствам статических столбцов можно отнести то, что для их объектов имеется возможность установить значения свойств, отличные от свойств соответствующего поля. Например, если для некоторого статического столбца установить свое название, то оно не будет меняться даже в том случае, если с этим столбцом связывается другое поле набора данных. Кроме того, объекты типа **TColumn** статических столбцов создаются при разработке приложения и их свойства доступны через Инспектор объектов.

Чтобы запустить Редактор столбцов (рисунок 10.13), нужно вызвать контекстное меню компонента **DBGrid** и выбрать в нем пункт **Columns Editor...** (Редактор столбцов). Редактор столбцов можно вызвать также через значение свойства **Columns** в Инспекторе объектов. В заголовке редактора столбцов выводится составное имя массива столбцов, например, **DBGrid1.Columns**. Большую часть редактора занимает список статических столбцов, при этом столбцы перечисляются в порядке их создания, этот порядок может отличаться от порядка полей в наборе данных.

Первоначально список статических столбцов пуст. Это означает, что столбцы сетки являются динамическими. С помощью редактора столбцов можно выполнить следующее:

- создать новый статический столбец;
- удалить статический столбец;
- изменить порядок статических столбцов.

Работа с редактором столбцов мало, чем отличается от работы с рассмотренным выше редактором полей.

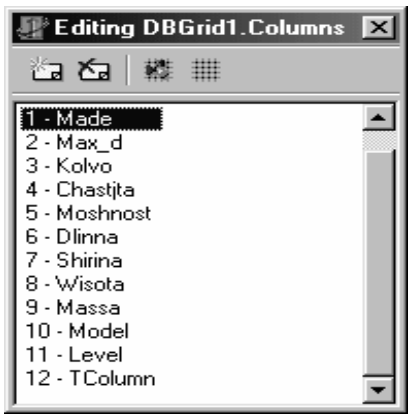



Рисунок 10.13 - Окно Редактора столбцов

При изменении порядка столбцов сетки автоматически изменяется порядок связанных с ними полей набора данных, что необходимо учитывать при доступе к полям набора данных по номерам объектов типа TField, а не по именам этих объектов.

После создания статических столбцов с ними можно работать с помощью свойства **Columns** типа TDBGridColumns. При проектировании приложений свойство **Columns** доступно через Инспектор объектов.

Основные свойства объекта столбца:

- **Alignment** типа TAlignment управляет выравниванием значений в ячейках столбца;

- **PickList** типа TStrings представляет собой список для выбора заносимых в поле значений. Текущая ячейка вместе со списком PickList образуют своего рода компонент **ComboBox**. Если для столбца сформирован список выбора, то при попытке редактирования ячейки этого столбца справа появляется стрелка, при нажатии которой раскрывается список, в котором можно выбрать один из элементов. При этом можно ввести в ячейку любое допустимое значение.

- **Title** типа TColumnTitle представляет собой объект заголовка столбца. Этот объект, в свою очередь, имеет такие свойства, как **Caption, Alignment, Color** и **Font**, определяющие название, выравнивание, цвет и шрифт заголовка, соответственно.

Свойства столбца, управляющие форматированием, видимостью или возможностью модификации значений не отличаются от соответствующих свойств поля.

Рассмотрим пример создания поля выбора. Пусть поле «Уровень автоматизации» (стоящее в Редакторе столбцов под номером 0) может принимать значение «Автомат» и «Полуавтомат», поле «Подгруппа станка» (1) – «Одношпиндельные» и «Многошпиндельные», поле «Исполнение» (9) – «Продольного точения», «Горизонтальный», «Вертикальный».

Для создания поля создать статические поля для всех столбцов БД (с помощью Редактора столбцов). Активизировать щелчком мыши компонент **Form3**. В Инспекторе объектов на вкладке **Events** два раза щелкнуть мышкой

на компоненте списка событий **onCreate** и ввести следующий программный код:

```
«procedure TForm3.FormCreate(Sender: TObject);
    begin
    DBGrid1.Columns[0].picklist.add('автомат');
    DBGrid1.Columns[0].picklist.add('полуавтомат');
    DBGrid1.Columns[9].picklist.add('продольного точения');
    DBGrid1.Columns[9].picklist.add('горизонтальный');
    DBGrid1.Columns[9].picklist.add('вертикальный');
    DBGrid1.Columns[1].picklist.add('одношпиндельные');
    DBGrid1.Columns[1].picklist.add('многошпиндельные');
    end;».
```

При работе приложения с **Form3** редактор столбцов обращается к полю с указанным номером, поэтому необходимо при формировании программного кода следить, чтобы было совпадение номеров.

### 10.2 Задание на выполнение работы

10.2.1. Создать на диске D:\ в папке с именем группы свой каталог, в котором будут храниться все файлы, относящиеся к данной лабораторной работе. При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его в своём каталоге.

10.2.2 Запустить утилиту **Database Desktop**, установить умалчиваемый псевдоним.

10.2.3 Задать структуру таблицы «Токарные автоматы и полуавтоматы», содержание которой приведено в таблице 10.2, сохранить в соответствующем файле. Использовать для таблицы тип Paradox 7.

10.2.4 Создать форму **Form1** «Выбор группы станков» в соответствии с рисунком 10.1, при этом вариант «Токарные» должен приводить к дальнейшей работе приложения. При выборе вариантов «Сверлильные», «Шлифовальные» или «Фрезерные» должно появляться сообщение «Данная группа станков недоступна». При варианте «Выход» форма должна закрываться.

10.2.5 Создать форму **Form2** «Выбор типа станка» в соответствии с рисунком 10.2. Активными сделать только объекты «Автоматы и полуавтоматы» и «Выход».

10.2.6 С помощью **Мастера форм** создать экранную форму **Form3** для таблицы «Токарные автоматы и полуавтоматы». Для компонента **Table1** свойству **Active** задать значение True, тем самым сделать таблицу активной. Запустить приложение. Заполнить несколькими записями базу данных. Посмотреть работу навигатора.

10.2.7 Закрыть приложение. Удалить все компоненты с формы **Form3**, кроме **Table1** и **DataSource1**. Разместить на форме компонент **DBGrid**1, со страницы **Data Controls**, его свойству **DataSource** задать значение DataSource1. Заменить латинские названия полей на русские. Используя компонент **Label**, поместить на форму заголовок базы данных.

Таблица 10.2 – Токарные автоматы и полуавтоматы

| Модель | Уровень автоматизации | Подгруппа станка | Исполнение | Максимальный диаметр заготовки. мм | Число шпинделей | Частота вращения, об./мин | Мощность главного привода, кВТ | Длина, мм | Ширина, мм | Высота, мм | Масса, кг |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1103А | Автомат | Одношпиндельный | Проточного точения | 4 | 1 | 1600 - 12500 | 1 | 1050 | 690 | 1345 | 400 |
| 1Б10В | Автомат | Одношпиндельный | Проточного точения | 6 | 1 | 1400 - 10000 | 1,5 | 1250 | 810 | 1430 | 630 |
| 11Т16В | Автомат | Одношпиндельный | Проточного точения | 16 | 1 | 450 - 6300 | 3,0 | 1900 | 945 | 1520 | 1200 |
| 1М32В | Автомат | Одношпиндельный | Проточного точения | 32 | 1 | 280 - 3550 | 3,1 | 2360 | 1150 | 1630 | 1700 |
| 1216-4К | Автомат | Многошпиндельный | Горизонтальный | 20 | 4 | 279 - 1995 | 7,5 | 5385 | 1000 | 1520 | 4000 |
| 1216-6К | Автомат | Многошпиндельный | Горизонтальный | 16 | 6 | 370 - 2650 | 7,5 | 5385 | 1000 | 1520 | 4000 |
| 1Б240П-4К | Полуавтомат | Многошпиндельный | Горизонтальный | 160 | 4 | 63 - 1048 | 13 | 4330 | 1600 | 1985 | 9000 |
| 1Б225П-6К | Полуавтомат | Многошпиндельный | Горизонтальный | 100 | 8 | 120 - 1700 | 15 | 4105 | 1320 | 1920 | 5800 |
| 1А286-6 | Полуавтомат | Многошпиндельный | Вертикальный | 630 | 6 | 125 - 250 | 10 | 4790 | 4790 | 4925 | 35500 |

10.2.8 Создать на форме **Form3** кнопки управления для перемещения по записям и их редактирования в соответствии с рисунком 10.8. Группа «Управление» должна включать кнопки «Первая», «Предыдущая», «Следующая», «Последняя», «Перейти на», группа «Редактирование» – «Восстановить», «Удалить», «Сохранить», «Отменить», «Обновить».

10.2.9 Создать на форме **Form3** фильтры по выражению «По типу станка» и «По диапазону максимального диаметра заготовки», как это показано на рисунке 10.9. Поместить две кнопки **Button**: одну для выполнения фильтрации, вторую - для её отмены.

10.2.10 Создать вычисляемое поле «Площадь».

10.2.11 Ввести ограничение на значение поля «Число шпинделей», которое может принимать значения от 1 до 8.

10.2.12 Создать поля выбора «Уровень автоматизации», «Подгруппа станка», «Исполнение» с помощью редактора столбцов. Значения для полей указаны в таблице 10.2.

10.2.13 Запустить созданное приложение. Протестировать его работу, вводя остальные записи из таблицы 10.2.

10.2.14 Оформить отчет по лабораторной работе.

### 10.3 Содержание отчёта

10.3.1 Название работы.

10.3.2 Цель работы.

10.3.3 Структура таблицы «Токарные автоматы и полуавтоматы».

10.3.4 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

10.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

### 10.4 Тесты и контрольные вопросы

10.4.1 Визуальный компонент **RadioButton** расположен на странице:

а) **Data Controls**; б) **Data Access**; в) **Standard**;
г) **Additional**; д) **BDE**.

10.4.2 Для чего предназначен невизуальный компонент **MainMenu**:

а) для создания главного и раскрывающегося меню;
б) для создания главного меню;
в) для создания раскрывающегося меню;
г) для создания меню в виде списка;
д) для создания меню в виде кнопок?

10.4.3 Какому свойству редактируемого пункта компонента **MainMenu** необходимо установить в Инспекторе объектов значение False, чтобы сделать пункт недоступным:

а) **Caption**; б) **Enabled**; в) **Default**;
г) **Checked**; д) **RadioItem**?

10.4.4 С какой закладки в диалоговом окне **New Items** запускается команда Мастера форм базы данных **DataBase Form Wizard**:

а) **Business**;

б) **Forms**;

в) **Projects**;

г) **Dialogs**;

д) **New**?

10.4.5 Для какого свойства навигатора **DBNavigator** необходимо установить значение True, чтобы отображались подсказки:

а) **TabStop**; б) **ShowHint**; в) **ParentShowHint**;

г) **Enabled**; д) **Flat**?

10.4.6 Компонент **GroupBox** предназначен:

а) для пересечения группы связанных элементов;

б) для объединения группы несвязанных элементов;

в) для суммирования группы связанных элементов;

г) для суммирования группы несвязанных элементов;

д) для объединения группы связанных элементов.

10.4.7 Какое действие выполняется при нажатии на кнопку со следующим программным кодом:

```
procedure TForm1.FirstButtonClick(Sender: TObject);
begin
  Table1.First;
end;
```

а) переход на первую запись;

б) переход на последнюю запись;

в) переход на следующую запись;

г) переход на предыдущую запись;

д) удаление первой записи?

10.4.8 Какой компонент используется для ввода и отображения текста:

а) **DataSource**; б) **Label**; в) **Button**;

г) **DBEdit**; д) **DBGrid**?

10.4.9 Что является достоинством фильтрации по диапазону:

а) низкая скорость обработки записей;

б) малый объем записей;

в) высокая скорость обработки записей;

г) большой объем записей?

10.4.10 К типам статических полей относятся:

а) поле данных, вычисляемое поле, поле выбора;

б) поле данных, суммируемое поле, поле выбора;

в) поле данных, вычисляемое поле, поле списка;

г) поле записей, вычисляемое поле, поле выбора;

д) поле записей, суммируемое поле, поле выбора.

10.4.11 Назовите достоинства системы визуального программирования Delphi.

10.4.12 Какие операции выполняются при создании приложений?

10.4.13 Для каких целей предназначен визуальный компонент Label?

10.4.14 Какие компоненты используются для создания списка вариантов, из которых необходимо выбрать только один?

10.4.15 Какое свойство в окне инспектора объектов для визуальных компонентов **RadioButton**, **Button** необходимо установить для задания надписи?

10.4.16 Прокомментируйте программный код кнопки «Далее» в форме Form1.

10.4.17 Какие этапы выполняются при создании главного и выпадающих меню с помощью конструктора меню?

10.4.18 Как осуществляются деактивация пунктов меню?

10.4.19 Прокомментируйте программные коды, которые нужно ввести для работы формы Form2.

10.4.20 Какие этапы выполняются при формировании формы командой **Database Form Wizard**?

10.4.21 Что представляет собой Мастер форм базы данных?

10.4.22 Каким образом можно изменить названия столбцов на русские наименования?

10.4.23 Для каких целей предназначен компонент **DBNavigator**?

10.4.24 Какие кнопки включает навигатор базы данных?

10.4.25 Каким визуальным компонентом осуществляется объединение группы связанных органов управления?

10.4.26 Прокомментируйте программные коды, которые задаются для компонентов **Button,** объединённых в группы «Управление» и «Редактирование».

10.4.27 Для каких целей используется компонент **Edit**, помещённый внутри компонента **GroupBox** «Управление»?

10.4.28 Что такое «фильтрация записей»?

10.4.29 Какие возможности представляет Delphi по фильтрации записей?

10.4.30 Какое свойство используется для определения выражения фильтра?

10.4.31 Что нужно включить в программный код, чтобы задать список для выбора значений поля?

10.4.32 Какие элементы входят в выражение фильтра?

10.4.33 Какое свойство используется для активации и деактивации фильтра?

10.4.34 Что задаётся с помощью свойств **FilterOptions**?

10.4.35 Что такое «статическое поле»?

10.4.36 Для чего используется редактор полей?

10.4.37 Какие типы статических полей можно определить?

10.4.38 Как создать вычисляемое поле?

10.4.39 Как ввести ограничения на значение поля?

10.4.40 Что такое сетка?

10.4.41 Для чего используется Редактор столбцов?

# 11 Создание диаграмм и отчетов при работе с базой данных в Delphi

Целью работы является создание в Delphi пользовательского приложения по работе с базой данных, которое включает формы с таблицей базы данных, диаграммами и отчетом, содержит кнопки для перехода между формами.

## 11.1 Общие положения

### 11.1.1 Мастер диаграмм

Одной из наглядных форм представления информации является представление их в виде диаграмм. Система программирования Delphi позволяет создавать диаграммы с помощью Мастера диаграмм **TeeChart Wizard** и с помощью компонента **TeeChart**.

Мастер является программой, устанавливаемой вместе с Delphi, которая на основе ряда заданных вопросов и предложенных вариантов ответов создает нужный объект. Запуск Мастера диаграмм осуществляется выбором в главном меню Delphi команды **File => New => Other…**, в диалоговом окне **New Items** выбирается вкладка **Business,** на которой запускается команда с пиктограммой компонента **TeeChart Wizard**  . На экран будет выведено диалоговое окно, представленное на рисунке 11.1.

Мастер предлагает выбрать стиль диаграммы. Есть два стиля диаграмм – связанные с базой данных и несвязанные. В лабораторной работе 11 необходимо в разделе **Select a Chart style** выбрать значение **Database Chart,** поскольку данные берутся из базы данных. Затем нажать кнопку **Next>** для перехода к следующей странице.

В открывшемся диалоговом окне **Select a Database Table** пользователь выбирает имя таблицы, из которой будут получены данные. В окне **Directories** определяется путь к файлу базы данных. Комбинированный список **Drive or Alias name** позволяет выбрать имя таблицы точно так же, как и при установке свойства **Database Name** компонента **DataSet** на этапе проектирования. Пример вида окна выбора имени таблицы представлен на рисунке 11.2. Для перехода к следующей странице необходимо щелкнуть на кнопке **Next>**.

На третьей странице мастера предлагается выбрать поля таблицы, которые должны будут использоваться в диаграмме. Список **Available Fields** показывает поля, которые являются доступными в выбранной таблице. Список **Selected Fields** содержит поля, которые являются выбранными для создаваемого графика. Между двумя окнами списков расположены четыре кнопки для добавления или удаления полей. Чтобы добавить поле, необходимо щелкнуть на его имени в списке **Available Fields**, а затем на кнопке >. Чтобы добавить все поля за один раз, необходимо щелкнуть на кнопке >>. Удаление одного поля или группы выделенных полей осуществляется по кнопке <, всех полей за один раз - <<. Включив нужные поля в список **Selected Fields**, можно изменить их

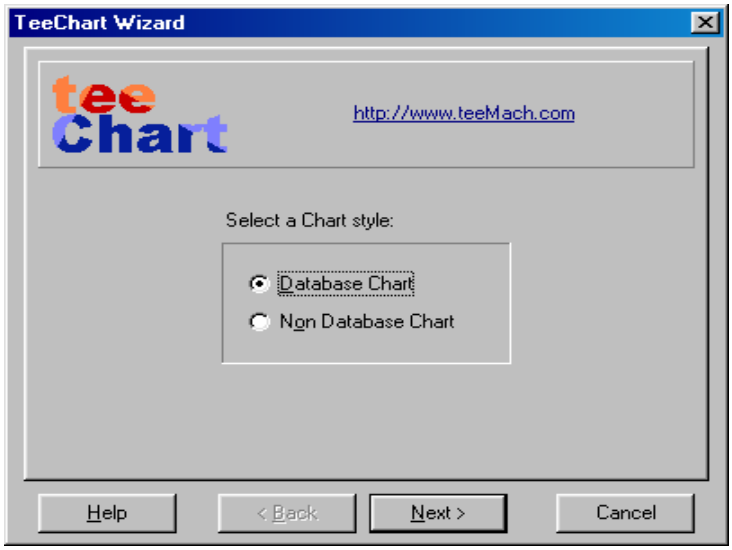



Рисунок 11.1 - Первое диалоговое окно Мастера диаграмм

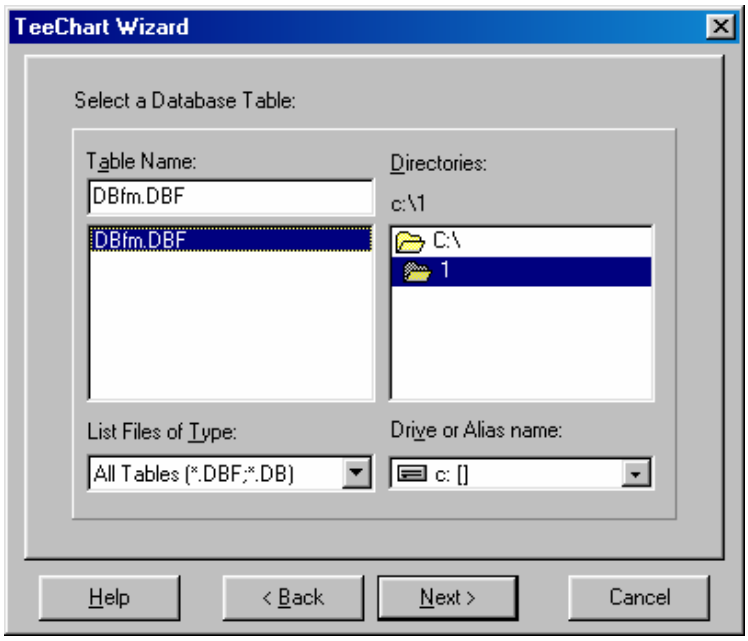



Рисунок 11.2 - Окно выбора имени таблицы

порядок путем перетаскивания или, щелкая на кнопках со стрелками под списком. В лабораторной работе следует выбрать одно поле, по которому будет создаваться круговая диаграмма. Например, поле Prase. В поле **Select a text labels Fields** пользователь должен выбрать то поле, которое будет использоваться в качестве пояснительного текста. На рисунке 11.3 показана третья страница мастера во время добавления полей. Для перехода к следующей странице необходимо нажать кнопку **Next>**.

В следующем окне предлагается выбрать тип диаграммы, а также способ отображения. В лабораторной работе рекомендуется активизировать пиктограмму с круговой диаграммой (Pie) и поставить переключатель у трехмерного

изображения (3D), как показано на рисунке 11.4. Для перехода к следующей странице нажать кнопку **Next>**.

На последней странице мастера диаграмм выполняется предварительный просмотр полученной диаграммы. В случае необходимости пользователь может вернуться на предыдущий шаг или согласиться. Здесь также возможно задание режимов показа:

- легенды **Show Legend** (области диаграммы, в которой приводится поясняющая информация);
- марок **Show Marks** (меток со значениями соответствующих секторов).

Для размещения диаграммы на форме необходимо щелкнуть на кнопке **Finish**.

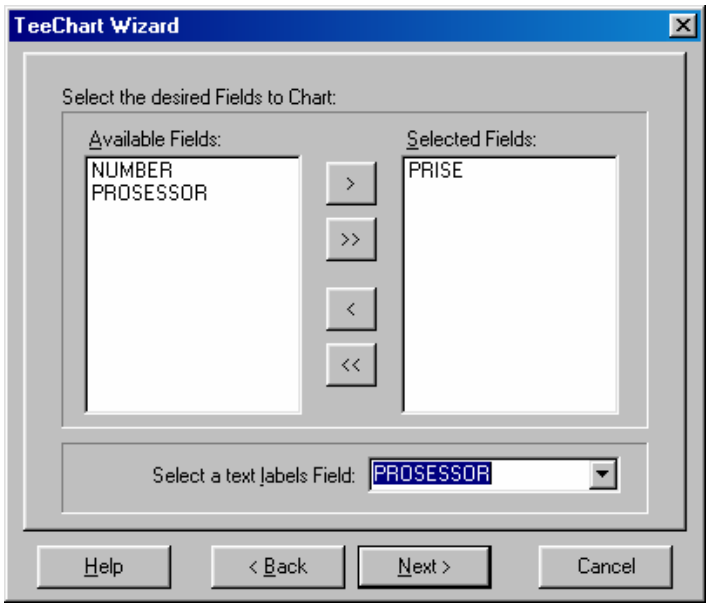



Рисунок 11.3 - Третья страница мастера во время выбора полей

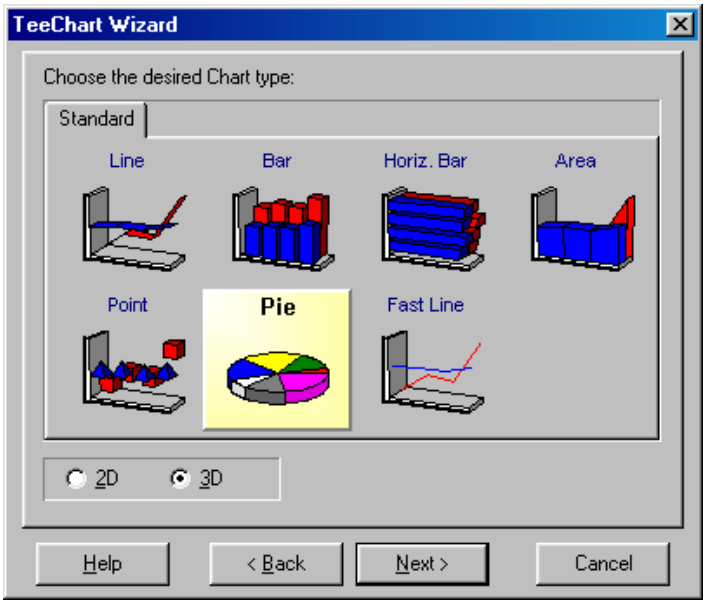



Рисунок 11.4 - Выбор типа диаграммы

### 11.1.2 Компонент DBChart

Для построения диаграмм на основании информации, содержащейся в наборе данных, предназначен компонент - диаграмма **DBChart**  . Он позволяет выводить диаграммы различных типов, в том числе объемные. Этот компонент является довольно сложным и имеет большое количество разнообразных свойств, многие из которых являются объектами и также имеют свои свойства. Этот компонент расположен на странице **Data Controls** палитры компонентов Delphi. После помещения на форму компонента в ней будет создана заготовка, как показано на рисунке 11.5.

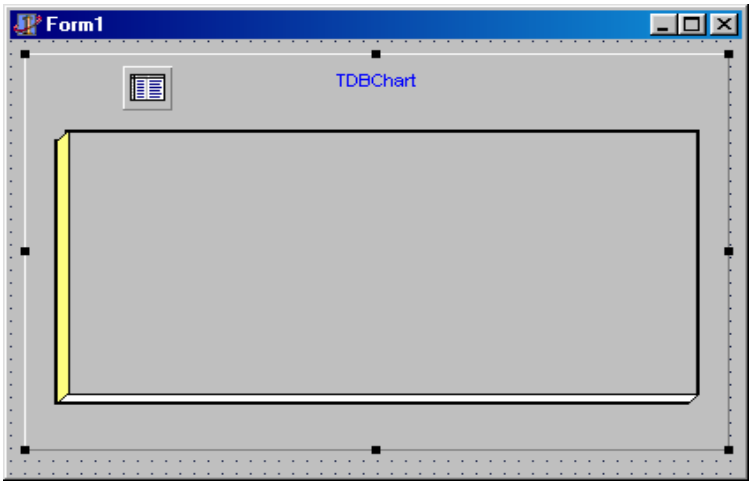



Рисунок 11.5 - Заготовка диаграммы

При разработке приложения свойства компонента **DBChart** устанавливаются с помощью Редактора диаграмм, чье окно **Editing DBChart** показано на рисунке 11.6. Редактор дает возможность оперировать со свойствами-объектами, информация о которых отображается на его страницах, и вызывается двойным щелчком на компоненте **DBChart**. Содержимое окна редактора представляет собой табулированный блокнот. Для нового графика первой всегда показывается закладка **Chart** и для страницы **Chart** - закладка **Series** (рисунок 11.6). Каждая из закладок предназначена для установки параметров того или иного компонента диаграммы. Для каждой диаграммы можно установить: тип, название, оси, описание, источник данных и другие параметры.

Пользователь должен как минимум указать тип диаграммы и источник данных. Тип выбранной диаграммы и ее название отображаются на странице **Chart-Series** Редактора диаграмм. Для добавления новой диаграммы нужно нажать кнопку **Add** (добавление), в результате чего откроется окно с пиктограммами. Например, выбрана линейная диаграмма **Line**. После выбора типа диаграммы, объемного или плоского варианта ее построения и нажатия кнопки **ОК**, диаграмма добавляется к значению свойства **Series** и отображается на соответствующей странице Редактора диаграмм.

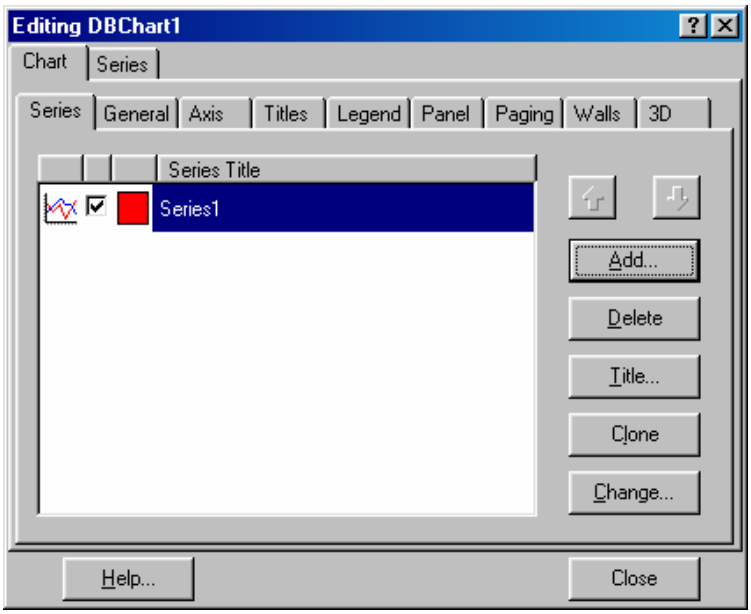



Рисунок 11.6 - Редактор диаграмм

Для выбранной диаграммы можно выполнить следующие действия:

- изменить название по умолчанию (Series1, Series2 и так далее) кнопкой **Title** (Название);
- изменить тип диаграммы кнопкой **Change** (Изменить);
- скопировать диаграмму кнопкой **Clone** (Клонировать);
- удалить диаграмму кнопкой **Delete** (Удалить).

Далее необходимо выбрать источник данных на странице **Series-DataSource** из следующих вариантов:

- **No Data** (значения, вводимые программно);
- **Random Values** (случайные значения);
- **Function** (значения, определяемые выбранной функцией);
- **DataSet** (значения набора данных).

При задании набора данных (**DataSet**) в качестве источника данных для диаграммы становится видимой панель для ввода информации о наборе данных. В списке **DataSet** содержатся имена наборов данных, доступных в модуле той формы, на которой расположен компонент **DBChart**. В лабораторной работе – это набор данных **Table1**. В списке **Labels** выбирается имя поля, данные из которого используются в качестве меток, а в списках **X** и **Y** - соответствующие имена полей, из которых выбираются данные по осям. К моменту выбора полей компонент **Table1** должен быть помещен на форму и связан с нужной таблицей.

Кроме **DataSource** на странице **Series** имеются вкладки **Format**, **Generel**, **Marks**. С помощью **Format** определяются свойства палитры, линий графика и так далее, с помощью **Generel** задаются форматы данных, а закладка **Marks** предназначена для установки марок – значений над точками серии. Марки отображаются на графике, если отмечен переключатель **Visible**. Переключатели **Style** определяют вид марок.

После закрытия окна редактора диаграмма автоматически строится системой Delphi на основании записей, составляющих набор данных. При выпол-

нении приложения диаграмма выглядит также, как и при проектировании. При этом ее функционирование является динамическим, то есть при изменении данных, содержащихся в наборе, диаграмма изменяется автоматически.

### 11.1.3 Rave отчеты

*Отчет* – это печатный документ, содержащий данные, аналогичные получаемым в результате выполнения запроса к базе данных или из некоторого другого источника – электронной таблицы, сообщения электронной почты, текстового документа и других [18]. Можно выделить следующие виды отчетов: простой отчет; отчет с группированием данных; отчет для таблиц, связанных отношением «главный – подчиненный»; составной отчет, объединяющий несколько разных отчетов. В лабораторной работе изучается технология получения простого отчета.

Одним из быстрых способов создания отчета является использование программы **Rave Designer**, являющейся визуальным конструктором отчетов. В состав генератора отчетов входит ядро, которое обеспечивает управление отчетом, предварительный просмотр и отправку на печать. Код ядра при компиляции помещается в приложение, тем самым обеспечивается автономность последнего. Перед тем как начать разрабатывать сам отчет необходимо поместить на форму компоненты **RvProject** (  ) и **RvDataSetConnection** (  ) со страницы палитры компонентов **Rave.**

Запуск программы **Rave Designer** осуществляется из меню командой **Tools => Rave Designer**. После запуска откроется окно программы **Rave Reports**, в котором можно визуально спроектировать отчет. Окно проектировщика состоит из четырех основных частей. В верхней части расположены кнопки управления и панели компонентов. В центре можно видеть проектируемый отчет. В левой части находится редактор свойств текущего объекта, в правой разработчику доступен Просмотрщик объектов.

Для того, чтобы отчет был связан с данными из таблицы, необходимо выбрать в меню **File => New Data Object** и в появившемся окне выбора типов объектов **Data Connection** выбрать **Direct Data View** (Прямой просмотр данных), обеспечивающий просмотр данных для активного соединения с источником данных. Диалоговое окно **Data Connection** показано на рисунке 11.7.

Нажать кнопку **Next>**, в следующем открывшемся окне необходимо выбрать соединение с базой данных. Здесь будет доступно **RvDataSetConnection1** (имя компонента, помещенного на форму Delphi). В Rave Reports в дереве объектов (в правой части окна) появился объект **DataView1**, который обеспечивает доступ к полям таблицы базы данных.

Создание простого отчета можно осуществлять два способами:

- путем конструирования с помощью компонентов **Region component**, **Band component** и **DataBand component** и связи их с таблицей;
- с помощью Мастера создания простых отчетов в таблице.

Первый способ. Для визуальной разработки отчета необходимо помес-

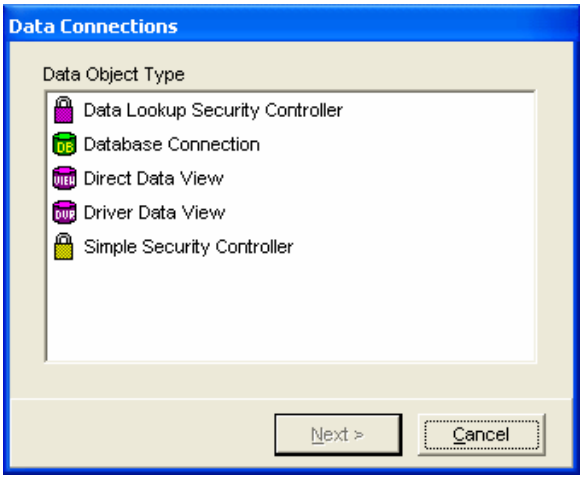



Рисунок 11.7 - Диалоговое окно выбора типов объектов **Data Connection**

тить на рабочее поле компонент **Region component** ( ) с вкладки **Report**. Этот компонент служит для выделения области, на которой размещаются другие компоненты отображения данных, поэтому его нужно растянуть на всю рабочую область.

Для создания заголовка отчета необходимо поместить компонент **Band component** ( ), предназначенный для создания полосы отчета, на которой располагаются другие компоненты отчета. **Band component** будет печататься один раз, размер его можно изменять. Для того, чтобы поместить сам текст заголовка, необходим компонент **Text** ( ) с вкладки **Standart.** В свойство **Text** заносится строка текста, которая будет отображаться в отчете, в данном случае – заголовок отчета. Свойство **Font** позволяет изменить шрифт текста. Аналогично можно поместить любой текст на форму отчета (например, названия столбцов).

Для того, чтобы вывести все записи, поместим на форму компонент **DataBand component** ( ), который задает полосу отчета, представляющую модель строки просмотра данных. Компонент будет печататься столько, сколько записей в таблице базы данных. На **DataBand component** необходимо поместить компоненты, в которые непосредственно будет выводиться текст. Для этого предназначен компонент **DataText** ( ) с вкладки **Report**. Чтобы связать его с полем, надо в свойстве **DataView** выбрать набор данных **DataView1**, а в свойстве **DataField -** то поле, которое будет выводиться. Можно ввести это поле вручную или воспользоваться редактором (рисунок 11.8), вызываемым щелчком кнопки «…». Из раскрывающегося списка можно выбрать нужное поле, и щелчком на кнопке **Insert Field** это поле добавится в **Data Text**. Можно выбрать несколько полей. По окончании нажать кнопку **OK**.

Второй способ. Запуск Мастера создания простых отчетов в таблице осуществляется командой **Tools** => **Report Wizards** => **Simple Table**. На первом этапе создания выбрать вариант **DataView1** и нажать кнопку **Next>**. На

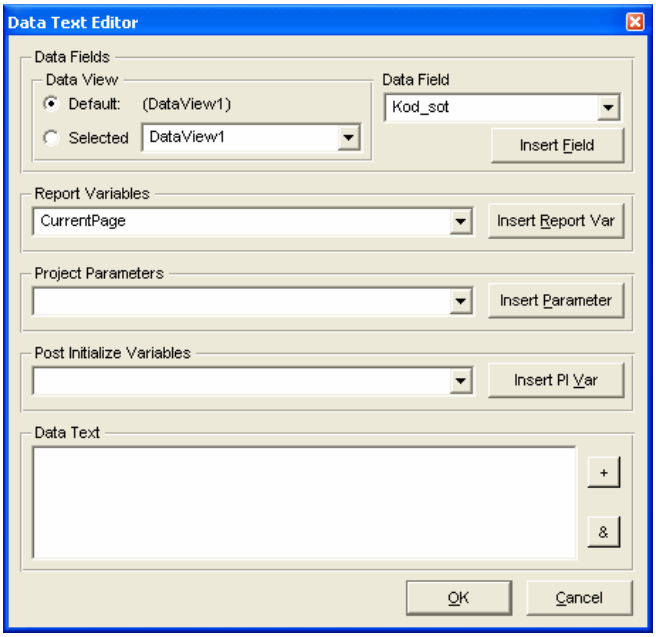



Рисунок 11.8 - Редактор поля

последующих шагах работы с Мастером выбираются поля таблицы для отображения в отчете, при необходимости можно изменить очередность следования полей, установить параметры полей страницы, текст заголовков и шрифты, используемые в отчете. На заключительном этапе работы с мастером нажатием кнопки **Generate** запустить процесс генерации отчета.

Вид формы отчета на этапе разработки приведен на рисунке 11.9. Для изменения заголовка, названий столбцов необходимо задать соответствующие значения свойству **Text** компонентов **Band component** и **DataBand component**. Для того, чтобы просмотреть готовый отчет, нажать клавишу **F9** или выбрать команду **File** => **ExecuteReport**, в открывшемся диалоговом окне **Output Option** в поле **Report Destination** выбрать переключатель **Preview** и нажать **OK**. По завершении разработки отчета необходимо его сохранить **File** => **Save as.**

После создания файла проекта отчета необходимо вернуться в Delphi и в компоненте **RvProject1** изменить значение свойства **ProjectFile** на имя только, что созданного отчета.

Для того, чтобы отчет вызывался, например, по щелчку кнопки, нужно поместить на форму приложения компонент **Button** (значение свойства **Caption** - «Отчет»). В качестве обработчика событий **OnClick** нажатия этой кнопки задать вызов метода ExecuteReport, обеспечивающего выполнение отчета с заданным именем из состава проекта отчета (компонента RvProject1). Необходимо ввести следующий текст:

```
«procedure TForm1.Button2Click(Sender: TObject);
    begin
    RvProject1.Open;
    try
    RvProject1.ExecuteReport('Report1');
```

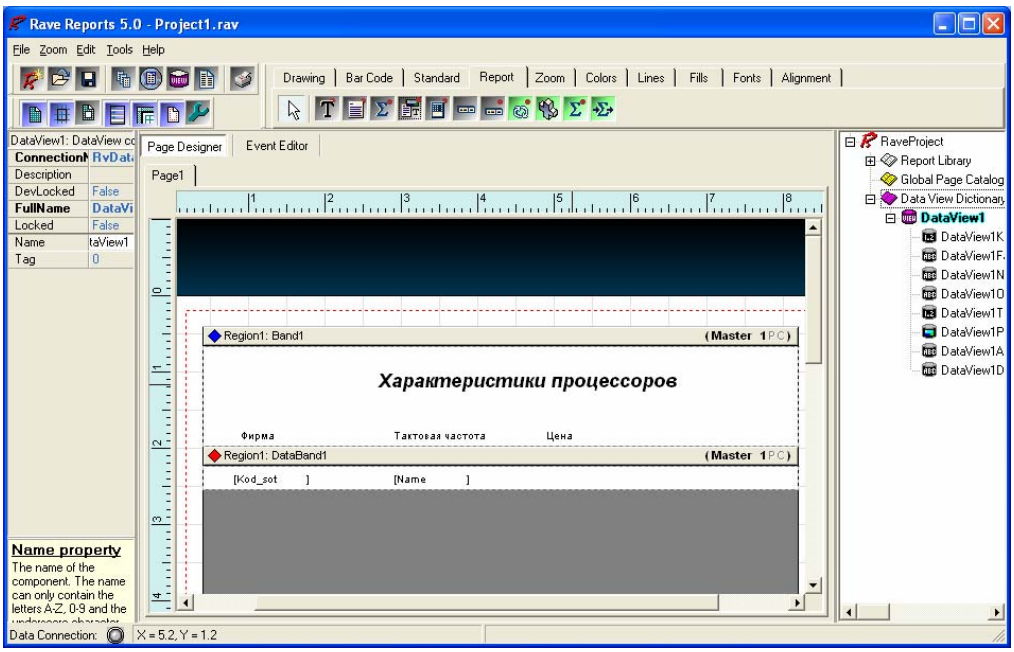



Рисунок 11.9 - **Rave Reports** на этапе разработки

finally
RvProject1.Close;
end;».

Если запустить приложение на выполнение, то после нажатия кнопки «Отчет» в открывшемся диалоговом окне можно выбрать вариант печати простого табличного отчета, содержащего данные из таблицы приложения базы данных.

## 11.2 Задание на выполнение работы

11.2.1 Создать на диске D:\ в папке с именем группы свой каталог, в котором будут храниться все файлы, относящиеся к данной лабораторной работе. При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его в своём каталоге.

11.2.2 Запустить утилиту **Database Desktop**, установить умалчиваемый псевдоним. Задать структуру таблицы «Характеристики процессоров», содержание которой приведено в таблице 11.1. Для таблицы использовать тип Paradox 7. Сохранить созданную структуру в соответствующем файле.

11.2.3 С помощью команды **DataBase Form Wizard** создать экранную форму **Form1** для таблицы. Задать значение True свойству **Active** компонента **Table1**, тем самым сделать таблицу активной. Запустить приложение. Заполнить записями базу данных. Посмотреть работу навигатора.

11.2.4 Создать на форме **Form1** три кнопки **Button** со страницы **Standart**: для просмотра диаграммы на форме **Form2**, для просмотра отчета на форме **Form3** и для выхода из приложения. В таблице 11.2 приведены значения свойств и программные коды для созданных кнопок.

11.2.5 С помощью компонента **DBChart** построить линейный график зависимости цены процессора от значения тактовой частоты, поместив его на форму **Form1** ниже таблицы базы данных в соответствии с рисунком 11.10. Для графика задать заголовок, легенду и марки.

11.2.6 С помощью Мастера диаграмм **TeeChart Wizard** построить круговую диаграмму Pie по полю «Тактовая частота», поместив ее на форму **Form2**. Для диаграммы задать заголовок, легенду и установить марки. На форме поместить кнопку **Button** «Выход» для возврата в форму **Form1**.

Таблица 11.1 – Характеристики процессоров

| Номер | Фирма | Тактовая частота | Шина | Цена в у.е. |
|---|---|---|---|---|
| 1 | Intel Celeron | 433 | PGA-370 | 34 |
| 2 | Intel Celeron | 500 | PGA-370 | 36 |
| 3 | Intel Celeron | 667 | FC-PGA | 45 |
| 4 | Intel Pentium-III | 733 | FC-PGA | 131 |
| 5 | Intel Pentium-III | 750 | FC-PGA | 134 |
| 6 | Intel Pentium-III | 800 | FC-PGA | 148 |
| 7 | Intel Pentium-III | 866 | FC-PGA | 158 |
| 8 | Intel Pentium-III | 933 | FC-PGA | 183 |
| 9 | Intel Pentium-IV | 1300 | Socket 42 | 195 |
| 10 | Intel Pentium-IV | 1500 | Socket 47 | 219 |
| 11 | Intel Pentium-IV | 1700 | Socket 47 | 241 |

Таблица 11.2 – Значения свойств и программные коды для кнопок

| Значение свойства **Name** | Действие | Значение свойства **Caption** | Программный код |
|---|---|---|---|
| **Button1** | Переход к форме **Form2** для показа круговой диаграммы | Диаграмма | Procedure TForm1.Button1Click(Sender: TObject);<br>begin<br>form2.DBChart1.RefreshData;<br>form2.ShowModal;<br>end; |
| **Button2** | Переход к форме **Form3** для показа отчета | Отчет | Procedure TForm1.Button2Click(Sender: TObject);<br>Begin<br>form3.QuickRep1.Preview;<br>end; |
| **Button3** | Выход из **Form1** | Выход | Procedure TForm1.Button3Click(Sender: Tobject);<br>Begin<br>Halt;<br>end; |

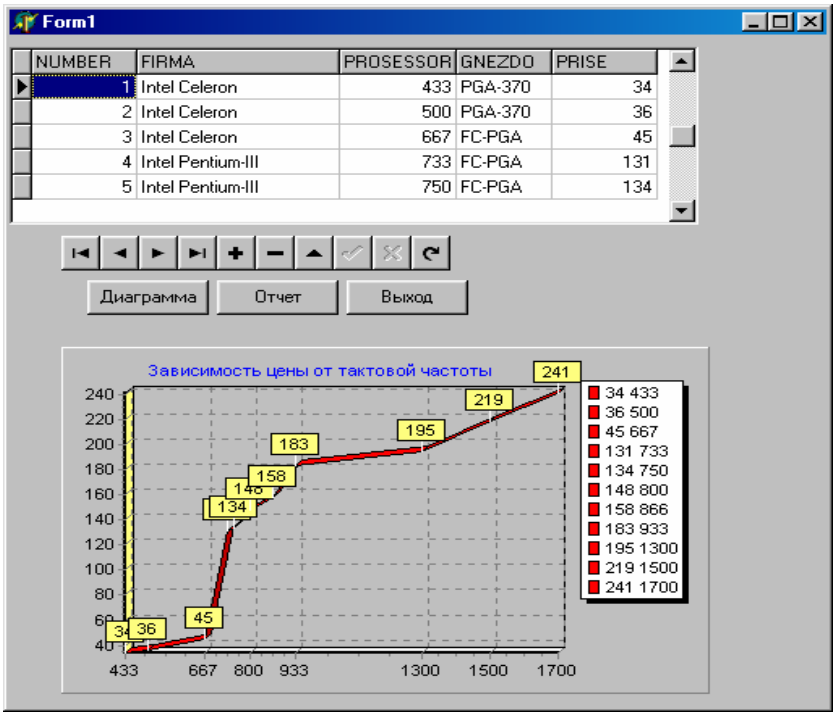



Рисунок 11.10 - Вид формы **Form1**

11.2.7 С помощью программы **Rave Designer** создать для таблицы «Характеристики процессоров» отчет. В отчет поместить все поля, задать заголовок отчета. Вид отчета приведен на рисунке 11.11.

11.2.8 Запустить созданное приложение.

11.2.9 Оформить отчет по лабораторной работе.

## 11.3 Содержание отчёта

11.3.1 Название работы.

11.3.2 Цель работы.

11.3.3 Структура таблицы «Характеристика процессоров».

11.3.4 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

11.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

## 11.4 Тесты и контрольные вопросы

11.4.1 С какой вкладки в диалоговом окне **New Items** запускается команда Мастер диаграмм **TeeChart Wizard**:

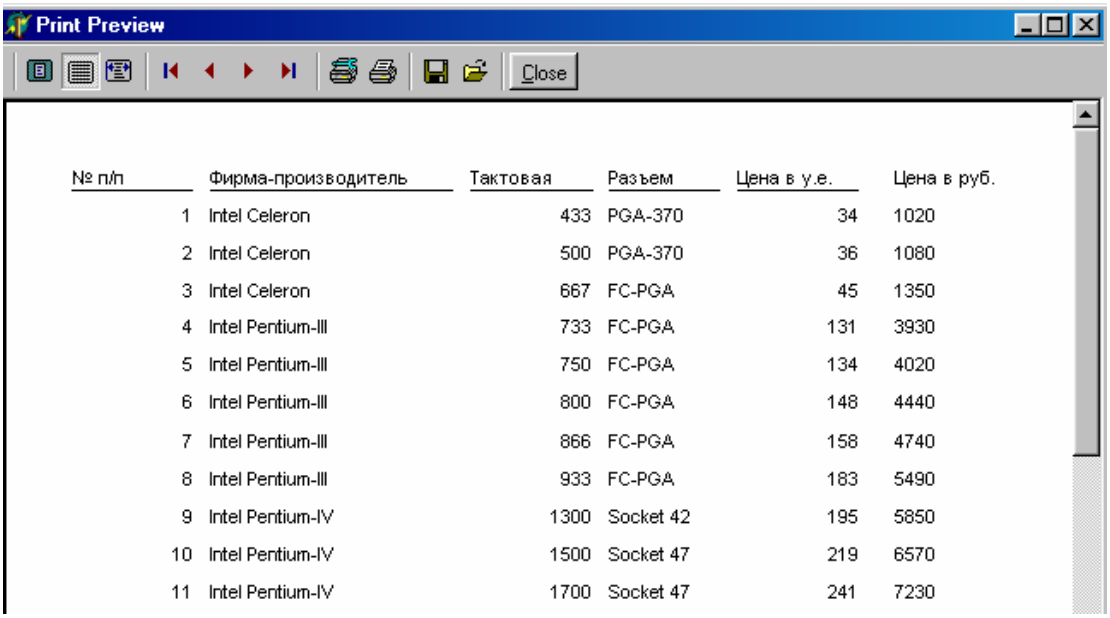
а) **Business**;

б) **Forms**;

в) **Projects**;

г) **Dialogs**;

д) **New**?




| № п/п | Фирма-производитель | Тактовая | Разъем | Цена в у.е. | Цена в руб. |
|---|---|---|---|---|---|
| 1 | Intel Celeron | 433 | PGA-370 | 34 | 1020 |
| 2 | Intel Celeron | 500 | PGA-370 | 36 | 1080 |
| 3 | Intel Celeron | 667 | FC-PGA | 45 | 1350 |
| 4 | Intel Pentium-III | 733 | FC-PGA | 131 | 3930 |
| 5 | Intel Pentium-III | 750 | FC-PGA | 134 | 4020 |
| 6 | Intel Pentium-III | 800 | FC-PGA | 148 | 4440 |
| 7 | Intel Pentium-III | 866 | FC-PGA | 158 | 4740 |
| 8 | Intel Pentium-III | 933 | FC-PGA | 183 | 5490 |
| 9 | Intel Pentium-IV | 1300 | Socket 42 | 195 | 5850 |
| 10 | Intel Pentium-IV | 1500 | Socket 47 | 219 | 6570 |
| 11 | Intel Pentium-IV | 1700 | Socket 47 | 241 | 7230 |

Рисунок 11.11 - Вид отчета

11.4.2 Что такое легенда:

а) область диаграммы, в которой приводится поясняющая информация;

б) область формы, в которой приводится поясняющая информация;

в) пространство диаграммы, в которой приводится поясняющая информация;

г) область диаграммы, в которой можно изменить ее тип;

д) область диаграммы, в которой можно изменить источник данных?

11.4.3 Какой тип диаграммы является круговой диаграммой:

а) Line; б) Pie; в) Bar;

г) Area; д) Point?

11.4.4 На какой странице расположен компонент **DBChart**:

а) **Data Controls**; б) **Data Access**; в) **Standard**;

г) **Additional**; д) **BDE**?

11.4.5 Что задается на вкладке **Series**:

а) тип диаграммы; б) название;

в) оси; г) описание;

д) источник данных?

11.4.6 Что задается на вкладке **Legend**:

а) тип диаграммы; б) название;

в) оси; г) описание;

д) источник данных?

11.4.7 Что задается на вкладке **Axis**:

а) тип диаграммы; б) название; в) оси;

г) описание; д) источник данных?

11.4.8 Что задается на вкладке **Titles**:

а) тип диаграммы; б) название;

в) оси; г) описание;

д) источник данных?

11.4.9 Какая программа Delphi позволяет создать отчет:

а) **BDE Administrator**; б) **Rave Designer**;

в) **Image Editor**; г) **DataBase Desktop**;

д) **SQL Explorer**?

11.4.10 Поставьте в соответствие имя компонента, используемого при построении отчета, и его назначение:

1 - компонент предназначен для создания полосы отчета, на которой располагаются другие компоненты отчета;

2 - компонент служит для выделения области, на которой размещаются другие компоненты отображения данных;

3 - компонент задает полосу отчета, представляющую модель строки просмотра данных;

1 - **Region component**;

2 - **DataBand component**;

3 - **Band component**;

а) 1 - 2, 2 - 1, 3 – 3;

б) 1 - 1, 2 - 3, 3 – 2;

в) 1 - 3, 2 - 1, 3 – 2;

г) 1 - 3, 2 - 2, 3 – 1;

д) 1 - 2, 2 - 3, 3 – 1.

11.4.11 Что такое Мастер диаграмм?

11.4.12 Какие этапы выполняются при построении диаграммы с помощью Мастера диаграмм?

11.4.13 Какие типы диаграмм позволяет создавать Delphi?

11.4.14 Что такое легенда?

11.4.15 Что такое марки?

11.4.16 Для чего предназначен компонент **DBChart**?

11.4.17 Как вызвать окно редактора диаграмм?

11.4.18 Какие закладки имеет редактор диаграмм?

11.4.19 Что устанавливается для диаграммы, построенной с помощью компонента **DBChart**?

11.4.20 Как выбирается источник данных для диаграмм?

11.4.21 Какая программа является визуальным конструктором отчетов?

11.4.22 Как создается отчет при помощи Мастера отчетов?

11.4.23 Что такое отчет?

11.4.24 Что такое простой отчет?

11.4.25 Какие компоненты программы **Rave Designer** использовались?

11.4.26 Какой компонент предназначен для создания полосы отчета, на которой располагаются другие компоненты отчета?

11.4.27 Из каких частей состоит окно проектировщика отчетов?

11.4.28 Какие способы создания простого отчета можно использовать?

11.4.29 Какой компонент используется для создания заголовка отчета?

11.4.30 Какие компоненты используются для ввода текста?

# 12 Реляционный способ доступа к данным при работе в Delphi

Целью работы является создание приложения по работе с базой данных, которое использует структурированный язык запросов SQL в среде программирования Delphi.

## 12.1 Общие положения

### 12.1.1 Особенности языка SQL при работе в Delphi

Реляционный способ доступа к данным основывается на операциях с группами записей. Для задания операций используются средства языка структурированных запросов SQL, поэтому реляционный способ доступа называют также SQL-ориентированным. Средства SQL применимы для выполнения операций с локальными и удаленными базами данных (БД). Применительно к локальным БД использование реляционного способа доступа не дает существенного преимущества, но в этом случае с помощью SQL-запроса можно:

- формировать состав полей набора данных при выполнении приложения;
- включать в набор данных поля и записи из нескольких таблиц;
- отбирать записи по сложным критериям;
- сортировать набор данных по любому полю, в том числе неиндексированному;
- осуществлять поиск данных по частичному совпадению со значениями в поле.

Многие из названных действий не применимы к набору данных **Table**.

Язык SQL ориентирован на выполнение действий с таблицами БД и данными в этих таблицах, а также некоторых вспомогательных действий. В отличие от процедурных языков программирования, в нем нет операторов управления вычислительным процессом (циклов, переходов, ветвления) и средств ввода-вывода.

Составленную на языке SQL программу называют SQL-запросом. Для его реализации в приложениях Delphi в качестве набора данных должен применяться компонент **Query**, позволяющий выполнять SQL-запрос. Текст SQL-запроса является значением свойства SQL компонента **Query** и формируется или при разработке приложения, или во время его выполнения. Компонент **Query** обеспечивает выполнение SQL-запроса и получение соответствующего набора данных. Формирование набора данных выполняется при активизации компонента **Query** путем вызова метода **Open** или установкой свойству **Active** значения **True**.

Набирать и выполнять в интерактивном режиме текст SQL-запроса позволяют инструментальные программы, поставляемые вместе с Delphi, например такие, как **Database Desktop**, **SQL Builder**, **SQL Explorer**.

Проверка синтаксиса и отработка запроса, встроенного в приложение (чаще всего с помощью компонента **Query**), производится на этапе выполнения

приложения. После отладки текст запроса вставляется в разрабатываемое приложение.

В качестве результата выполнения SQL-запроса может возвращаться набор данных, который составляют отобранные с его помощью записи. Этот набор данных называют результирующим.

Для создания приложения по работе с SQL-запросом необходимо выполнить следующие действия:

- открыть пустую форму, например **Form1**, выполнив команду главного меню **File** => **New Application** (Новое приложение);
- расположить на форме компонент **Query**  , взяв его со страницы **BDE** палитры компонентов, установить в свойство **Query1.DatabaseName** значение псевдонима базы данных;
- расположить на форме компонент **DataSource**, связать этот компонент с компонентом **Query1**, установив в свойство **DataSet** значение Query1;
- расположить на форме компонент **DBGrid** со страницы **DataControls**, связать этот компонент с **DataSource1**, установив в свойство **DataSource** значение DataSource1;
- активизировать компонент **Query1**, раскрыть окно SQL Builder;
- сформировать SQL-запрос;
- установить в свойство **Query1.Active** значение True;
- сохранить созданный модуль и проект.

### 12.1.2 Использование SQL Builder

SQL Builder представляет собой встроенное в Delphi средство, позволяющее создать SQL-запрос без какого-либо знания языка SQL. С помощью этой программы разработчик может достаточно удобно конструировать запросы, сохраняя их в виде текстового файла с расширением SQL.

Программа SQL Builder вызывается выбором из контекстного меню компонента **Query** команды **SQL Builder**. Во время работы программы SQL Builder нельзя перейти в другие окна Delphi, такой переход возможен только по завершению работы с ней.

В верхней части окна программы SQL Builder размещаются таблицы, выбранные для построения запроса, а в нижней части – многостраничный блокнот.

Чтобы добавить в окно программы таблицу, нужно в поле **Database** указать расположение базы данных, выбрав псевдоним из раскрывающегося списка или введя вручную путь к каталогу с файлами БД. После этого в списке **Table** выбирается нужная таблица, которая автоматически добавляется в верхнюю часть окна. Добавляемые таблицы могут располагаться и в разных каталогах.

Программа SQL Builder позволяет достаточно просто и удобно выполнить связывание (соединение) таблиц. Для этого нужно выбрать мышью поле в одной таблице и перетащить его в используемое для связи поле другой таблицы. После отпускания кнопки мыши между таблицами устанавливается связь,

что отображается линией, соединяющей эти таблицы. В обеих таблицах поля, используемые для связи, отображаются в списке полей первыми (верхними) и выделяются жирным шрифтом. В отличие от ряда других программ, например, Microsoft Access, эта линия соединяет названия таблиц, а не используемые для связи поля обеих таблиц.

Напомним, что в главной таблице поле для связи должно быть ключевым, а в подчиненной таблице – индексным. Поля связи должны иметь одинаковые или совместимые типы, в противном случае выдается сообщение об ошибке несоответствия типов. На рисунке 12.1 показано окно визуального построителя SQL Builder с установленными связями межу таблицами.

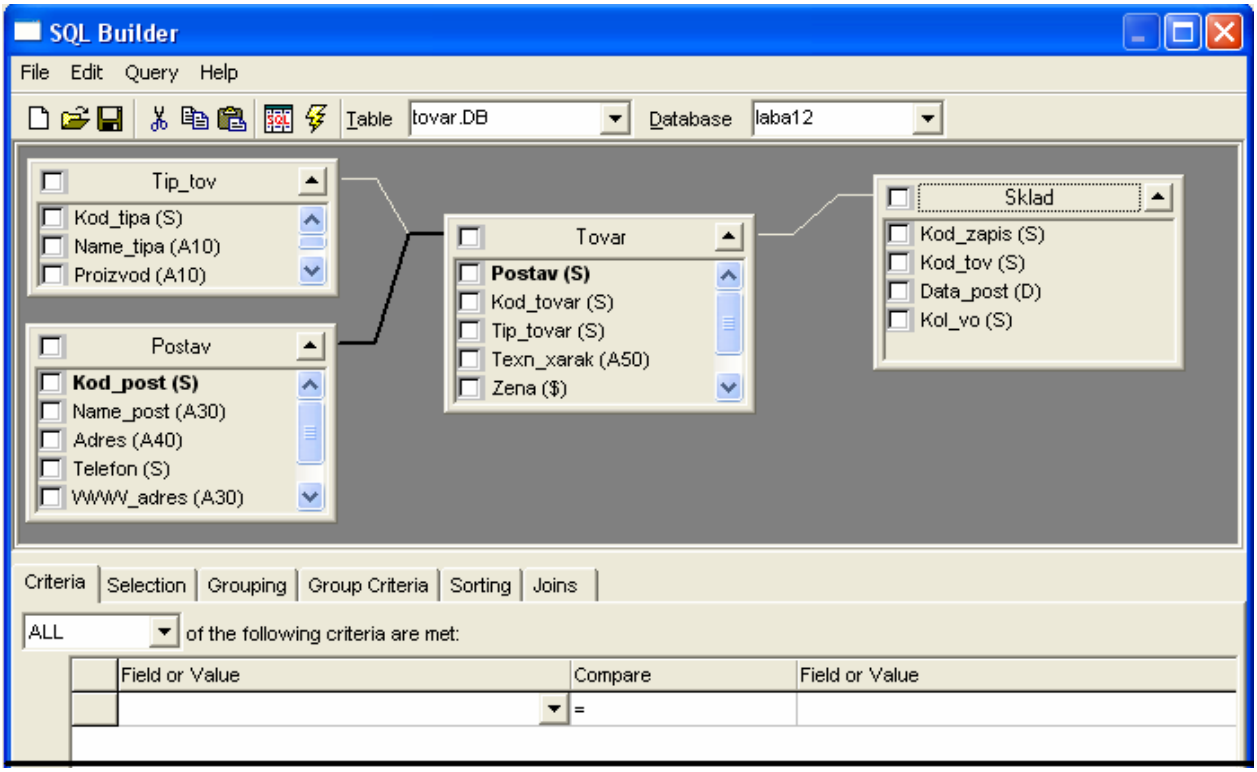



Рисунок 12.1 - Формирование запроса к базе данных с использованием визуального построителя

Для удаления связи нужно щелкнуть правой кнопкой мыши на линии, обозначающей связь между таблицами, вызвав контекстное меню, и выполнить его единственную команду **Delete Join** (Удаление связи). После подтверждения этой операции связь удаляется.

Таблица представляется изображением, похожим на комбинированный список. В верхней части этого списка находятся псевдоним (Alies) таблицы, флажок и стрелка. Стрелка позволяет свернуть или развернуть список, а флажок служит для управления включением всех полей таблицы в результат запроса или исключением из него.

Для смены псевдонима таблицы нужно вызвать команду **Edit Table Alies** контекстного меню этой таблицы. При этом в названии псевдонима отображается текстовый курсор, что показывает готовность его к изменению. После на-

жатия клавиши **Enter** новое название псевдонима вступает в действие. На рисунке 12.2 приведен выбор отображаемых полей и изменение название псевдонима таблицы.

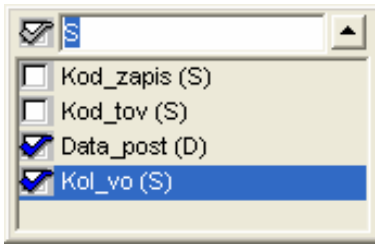



Рисунок 12.2 – Выбор отображаемых полей и изменение названия псевдонима таблицы

Удаление таблицы из окна программы выполняется командой **Remove Table** контекстного меню этой таблицы. Можно также выделить таблицу, щелкнув на ее изображение, при этом вокруг ее псевдонима отобразится пунктирный прямоугольник. После выделения таблица удаляется нажатием клавиши **Delete**.

В нижней части окна программы SQL Builder находится блокнот, который автоматически появляется при добавлении к окну программы первой таблицы. С помощью блокнота можно включать в запрос условия отбора записей или указывать направления их сортировки.

Страница **Criteria** (Условия) позволяет задать условия отбора записей. Каждое условие вводится в отдельной строке и состоит из двух имен полей или значений, разделенных операцией сравнения. Имена полей выбираются из списка или перетаскиваются мышью из соответствующей таблицы в верхней части окна. Операция сравнения выбирается из списка. Если условие введено с ошибкой, то выдается соответствующее сообщение, а само условие в запрос не включается и отображается красным цветом. На рисунке 12.3 приведен пример на формирование условия на отбор записей.

Удаление строки с условием выполняется командой **Delete Row** (удаление строки) контекстного меню удаляемой строки. Отметим, что аналогично можно удалить строку и из списка других страниц блокнота.

Список **of the following criteria are met** (… из перечисленных условий выполнены) позволяет задать логические операции, которые связывают строки с отдельными условиями. В левой части строк с условиями отображаются на-

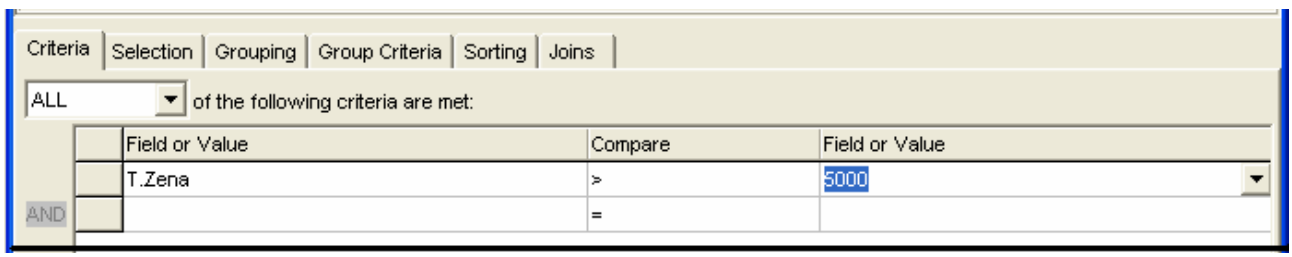



Рисунок 12.3 – Формирование условия на отбор записей

звания логических операций. По умолчанию список содержит значение **ALL** (ВСЕ), что соответствует операции логического умножения.

Страница **Selection** (Отбор) позволяет изменить названия заголовков столбцов, используемых для отображения значений полей в результирующем наборе, а также включить в запрос статистические функции по полям.

В правой части списка указываются поля, включенные в результат запроса – им соответствуют установленные флажки в изображениях таблиц. В левой части списка для каждого поля помещается название соответствующего столбца, которое по умолчанию совпадает с именем поля (без указания имени таблицы).

Добавление к запросу статистической функции выполняется с помощью команды **Summary** (Итог) контекстного меню страницы. В результате появляются списки, из которых выбираются функция (например, SUM) и поле.

Переключатель **Remove Dublicates** (Исключить повторы) позволяет убрать из результирующего набора одинаковые записи. По умолчанию он выключен, и в результат попадают все записи (в том числе и совпадающие), которые отвечают заданным условиям отбора.

Страница **Grouping** (Группирование) позволяет сгруппировать записи по полям. Список **Output Fields** (Выходные поля) содержит поля, которые можно использовать для группирования записей. В список **Grouped On** (Группирование по) отбираются имена полей, по которым выполняется группирование записей. Перемещение полей первого списка во второй и удаление полей из второго списка выполняется с помощью кнопок Add (Добавить) и **Remove** (Удалить), которые становятся активными при выборе поля (полей) в соответствующем списке. Перемещение поля между списками можно также выполнить двойным щелчком на его имени.

Доступными являются поля, флажок которых в изображении таблицы отмечен. Имя поля состоит из псевдонима (имени) таблицы и названия столбца для этого поля, заданного на странице **Selection.**

Страница **Group Criteria** (Критерии группирования) содержит условия, используемые для группирования записей. Условия группирования вводятся аналогично условиям отбора записей, задаваемым на странице **Criteria.**

Страница **Sorting** (Сортировка) определяет порядок сортировки записей. Список **Sorted By** (Сортировка по) содержит имена полей, по которым выполняется сортировка. Порядок следования полей определяет последовательность сортировки: сначала записи упорядочиваются по значениям поля, указанного в списке первым (верхним), затем записи, имеющие одинаковое значение первого поля, упорядочиваются по второму полю и так далее. Для изменения порядка расположения полей, по которым выполняется сортировка, служат кнопки с изображением черных треугольников, расположенные над именами полей. Кнопка с треугольником, направленным вниз, перемещает выделенное поле (поля) на одну строку вниз, кнопка с треугольником, направленным вверх, - на одну строку вверх.

По умолчанию сортировка выполняется в порядке возрастания значения поля, на что указывает признак **Ascending** (По возрастанию) справа от имени

поля. Для задания обратного порядка сортировки по полю нужно его выделить и нажать кнопку **Z..A,** при этом вместо **Ascending** устанавливается признак **Descending** (По убыванию). Нажатие кнопки **A..Z** возвращает прежний порядок сортировки.

Список **Output Fields** содержит поля, которые можно использовать для сортировки записей. Обозначение полей и способы их перемещения между списками не отличаются от обозначения полей и способов их перемещения для страницы **Grouping**.

Страница **Joins** позволяет устанавливать связи (соединения между таблицами).

### 12.2 Задание на выполнение лабораторной работы

12.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к лабораторной работе.

12.2.2 При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его в созданном каталоге.

12.2.3 Запустить утилиту **Database Desktop**, установить умалчиваемый псевдоним.

12.2.4 Задать структуры таблиц «Поставщики», «Типы товаров», «Товар», «Склад» в соответствии с таблицами 12.1 – 12.4. Для таблиц использовать тип Paradox 7. Определить ссылочную целостность между таблицами в соответствии с типами связей, существующих между таблицами.

12.2.5 Сохранить созданную структуру в соответствующих файлах. Занести в описанные таблицы соответствующие записи.

12.2.6 На **Form1** поместить компоненты, задать их свойства, чтобы при запуске приложения на этой странице появлялся заголовок «Поступление товаров» и имелись переключатели для открытия форм «Поставщики», «Товары» и «Склад».

12.2.7 На форме «Поставщики» должен открываться список со всеми поставщиками, включающий поля «Наименование», «Адрес», «Телефон», «WWW-адрес» и сведения о количестве поставщиков.

12.2.8 На форме «Товары» должно быть меню для выбора режимов: «Все товары» (поля – «Наименование типа товара», «Производитель», «Технические характеристики», «Наименование поставщика», «Цена»; сортировка по убыванию цены), «Мониторы» (поля – «Технические характеристики», «Производитель»; сортировка по производителю), «Процессоры» (поля – «Технические характеристики», «Поставщик»; сортировка по поставщику), «С ценой >5000» («Наименование типа товара», «Цена», «Поставщик», «Производитель», группировка по типу товара).

12.2.9 На форме «Склад» должен открываться список всех имеющихся на складе товаров (поля – «Наименование типа товаров», «Цена», «Дата поступления», «Количество», «Стоимость») и общая их стоимость. Пример формы «Склад» приведен на рисунке 12.1.

12.2.10 Оформить отчет по лабораторной работе.

Таблица 12.1 - Поставщики

| Код поставщика | Наименование | Адрес | Телефон | WWW- адрес |
|---|---|---|---|---|
| 1 | Компьютерный салон «Центр» | ул. Пролетарская, 42 | 774711 | http://www.kscenter.ru |
| 2 | Компьютерный магазин «Байт» | ул. Дзержинского, 20 | 369582 | http://www. mbyte.ru |
| 3 | ООО «Компания «Мехатроника» | ул. Гая, 5 | 780757 | http://www.Mtron.ru |
| 4 | Компьютерный салон «Глюк» | ул. Володарского, 10 | 777665 | http://www.Gluckol.net |
| 5 | ТЦ «Джаз» | ул. Володарского, 27 | 770268 | http://www.jazz-comtutor.nm.ru |
| 6 | ООО «Константа - Сервис» | пр-т Парковый, 46 | 724364 | http://www.Constanta.ru |

Таблица 12.2 - Типы товаров

| Код типа | Наименование | Производитель |
|---|---|---|
| 10 | Монитор | Samsung |
| 15 | Монитор | LCD Acer |
| 20 | Монитор | LCD LG |
| 25 | Принтер | Samsung |
| 30 | Принтер | HP |
| 35 | Процессор | Intel Celeron |
| 40 | Процессор | AMD |

Таблица 12.3 – Товары

| Код товара | Тип товара | Технические характеристики | Поставщик | Цена |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 100 | 10 | 17" 723N AKS <Silver> (LCD, 1280x1024) | 4 | 4704 |
| 102 | 20 | 17" L1734S-BN Flatron <Black> (LCD, 1280x1024) | 1 | 8960 |
| 104 | 30 | LaserJet P1005 <CB410A> A4 14с./мин 2Мб USB2.0 | 3 | 10818 |
| 110 | 35 | CPU Intel Celeron 430 1.8 ГГц/ 512K/ 800МГц 775-LGA | 5 | 1030 |
| 112 | 15 | 17" AL1716Fs (LCD, 1280x1024) | 5 | 14900 |
| 114 | 25 | ML-1640 (A4, 8Mb, лазерный, 16 с./мин, 1200dpi, USB 2.0) | 6 | 6432 |
| 116 | 35 | CPU Intel Celeron D 336 2.8 ГГц\ 256K\ 533МГц 775-LGA | 1 | 2400 |
| 118 | 10 | 19" 932B SBV <Black> (LCD, 1280x1024, +DVI) | 1 | 6784 |

Продолжение таблицы 12.3

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 120 | 15 | 19" MONITOR Acer V193bm <Black> (LCD, 1280x1024) | 1 | 4672 |
| 122 | 25 | SCX-4200 (18 с./мин, 8Mb, лазерн.принтер A4, копир, сканер) | 3 | 16380 |
| 124 | 30 | hp LaserJet P1505 <CB412A> A4 23с./мин 2Мб USB2.0 | 4 | 4384 |
| 128 | 40 | CPU Intel Pentium 4 631 3.0 ГГц/ 2Мб/ 800МГц 775-LGA | 1 | 3904 |
| 130 | 35 | CPU Intel Core 2 Duo E6550 2.33 ГГц/ 4Мб/ 1333МГц 775-LGA | 2 | 2144 |
| 132 | 30 | LaserJet P2014 <CB450A> A4, 23 с./мин 32Mb USB2.0/LPT | 4 | 10912 |
| 134 | 25 | 20" V203Wb <Black> (LCD, Wide, 1680x1050) | 6 | 16448 |
| 136 | 15 | 19" L1942S-BF Flatron <Black> (LCD, 1280x1024) | 4 | 4832 |
| 138 | 20 | 19" 923NW NKS <Silver> (LCD, Wide, 1440x900) | 4 | 6208 |
| 140 | 10 | CPU Intel Core 2 Duo E6550 2.33 ГГц/ 4Мб/ 1333МГц 775-LGA | 1 | 7104 |

Таблица 12.4 – Склад

| Код записи | Код товара | Дата поступления | Количество |
|---|---|---|---|
| 1 | 102 | 12.01.2008 | 5 |
| 2 | 130 | 17.01.2008 | 10 |
| 3 | 108 | 17.01.2008 | 12 |
| 4 | 108 | 18.01.2008 | 8 |
| 5 | 104 | 20.01.2008 | 5 |
| 6 | 110 | 25.01.2008 | 3 |
| 7 | 124 | 26.01.2008 | 4 |
| 8 | 112 | 27.01.2008 | 20 |
| 9 | 140 | 12.02.2008 | 15 |
| 10 | 134 | 15.02.2008 | 10 |

### 12.3 Содержание отчёта

12.3.1 Название работы.

12.3.2 Цель работы.

12.3.3 Логические схемы данных созданных таблиц.

12.3.4 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

12.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

12.3.6 Формулировки всех SQL-запросов.

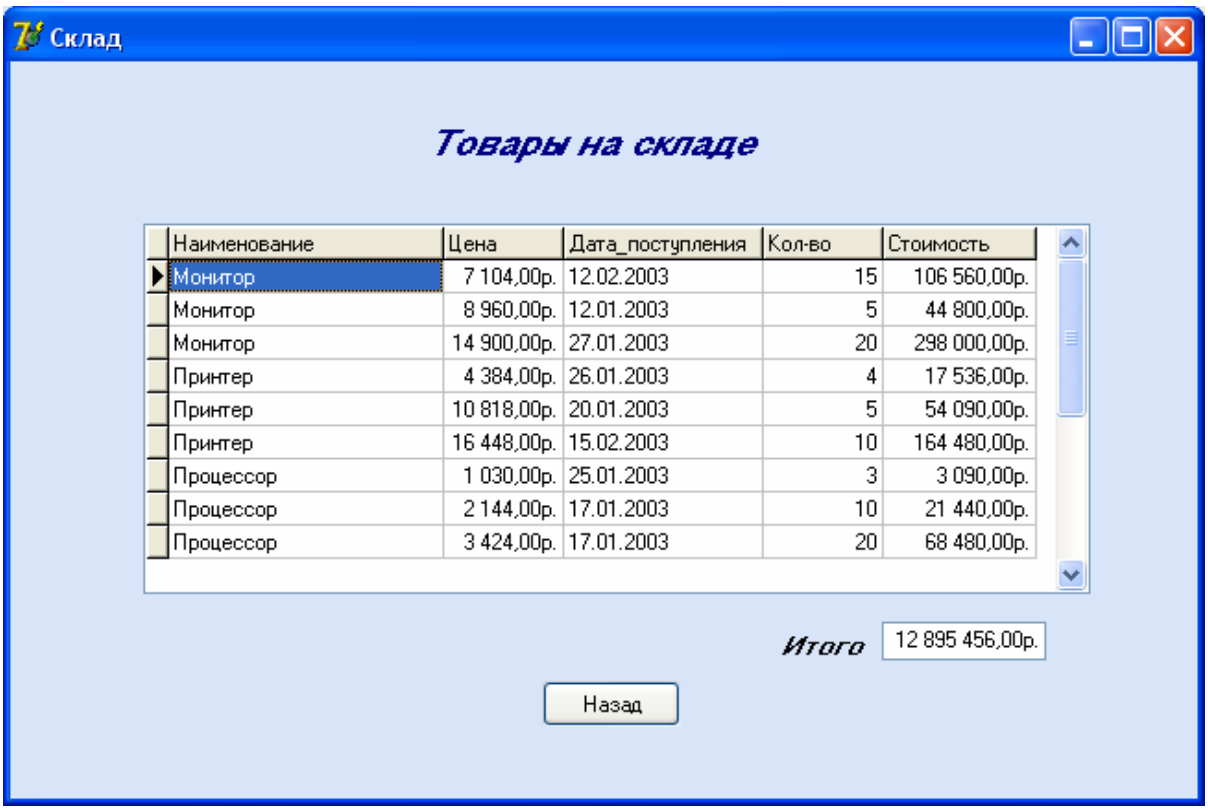



Рисунок 12.1 – Форма «Склад»

### 12.4 Тесты и контрольные вопросы

12.4.1 Какой компонент используется для создания SQL-запроса:
а) **Table**; б) **Label**;
в) **Query**; г) **DBEdit**;
д) **DBGrid**?

12.4.2 Для чего предназначена программа **SQL Builder**:
а) для выполнения в интерактивном режиме текста SQL-запроса;
б) для программирования в интерактивном режиме текста SQL-запроса;
в) для выбора в интерактивном режиме текста SQL-запроса;
г) для набора в интерактивном режиме текста SQL-запроса;
д) для сохранения в интерактивном режиме текста SQL-запроса?

12.4.3 Из контекстного меню какого компонента можно вызвать команду **SQL Builder**:
а) **Table**; б) **Query**;
в) **DBGrid**; г) **DBEdit**;
д) **Label**?

12.4.4 Какой командой можно изменить псевдоним таблицы:
а) **Edit Table Alies**;
б) **Remove Table**;
в) **Delete Join**;
г) **Delete**?

12.4.5 Какой командой можно удалить связь между таблицами:

а) **Edit Table Alies**; б) **Remove Table**;
в) **Delete Join**; г) **Delete**?

12.4.6 Какому свойству компонента **Query** необходимо установить значение **True**, чтобы осуществить формирование набора данных:

а) **Name**; б) **Active**;
в) **DataSource**; г) **SQL**;
д) **DatabaseName**?

12.4.7 На каком этапе осуществляется проверка синтаксиса и отработка SQL-запроса:

а) на этапе формирования приложения;
б) на этапе формирования запроса;
в) на этапе создания программного кода;
г) на этапе выполнения приложения;
д) на этапе формирования состава полей набора данных?

12.4.8 Связь компонентов **DataSource** и **Query** устанавливается через свойство:

а) **DataSource**; б) **Name**;
в) **DataSet**; г) **Query**.

12.4.9 На какой странице расположен компонент **Query**:

а) **Data Controls**; б) **Data Access**; в) **Standard**;
г) **Additional**; д) **BDE**?

12.4.10 Какая функция используется для определения общей стоимости товара:

а) AVG; б) COUNT; в) MAX;
г) VAR; д) SUM?

12.4.11 Что такое SQL?

12.4.12 В чем заключаются особенности языка SQL в Delphi?

12.4.13 Для чего предназначен SQL?

12.4.14 Для чего предназначен оператор **SELECT**?

12.4.15 Что включает синтаксис оператора **SELECT**?

12.4.16 Для чего предназначено средство SQL Builder?

12.4.17 В чем заключается достоинство SQL Builder?

12.4.18 Для чего используется компонент **Query**?

12.4.19 Какие панели используются в SQL Builder для уточнения запроса?

12.4.20 Для чего используется флажок в изображении таблицы в SQL Builder?

12.4.21 Какая команда позволяет сменить псевдоним таблицы?

12.4.22 Что формируется с помощью панели **Criteria**?

12.4.23 Что формируется с помощью панели **Selection**?

12.4.24 Что формируется с помощью панели **Grouping**?

12.4.25 Что формируется с помощью панели **Sorting**?

12.4.26 Как осуществить связывание таблиц в программе SQL Builder?

# 13 Работа с таблицами данных в Delphi

Целью работы является создание в системе программирования Delphi приложения по работе с многотабличной базой данных, которое демонстрирует использование наборов данных.

## 13.1 Общие положения

### 13.1.1 Кнопка с рисунком

В Delphi наряду с кнопкой **Button** (  ) используется кнопка с рисунком **BitBtn** (  ) со страницы **Additional**. Она отличается от стандартной кнопки тем, что помимо заголовка имеет возможность отображения растрового рисунка (*глифа*), видом и размещением которого на поверхности кнопки можно управлять с помощью свойств.

Свойство **Glyph** определяет растровое изображение кнопки. Изображение должно быть сохранено в файле растрового формата BMP. По умолчанию свойство **Glyph** имеет значение None, то есть кнопка не содержит рисунка. Кнопка одновременно может поддерживать до четырех изображений, каждое из которых соответствует определенному состоянию (кнопка не нажата, не активна, в момент щелчка, нажата). Количество подключаемых пиктограмм определяется свойством **NumGlyphs**. По умолчанию свойство **NumGlyphs** имеет значение 1, для отображения на кнопке всегда используется первое изображение.

Delphi предлагает для кнопки **BitBtn** несколько предопределенных видов, выбираемых с помощью свойства **Kind**. При выборе какого-либо вида для кнопки на ней отображается соответствующий глиф. Для задания вида кнопки могут использоваться следующие константы:

- bkCustom (кнопка имеет выбранное изображение, первоначально изображение отсутствует и его нужно загружать дополнительно);
- bkOK (кнопка имеет зеленую галочку и надпись «OK»; свойству **Default** кнопки установлено значение True, а свойству **ModalResult** - значение mrOK);
- bkCancel (кнопка имеет красный знак «x» и надпись «Cancel»; свойству **Cancel** кнопки установлено значение True, а свойству **ModalResult** - значение mrCancel);
- bkYes (кнопка имеет зеленую галочку и надпись «Yes»; свойству **Default** кнопки установлено значение True, а свойству **ModalResult** - значение mrYes);
- bkNo (кнопка имеет значение в виде красной перечеркнутой окружности и надпись «No»; свойству **Cancel** кнопки установлено значение True, а свойству **ModalResult** – значение mrNo);
- bkHelp (кнопка имеет изображение в виде сине-красного вопросительного знака и надпись «Help»);
- bkClose (кнопка имеет изображение в виде двери с обозначением выхода и надпись «Close», при нажатии кнопки форма автоматически закрывается);
- bkAbort (кнопка имеет красный знак «x» и надпись «Abort»);

- bkRetry (кнопка имеет изображение в виде зеленой стрелки повтора операции и надпись «Retry»);

- bkIgnore (кнопка отображает изображение игнорирования и надпись «Ignore»);

- bkAll (кнопка имеет изображение в виде двойной зеленой галочки и надпись «Yes to All»).

По умолчанию свойство **Kind** имеет значение bkCustom и пользователь может сам выбирать изображение, задавая значение свойства **Glyph**.

Расположением изображения на поверхности кнопки относительно надписи управляет свойство **Layout**,принимающее следующие значения:

- blGlyphLeft (изображение слева от надписи, значение по умолчанию);
- blGlyphRight (изображение справа от надписи);
- blGlyphTop (изображение над надписью);
- blGlyphBottom (изображение под надписью).

С помощью свойства **Margin** можно управлять выравниванием глифа и надписи относительно края поверхности кнопки. Это свойство задает расстояние в пикселях между краем кнопки и изображением и имеет по умолчанию значение «-1», что приводит к расположению изображения и надписи по центру кнопки.

Свойство **Spacing** определяет в пикселях размер промежутка, отделяющего глиф от надписи. По умолчанию значение этого свойства равно «-1», что приводит к центрированию надписи между краем глифа и дальним от него краем кнопки.

### 13.1.2 Создание заставки

Обычно программа в начале работы выводит на экран окно (панель) с краткой информацией о себе. Это окно присутствует на экране в процессе загрузки приложения, после чего исчезает автоматически или по команде пользователя. Подобное информационное окно называется *заставкой*.

Заставка позволяет занять время во время загрузки и сообщить пользователю дополнительные сведения о программе. Она сделает приложение более профессиональным, дает возможность приукрасить внешний вид, придать солидность. Кроме того, заставка позволяет не только отличить одно приложение от другого, но и отличить одну версию приложения от другой. На заставку можно поместить все, что угодно: от бесполезной информации, какого-то рисунка до названия программы, версии продукта, лицензии и контактного телефона.

Для отображения заставки можно использовать различные приемы. В лабораторной работе рассматривается случай, когда заставка представляет собой специально предназначенную для этого форму, которая при запуске приложения создается и отображается первой. Далее создается главная форма приложения, в которой выполнена задержка на время отображения заставки на экране. По истечении этого времени заставка удаляется с экрана и из памяти, а выполнение приложения продолжается обычным способом.

Если при создании главной формы выполняется большое количество действий, то задержка может не потребоваться. Но в этом случае время отображения заставки на экране будет зависеть от производительности компьютера.

Для отображения заставки описанным способом необходимо внести определенные изменения в файлы проекта и модуля главной формы. Пусть главная форма проекта называется **Form1**, а окно с заставкой - **Form2**. Необходимо разместить на форме-заставке **Form2** компонент **Image** (  ) со страницы **Additional**. Визуализация изображения компонента **Image** осуществляется при помощи свойства **Picture**. Нажатие на кнопку  в данном свойстве вызывает открытие окна редактора рисунка, с помощью которого можно вставить рисунок, после нажатия на кнопку «**Load…**» необходимо указать путь к файлу с рисунком, после указания пути нажать кнопку **OK**. Чтобы изображение пропорционально растягивалось или уменьшалось до размеров компонента, необходимо установить значение True свойства **Stretch**.

Для создания информационного текста на заставке можно использовать компоненты **Label** (  ) или **Memo** (  ). Для компонента **Label** со страницы **Standard** в свойство **Caption** устанавливается необходимое значение. При запуске приложения введенное значение будет той же надписью, которое отобразится на элементе управления. Для компонента **Memo** со страницы **Standard** в свойстве **Lines** нажать на кнопку  , в окно редактора текста **String List Editor** ввести необходимый текст, затем нажать кнопку **OK**. Свойство **Font** компонентов позволяет изменить шрифт надписи, начертание, размер, цвет и другие.

Затем для настройки формы **Form2** необходимо установить:

- значение poScreenCenter для свойства **Position**, чтобы окно-заставка располагалась посередине экрана;
- значение bsNone для свойства **BorderStyle**, чтобы у формы не было заголовка и системных кнопок;
- значение False свойства **Visible**.

Чтобы по истечении некоторого времени, например, пяти секунд, форма-заставка закрывалась, необходимо на форму поместить компонент **Timer** (  ) со страницы **System**. Его свойству **Interval** задать значение 5000 (значение 1000 соответствует закрытию формы через 1 секунду). В обработчик события компонента **OnTimer** необходимо написать: «Close;».

Сделать активной форму-заставку **Form2**. В обработчик события **OnClick** написать «Close;». Установить свойство формы **KeyPreview** в True (это делается для того, чтобы при нажатии любой клавиши вначале реагировала форма, а затем тот элемент, который был в фокусе в момент нажатия). А в обработчик события **OnKeyPress** также написать «Close;».

Форма-заставка готова полностью и теперь необходимо, чтобы она запускалась перед главной формой. Для этого необходимо сделать активной главную форму **Form1**, перейти на вкладку **Events** в Object Inspector и выбрать событие **OnShow**. В обработчике этого события надо написать следующее: «Form2.showmodal;».

Существуют и другие способы создания формы-заставки.

### 13.1.3 Создание информационного окна

Большинство приложений способно отображать информационное окно, в котором выводится краткая информация о приложении. Обычно это сведения о названии программного продукта, его версии, дате выпуска, авторах, а также другие данные и небольшой рисунок. Информационное окно часто имеет название «О программе» (About).

Создать такое окно можно на основе шаблона или самостоятельно.

При использовании шаблона в Хранилище объектов (**File** => **New** => **Other…**) в диалоговом окне **New Items** необходимо выбрать закладку **Forms** (Формы), затем выбрать шаблон формы **About box** (Информационное окно). В результате к приложению добавляется форма с именем **AboutBox** и заголовком **About** (рисунок 13.1).

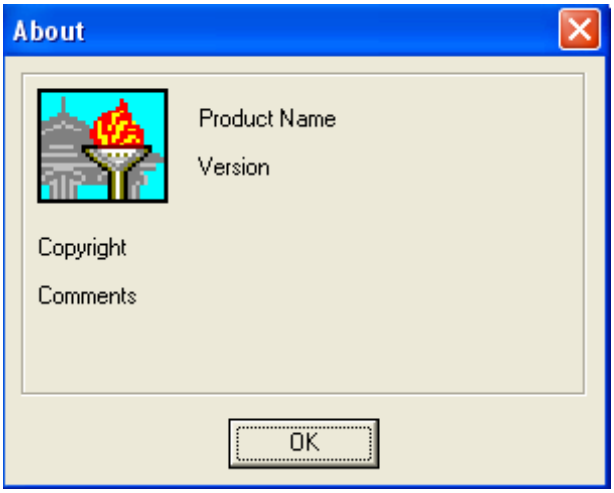



Рисунок 13.1 – Шаблон информационного окна **About**

На форме расположены информационная панель и кнопка **OK** закрытия окна. Свойству **ModalResult** кнопки установлено значение mrOk, поэтому при нажатии на нее форма **AboutBox** автоматически закрывается. При этом обработчик события нажатия кнопки не требуется. На панели находятся четыре надписи **Label** и компонент-образ **Image**. Для отображения в окне сведений о программном продукте и его разработчиках необходимо изменить значения свойств **Caption** компонентов **Label**. В компонент **Image** можно загрузить картинку, поясняющую назначение программы. Название окна также можно изменить.

При необходимости можно программно удалить ненужные элементы на форме или добавить какие-либо управляющие элементы, установив для них соответствующие значения свойств. Как правило, операции с информационным окном и его элементами выполняются при разработке программы через Инспектор объектов. При этом события элементов не используются, и обработчики событий не создаются.

### 13.1.4 Многостраничный блокнот

Многостраничный блокнот **PageControl** (  ), расположенный на странице **Win32**, состоит из несколько страниц, расположенных одна под другой. Каждая страница имеет свою закладку и относительно независима от других страниц. Компактное расположение страниц блокнота позволяет удобно размещать и группировать другие управляющие элементы. При выборе закладки автоматически выбирается и соответствующая страница, после чего пользователю становятся доступными расположенные на ней элементы управления. Отдельные страницы многостраничного блокнота называют также панелями, а сам компонент **PageControl** – множественной панелью. Достоинством данного компонента является экономия экранного пространства, фактически неограниченно увеличивая его «глубину».

Число страниц многостраничного блокнота указывает свойство **PageCount** типа Integer, действующее во время выполнения программы и доступное только для чтения.

Свойство **ActivePage** определяет выбранную закладку, а вместе с ней и страницу, находящуюся сверху. Это свойство доступно для записи и для чтения. При разработке приложения в качестве значения свойства **ActivePage** Инспектор объектов отображает только заголовок активной страницы.

Первоначально компонент, помещенный на форму, будет пустым – не содержащим ни одной страницы. Новая страница добавляется командой **New Page** (Создать страницу) из контекстного меню. При этом в списке объектов в Инспекторе объектов (и в описании класса **TForm**) появляется описывающий ее новый объект **TabSheet1**. Контекстное меню позволяет перейти к следующей или предыдущей странице, но такой переход проще выполнить, щелкнув на нужной закладке.

Для удаления страницы из многостраничного блокнота из контекстного меню выбирается пункт **Delete Page** (Удалить страницу). Перед вызовом контекстного меню должна быть выделена именно удаляемая страница (объект типа **TabSheet**), а не блокнот (объект типа **PageControl**), в противном случае произойдет удаление всего компонента **PageControl**. После переключения страницы выделенным оказывается весь блокнот. Для повторного выделения страницы можно щелкнуть мышью в области страницы (не на закладке) или выбрать страницу через Инспектор объектов.

Название закладки каждой страницы определяется ее свойством **Caption**, а рисунок, который может отображаться на закладке рядом с названием, задается как значение свойства **ImageIndex** типа Integer. Если рисунок, соответствующий указанному номеру, в компоненте **ImageList** не найден или значение свойства **ImageIndex** равно (-1), то для этой страницы блокнота рисунок считается отсутствующим. Список рисунков задается свойством **Images** блокнота. Ширина закладки страницы автоматически изменяется по размеру текста и рисунка, отображаемым на закладке. Свойство **Font** компонента позволяет изменить шрифт надписи, начертание, размер, цвет и другие.

На рисунке 13.2 показан пример использования компонента **PageControl**.

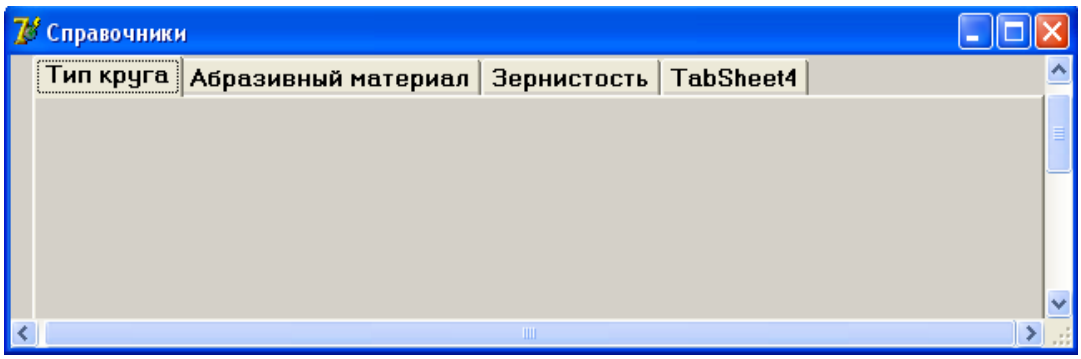



Рисунок 13.2 - Пример использования компонента **PageControl**

### 13.1.5 Использование модифицированной сетки

При работе с реляционными базами данных используются два способа отображения данных: таблицы и формы. В таблицах пользователь может видеть одновременно несколько записей, что облегчает ему контроль за данными и выбор необходимой записи. При использовании формы пользователь в каждый момент видит на экране лишь одну запись, но зато имеет простой доступ к любому ее полю, в том числе многострочному или графическому. В Delphi кроме компонента **DBGrid**, для управления записями таблицы предназначен компонент **DBCtrlGrid** (модифицированная сетка), который сочетает в себе удобства обоих способов: он представляет собой таблицу, каждый ряд или каждая ячейка которой отображается в виде формы.

Компонент **DBCtrlGrid** (  ), расположенный на странице **Data Controls**, представляет собой несколько отдельных панелей, на которых располагаются визуальные компоненты. Компонент показывает содержимое нескольких записей одновременно в однотипных наборах визуальных компонентов. Для работы с компонентом необходимо поместить его на форму, связать с компонентом **DataSource**, который в свою очередь связать с набором данных **Table**.

Модифицированная сетка **DBCtrlGrid** может отображать несколько одинаковых панелей, но текущей является только одна из них. При проектировании приложения визуальные компоненты, например, **DBEdit** или **DBText**, располагаются на одной (верхней) панели. Когда визуальные компоненты помещаются на панель модифицированной сетки, у них автоматически устанавливается нужное значение свойства **DataSource**, взятое из аналогичного свойства модифицированной сетки. При выполнении приложения компоненты, размещенные на одной панели, дублируются на другие панели сетки.

Число панелей, одновременно видимых в модифицированной сетке, определяется свойством **PanelCount** типа Integer, доступным для чтения во время выполнения приложения.

Все панели сетки имеют одинаковые размеры, определяемые свойствами **PanelHeight** (высота) и **PanelWidth** (ширина) типа Integer. Каждая панель может иметь рамку, что определяется свойством **PanelBorder**, принимающим следующие значения:

- gbNone (рамки нет);
- gbRaised (трехмерная приподнятая рамка, по умолчанию).

Число одновременно видимых строк и столбцов сетки задают свойства **RowCount** или **ColCount** типа Integer, значения которых по умолчанию равны трем и одному. Это соответствует трем панелям в одном столбце и наличию вертикальной полосы прокрутки. При задании более одного столбца у сетки появляется горизонтальная полоса прокрутки.

Приемы работы с **DBCtrlGrid** аналогичны приемам работы с **DBGrid**. Текущая строка (ячейка) выделяется пунктирным прямоугольником. Для навигации по сетке компонента используются те же клавиши, что и для **DBGrid**. Для вставки новой записи необходимо нажать на клавиатуре клавишу **Insert** или попытаться перейти с последней записи вниз на одну строку (для сетки с одной колонкой). Для изменения записи достаточно ввести новое значение в какое-либо поле. Для удаления записи необходимо нажать комбинацию клавиш **Ctrl**+**Delete**.

На возможности редактирования, добавления и удаления записей в **DBCtrlGrid** влияет ряд свойств как других компонентов, размещенных на панелях **DBCtrlGrid**, так и самого компонента **DBCtrlGrid**.

### 13.1.6 Формирование условного обозначения шлифовального круга

Рассмотрим использование данных, хранящихся в таблицах базы данных, на примере формирования условного обозначения шлифовального круга.

Шлифование (шлифовка) (от польск. szlifowac - точить - полировать, шлифовать) - обработка поверхностей изделий из различных материалов (металлических, деревянных, стеклянных, керамических и других) абразивным инструментом на шлифовальных станках. Шлифование обеспечивает получение высокой чистоты обработанной поверхности и высокой точности размеров обрабатываемых деталей. Шлифование выполняется абразивными инструментами. Абразивный инструмент представляет собой твердое тело, состоящее из зерен абразивного материала, скрепленных между собой связкой [26].

Абразивные инструменты в подавляющем большинстве используются в виде шлифовальных кругов разнообразной формы. Маркировка круга включает: тип круга, размеры, марку шлифовального материала, зернистость, твердость, наличие упрочняющих элементов, номер структуры, вид связки, рабочую скорость круга, класс неуравновешенности, обозначение нормативно-технической документации. Общая структура маркировки круга показана на рисунке 13.3. В качестве примера на рисунке приведено условное обозначение плоского круга прямого профиля типа 1 с размерами D = 150 мм, T = 20 мм и H = 32 мм из нормального электрокорунда 14A, средней зернистостью F36, степенью твердости P, с номером структуры 5, на бакелитовой связке B.

В качестве примера рассмотрим формирование части маркировки круга. Значение типа шлифовального круга пользователь может выбрать из таблицы «Тип круга» в соответствии с предъявляемыми требованиями и назначением инструмента. Аналогично абразивный материал и его марка могут быть вы-

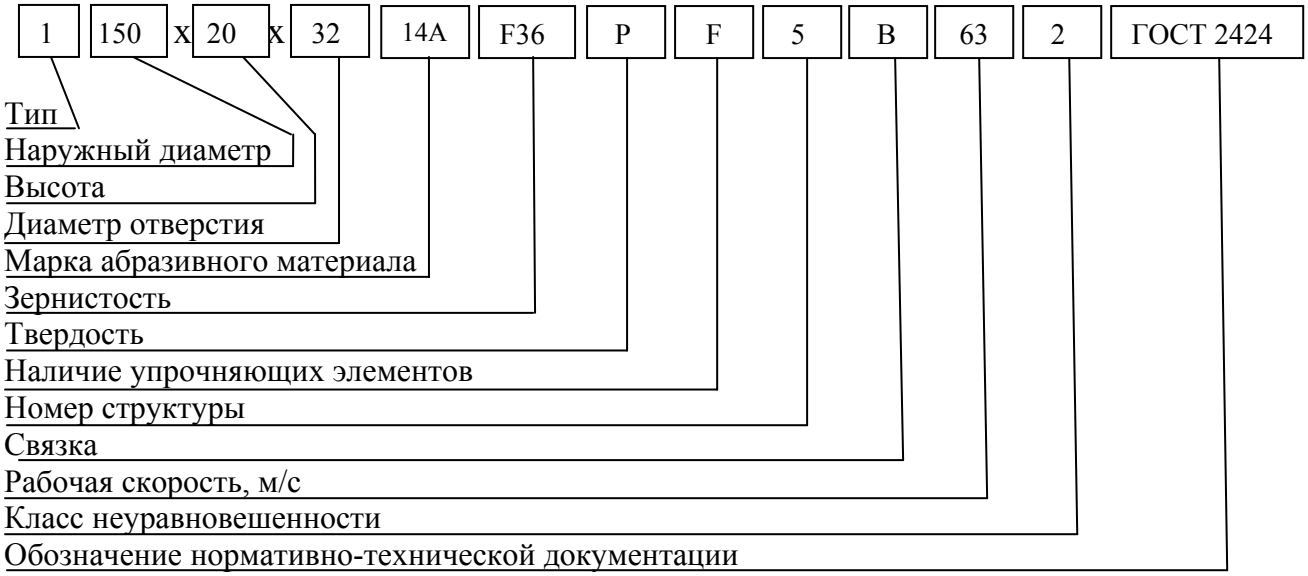



Рисунок 13.3 – Структура условного обозначения круга

браны из таблиц «Физико-механические свойства абразивных материалов» и «Марка абразивного материала». Для отображения значений параметров круга, указанных пользователем, можно использовать компонент **DBText**, который необходимо связать с источником данных **DataSource**.

Например, пусть требуется отображение в компоненте **DBText1**, расположенном на форме **Form6**, текущего значения номера типа (поле «Tip_nomer»), указание которого пользователь осуществляет на форме **Form5**. Программный код кнопки, по нажатию на которую осуществится переход к форме **Form6**, должен содержать следующее:
«Form6.DBText1.DataSource:=Form5.DataSource1;
Form6.DBText1.DataField:='Tip_nomer';».

Отображение значения, введенного в компонент редактирования **Edit1** предыдущей формы **Form5**, можно осуществить при помощи компонента **Label1**, для которого необходимо программно задать изменение значения свойства **Caption**. Например, «Form6.Label1.Caption:=Form5.Edit1.Text;».

Возможны и другие способы реализации в Delphi процедуры формирования условного обозначения шлифовального круга.

### 13.2 Задание на выполнение работы

13.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к данной лабораторной работе. При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его в своём каталоге.

13.2.2 Запустить утилиту **Database Desktop**, установить умалчиваемый псевдоним. Задать структуры таблиц «Тип круга», «Физико-механические свойства абразивных материалов», «Марка абразивного материала», «Зернистость», содержание которых приведено в таблицах 13.1 – 13.4. Для таблиц ис-

Таблица 13.1 – Тип круга

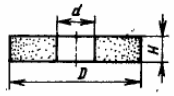
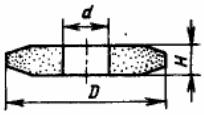
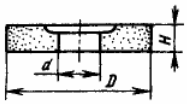
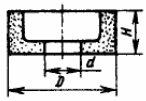
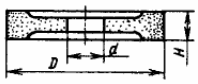
| Номер | Номер типа | Тип | Обозначение | Область применения | Форма |
|---|---|---|---|---|---|
| 1 | 1 | Плоские прямого профиля | ПП | Универсальное применение |  |
| 2 | 4 | Плоские с двусторонним коническим профилем | 2П | Резьбошлифование, шлицешлифование, зубошлифование |  |
| 3 | 5 | Плоские с выточкой | ПВ | Универсальное применение; выточки предназначены для обеспечения доступа круга к обрабатываемой заготовке |  |
| 4 | 6 | Чашечные цилиндрические | ЧЦ | Заточка и доводка режущего лезвийного инструмента, внутреннее и плоское шлифование |  |
| 5 | 7 | Плоские с двумя выточками | ПВД | Круглое наружное и внутреннее шлифование |  |

Таблица 11.2 – Физико-механические свойства абразивных материалов

| Код типа | Материал | Микротвердость, ГПа | Механическая прочность, Н | Абразивная способность зерна, г |
|---|---|---|---|---|
| 1 | Нормальный электрокорунд | 18,9 – 19,6 | 8,8 – 10,7 | 0,145 |
| 2 | Белый электрокорунд | 19,6 – 24,5 | 8,8 – 10,4 | 0,155 |
| 3 | Карбид кремния черный | 32,4 – 39,3 | 11,0 – 14,7 | 0,4 |
| 4 | Монокорунд | 22,6 – 23,5 | 11,7 – 13,7 | 0,150 |

Таблица 13.3 – Марка абразивного материала

| Код марки | Марка материала | Код типа материала | Область применения |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 13А | 1 | Обдирочное шлифование стальных заготовок инструментом на органических связках |
| 2 | 14А | 1 | Шлифование стальных заготовок инструментом на органических и неорганических связках |
| 3 | 15А | 1 | Шлифование стальных заготовок инструментом на органической связке |
| 4 | 23А | 2 | Шлифование стальных заготовок инструментом на органической связке |

Продолжение таблицы 13.3

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | 24 А | 2 | Шлифование закаленных заготовок |
| 6 | 25А | 2 | Скоростное шлифование, доводка стальных закаленных заготовок инструментом на керамических связках |
| 7 | 53С | 3 | Шлифование заготовок из чугуна, цветных металлов на всех связках |
| 8 | 54С | 3 | Шлифование заготовок из чугуна, цветных металлов на всех связках |
| 9 | 43А | 4 | Шлифование и заточка труднообрабатываемых сталей и сплавов инструментом на керамических связках |
| 10 | 45А | 4 | Для абразивного прецизионного инструмента на керамической связке. Отделка и доводка |

Таблица 13.4 – Зернистость

| Код записи | Зернистость | Рекомендации |
|---|---|---|
| 1 | F180 – F220 | При отделочном шлифовании металлов, стекла, мрамора, резьбошлифовании |
| 2 | F100 – F180 | При отделочном шлифовании, доводке твердых сплавов |
| 3 | F54 – F90 | При чистовом шлифовании, обработке профильных поверхностей |
| 4 | F30 – F60 | При предварительном и комбинированном шлифовании |
| 5 | F24 – F36 | При плоском шлифовании, заточке средних и крупных резцов |
| 6 | F16 – F24 | При обдирочных операциях, при зачистке отливок, поковок |
| 7 | F12 – F4 | При обдирочном силовом шлифовании |

пользовать тип Paradox 7. Сохранить созданные структуры с заданным ранее псевдонимом в соответствующих файлах.

13.2.3 Установить связь между таблицами «Физико-механические свойства абразивных материалов» и «Марка абразивного материала» (рисунок 13.4).

13.2.4 Создать экранную форму **Form1** для главной формы. На ней разместить две кнопки **Button** (  ) со страницы **Standart**: для перехода на форму **Form4** в режим «Справочники», для перехода на форму **Form5** в режим «Формирование обозначения круга». Со страницы **Additional** разместить две кнопки с рисунком **BitBtn** (  ): для перехода в информационное окно **AboutBox** «О программе» и для выхода из приложения (Close).

Задать для формы значение свойства **Caption**, равное «Главная форма». Для созданных на главной форме **Form1** кнопок **Button1**, **Button2**, **BitBtn1**,

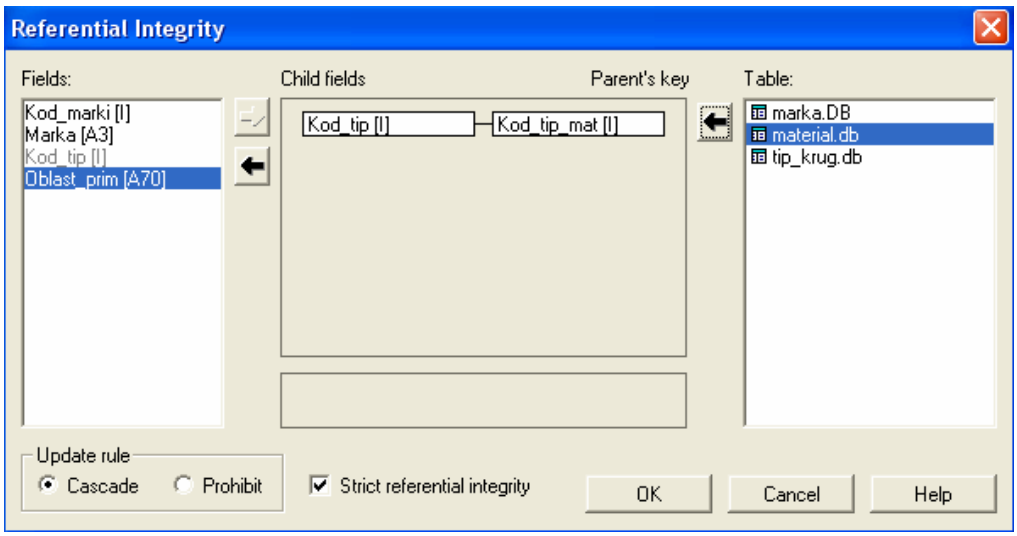



Рисунок 13.4 - Установление связи между таблицами «Физико-механические свойства абразивных материалов» и «Марка абразивного материала»

**BitBtn2** задать значения свойств в соответствии с таблицей 13.5 (программные коды пока не задавать!). При необходимости для текста, размещаемого на кнопках, изменить шрифт, размер и начертание. Пример вида главной формы показан на рисунке 13.5.

13.2.5 Создать экранную форму **Form2** для заставки приложения, для этого выполнить команду **File => New => Form**. Разместить на ней компоненты **Image** ( ), **Label** ( ), **Memo** ( ), **Timer** ( ) задать для них необходимые свойства. Для главной формы **Form1** и заставки **Form2** задать значения свойств и обработчиков событий в соответствии с этапами, описанными в пункте 13.1.2.

Проверить, чтобы для формы **Form2** программный код после задания значений свойств и обработчиков событий имел вид:

«procedure TForm2.Timer1Timer(Sender: TObject);
begin
close;
end;
procedure TForm2.FormKeyPress(Sender: TObject; var Key: Char);
begin
close;
end;
procedure TForm2.FormClick(Sender: TObject);
begin
close;
end;».

Сохранить изменения в приложении, протестировать его работу. Пример экранной формы заставки приложения приведен на рисунке 13.6.

Таблица 13.5 – Значения свойств и программные коды для кнопок на главной форме **Form1**

| Значение свойства **Name** | Действие | Значение свойств | Программный код |
|---|---|---|---|
| **BitBtn1** | Переход к форме **AboutBox**, являющейся информационным окном «О программе» | **Caption**: О программе<br>**Glyph**: (указывается путь к файлу с изображением, помещаемым на кнопку)<br>**Kind**: bkCuston | **procedure** TForm1.BitBtn1Click(Sender: TObject);<br>**begin**<br>AboutBox.show;<br>**end;** |
| **Button1** | Переход к форме **Form4** для работы со справочниками | **Caption**: Справочники | procedure TForm1.Button1Click(Sender: TObject);<br>begin<br>form4.show;<br>end; |
| **Button2** | Переход к форме **Form5** для формирования обозначения круга | **Caption**: Формирование обозначения круга | procedure TForm1.Button2Click(Sender: TObject);<br>begin<br>form5.show;<br>end; |
| **BitBtn2** | Закрытие приложения | **Caption**: &Выход из приложения<br>**Kind**: bkClose | |

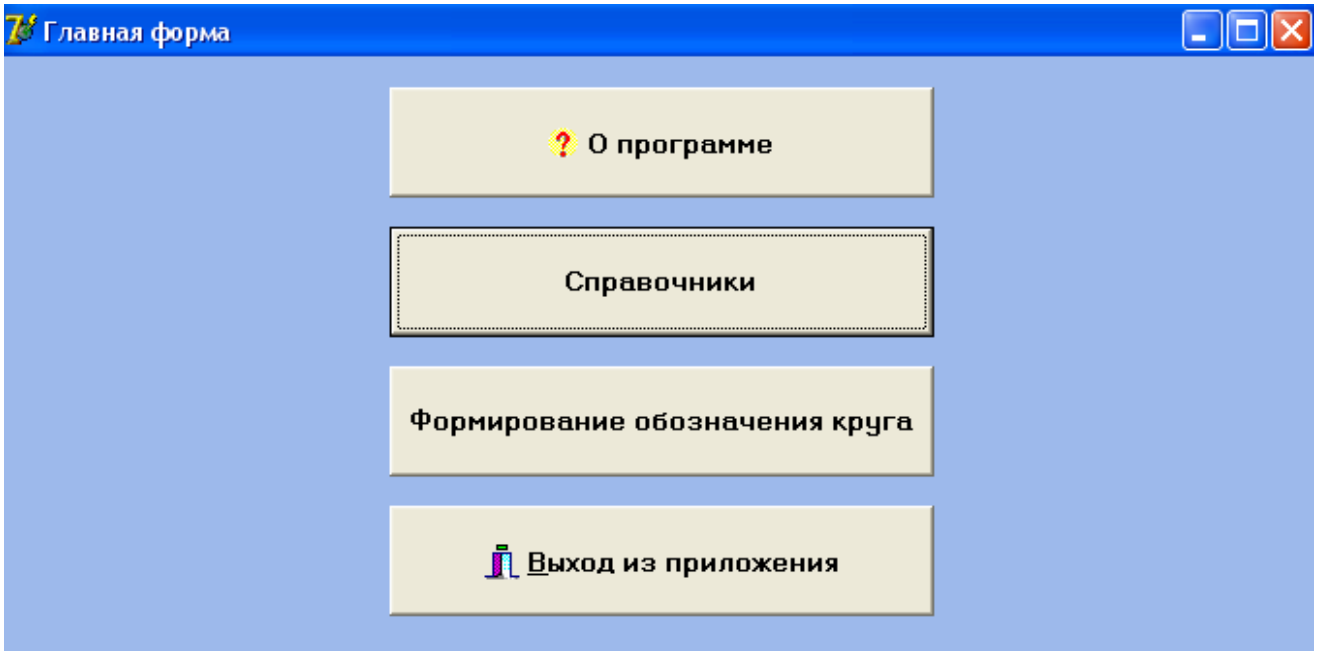



Рисунок 13.5 - Пример вида главной формы

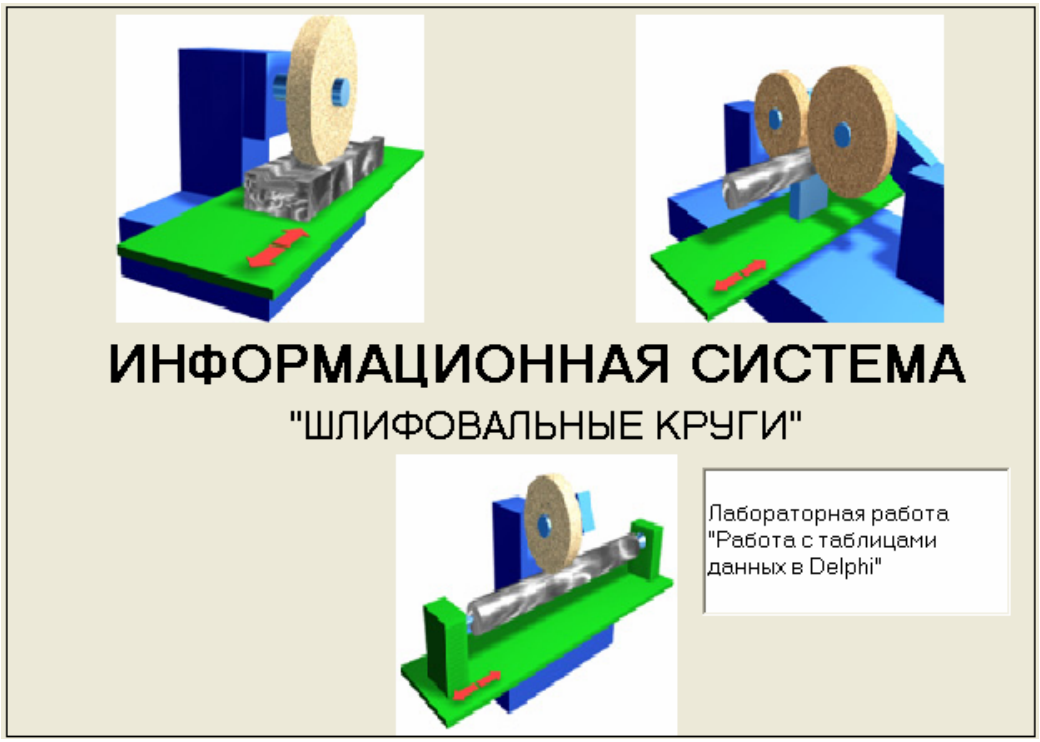



Рисунок 13.6 - Пример заставки

13.2.6 Создать экранную форму **AboutBox** для перехода в информационное окно «О программе», выполняя этапы в соответствии с пунктом 13.1.3, ввести необходимые сведения о программе.

13.2.7 Создать командой **File => New => Form** экранные формы: **Form4** для работы со справочниками, **Form5** для формирования обозначения круга. Задать для форм соответствующие значения свойства **Caption**.

13.2.8 Задать для созданных на главной форме **Form1** кнопок **Button1**, **Button2**, **BitBtn1**, **BitBtn2** программные коды (таблица 13.5). Сохранить изменения, протестировать работу приложения.

13.2.9 Поместить на форму **Form4** «Справочники» многостраничный блокнот **PageControl** ( ) с закладками «Тип круга», «Абразивный материал», «Зернистость», аналогично показанному на рисунке 13.2. С помощью свойства **Font** компонентов **TabSheet** изменить шрифт надписей, начертание, размер, цвет и другие.

13.2.10 На закладке «Тип круга» разместить следующие компоненты:

- набор данных **Table** ( ) со страницы **BDE** для установления связи приложения с таблицей базы данных «Тип круга» (задать свойствам **DatabaseName** псевдоним базы данных, **TableName** – имя таблицы, **Active** – значение True);

- источник данных **DataSource** ( ) со страницы **Data Access** для соединения компонента **Table** с компонентами, отображающими данные (свойству **DataSet** задать значение Table1);

- четыре метки **Label** (  ) со страницы **Standart** для задания имен полей таблицы «Тип круга» (свойству **Caption** компонентов присваиваются значения «Номер», «Тип», «Обозначение» или «Область применения»);

- пять полей редактирования **DBEdit** (  **)** со страницы **Data Controls** для отображения значений полей «Номер», «Тип», «Номер типа», «Обозначение» и «Область применения» (свойству **DataSourse** задать значение DataSourse1, для свойства **DataField** выбрать из раскрывающегося списка имя соответствующего поля);

- графическое изображение **DBImage** (  ) со страницы **Data Controls** для вывода графического изображения, содержащегося в графическом поле «Форма» (свойству **DataSourse** задать значение DataSourse1; для свойства **DataField** выбрать из раскрывающегося списка имя соответствующего поля; свойству **Stretch** задать значение True, чтобы выводимый рисунок пропорционально растягивался или уменьшался до размеров компонента);

- компонент **OpenPictureDialog** (  ) со страницы **Dialogs** для появления окна диалога выбора файла с изображением для открытия (при работе приложения после выбора нужного файла формата BMP, ICO, EMF, JPG, JPEG, WMF содержащееся в нем изображение загружается в компонент **DBImage**);

- кнопку **Button** (  ) со страницы **Standart** для открытия окна диалога, позволяющего вставлять изображение из файла (свойству **Caption** задать соответствующее значение); программный код для кнопки должен содержать следующую процедуру

«procedure TForm4.Button1Click(Sender: TObject);
begin
if Form4.OpenPictureDialog1.Execute then
Form4.DBImage1.Picture.LoadFromFile(Form4.OpenPictureDialog1.FileName);
end;»;

- навигатор **DBNavigator (**  **)** со страницы **Data Controls** (задать значение свойства **DataSourse**, равное DataSourse1; для отображения подсказок присвоить значение True свойству **ShowHint**; заменить текст подсказок с английского языка на русский, вызвав Строковый редактор String list editor в свойстве **Hints**).

Запустить приложение. Заполнить записями из таблицы 13.1 «Тип круга». Посмотреть работу навигатора. Пример вида справочника «Тип круга» показан на рисунке 13.7.

13.2.11 На закладке «Абразивный материал» разместить компоненты **Table2**, **DataSource2**, **DBGrid1** (  ), **DBNavigator2**, задать значения их свойств для отображения таблицы «Физико-механические свойства абразивных материалов» и ее управления. Используя Editing DBGrid1.Columns, изменить имена столбцов на русские названия (для этого для каждого столбца раскрыть свойство **Title**, заменить значение в строке **Caption**), расположение значений полей и имен полей (свойству **Alignment** задать значение taCenter, если требуется расположение по центру). Запустить приложение. Заполнить записями из таблицы

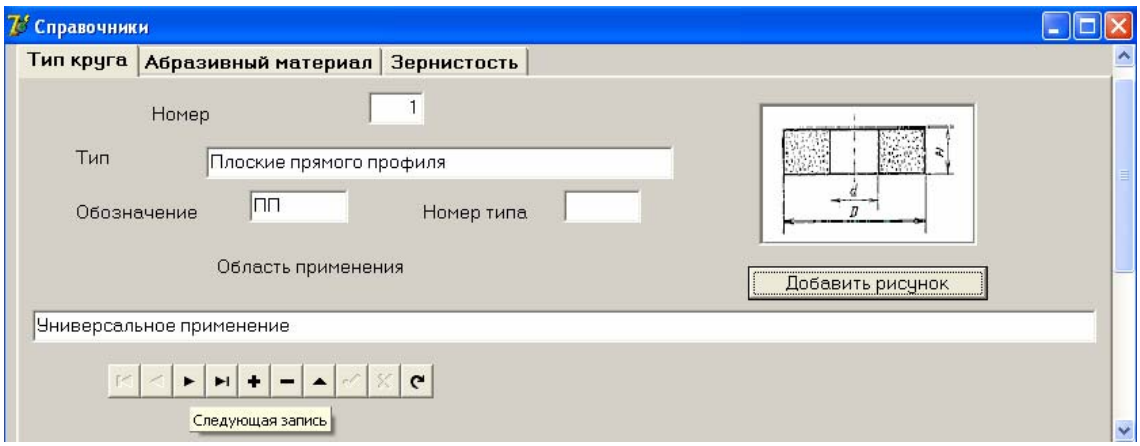



Рисунок 13.7 – Пример вида справочника «Тип круга»

13.2 «Физико-механические свойства абразивных материалов».

13.2.12 На закладке «Абразивный материал» разместить компоненты **Table3**, **DataSource3**, **DBCtrlGrid1** ( ), задать значения их свойств для работы с данными из таблицы «Марка абразивного материала».

Чтобы содержимое подчиненного набора соответствовало выбору записи в главном наборе, дочернюю таблицу «Марка абразивного материала» нужно связать с родительской «Физико-механические свойства абразивных материалов». Для этого выполнить следующие действия:

- активизировать компонент **Table3**;
- в окне Инспектора Объектов раскрыть список выбора в свойстве **MasterSource**, выбрать имеющееся в нем значение **DataSource2**;
- щелкнуть по правой части строки **MasterFields**, затем по появившейся в ней кнопке с тремя точками, чтобы раскрыть окно редактора связей;
- раскрыть список **Available Indexes** в верхней части окна и выбрать индекс, соответствующий внешнему ключу дочерней таблицы, в результате в окошке **Detail Fields** появится имя этого поля связи, щелкнуть по нему;
- выбрать это же поле в окошке **Master Fields**, щелкнуть по нему;
- нажать кнопку **Add**, связь между таблицами будет установлена, в окошке **Joined Fields** появится схема связи по имени поля;
- закрыть окно редактора связей кнопкой **OK**.

13.2.13 На компоненте **DBCtrlGrid1** разместить необходимые визуальные компоненты для работы с полями базы данных – **DBEdit**, **DBMemo** ( ), **Label**, задать их значения. Пример работы с **DBCtrlGrid** на этапе конструирования приложения показан на рисунке 13.8.

Запустить приложение. Заполнить записями из таблицы 13.3 «Марка абразивного материала». Для вставки новой записи достаточно нажать комбинацию клавиш **Ctrl+Insert**, при условии, что свойство **AllowInsert** находится в состоянии True. Если свойство **AllowDelete** установлено в True, пользователь может удалить текущую запись, нажав комбинацию клавиш **Ctrl+Delete**. Вид компонента **DBCtrlGrid** при выполнении приложения приведен на рисунке 13.9.

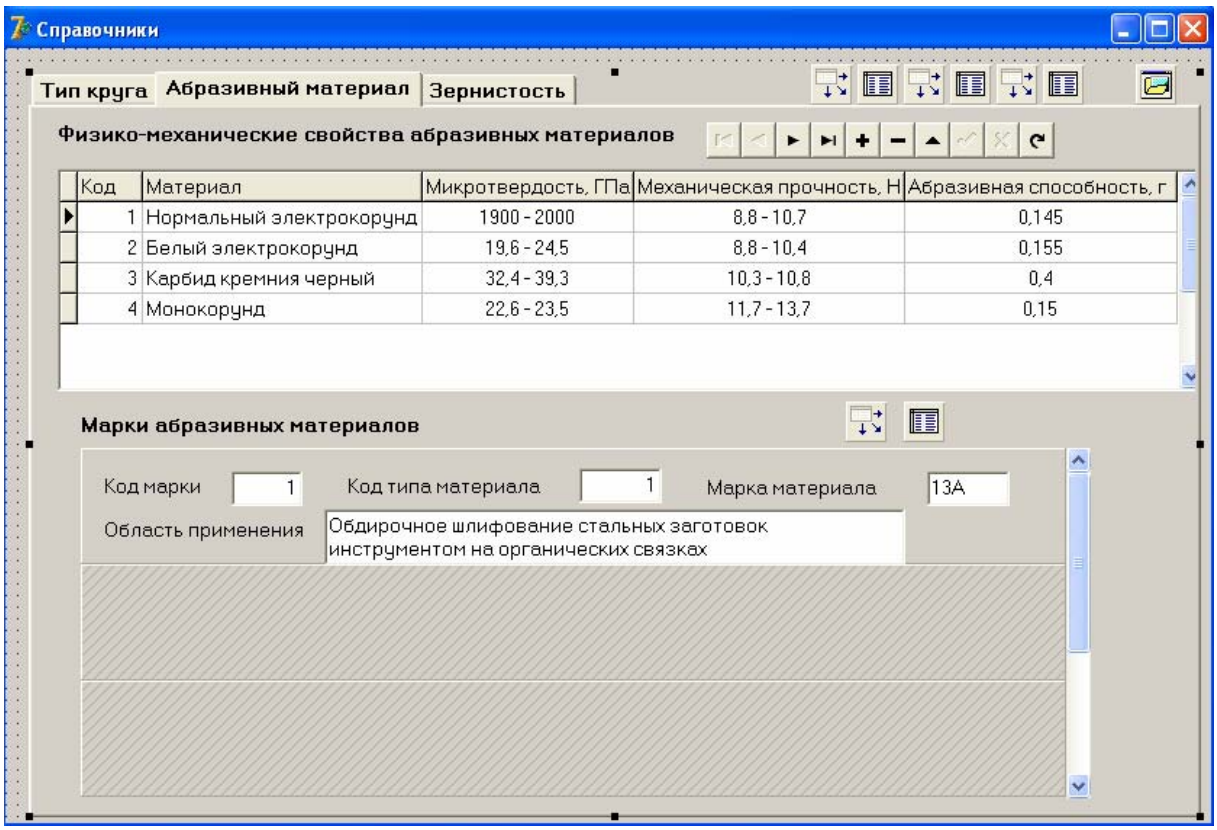



Рисунок 13.8 - Пример работы с **DBCtrlGrid** на этапе конструирования приложения

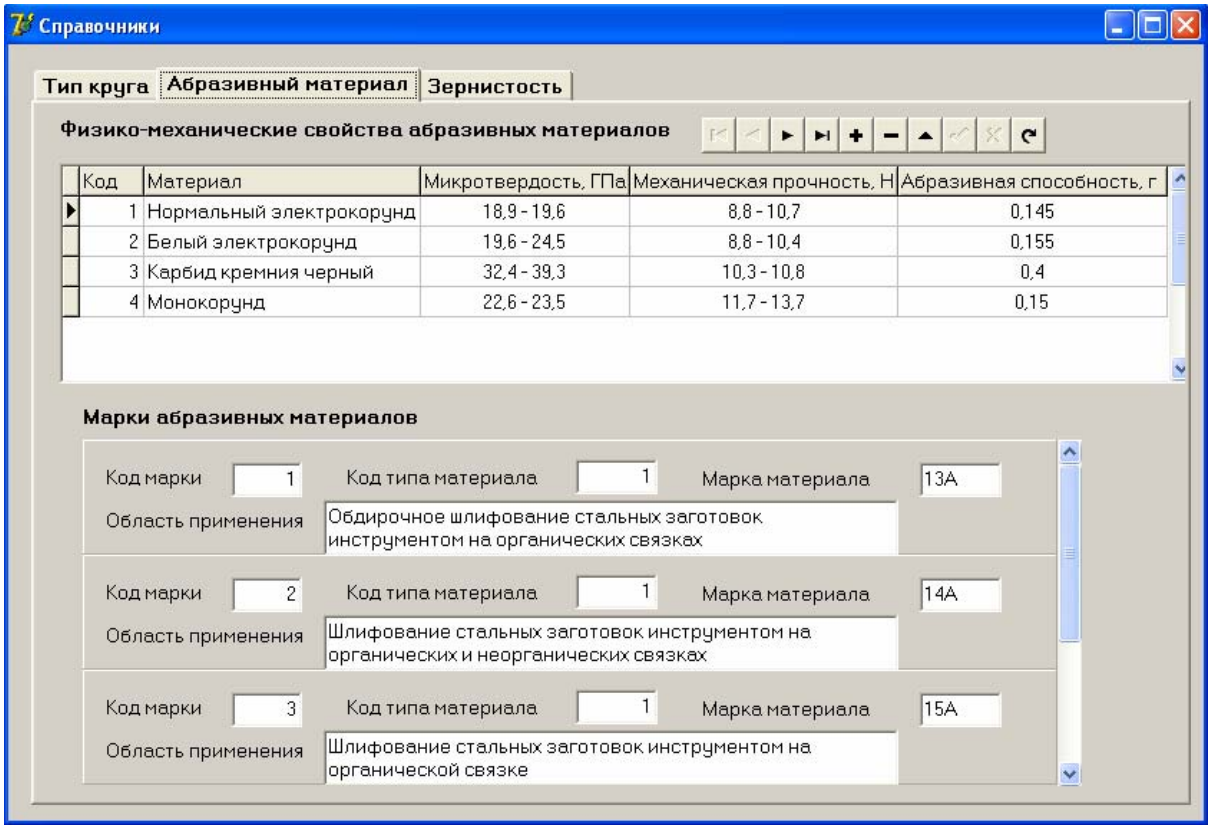



Рисунок 13.9 – Вид компонента **DBCtrlGrid** при выполнении приложения

13.2.14 На закладке «Зернистость» разместить компоненты **Table4**, **DataSource4**, **DBGrid2**, **DBNavigator3**, задать значения их свойств для отображения таблицы 13.4 «Зернистость» и ее управления. Запустить приложение. Заполнить записями таблицу «Зернистость». Пример вида закладки «Зернистость» показан на рисунке 13.10.

13.2.15 На форме **Form5** разместить компоненты для ввода данных при формировании обозначения шлифовального круга:

- четыре прямоугольные рамки с заголовком **GroupBox** ( ) со страницы **Standart** для объединения компонентов, относящихся к одному этапу процесса формирования обозначения (задать значения свойств **Caption** и **Font** для каждого компонента, аналогично тому, как показано на рисунке 13.11);

- на прямоугольной рамке **GroupBox1** набор данных **Table1**, источник данных **DataSource1**, сетку **DBGrid1** (задать значения их свойств для отображения данных из таблицы «Тип круга»);

- четыре метки **Label1** – **Label4** (задать значения их свойства **Caption**) и три компонента редактирования **Edit1** – **Edit3** ( ) на прямоугольной рамке **GroupBox2** для ввода значений размеров круга;

- на прямоугольной рамке **GroupBox3** метку **Label5**, набор данных **Table2**, источник данных **DataSource2**, сетку **DBGrid2** (задать значения их свойств для отображения данных из таблицы «Физико-механические свойства материалов»);

- на прямоугольной рамке **GroupBox3** метку **Label6**, набор данных **Table3**, источник данных **DataSource3**, модифицированную сетку **DBCtrlGrid1** (задать значения их свойств для отображения данных из таблицы «Марка материала», связать с главной таблицей);

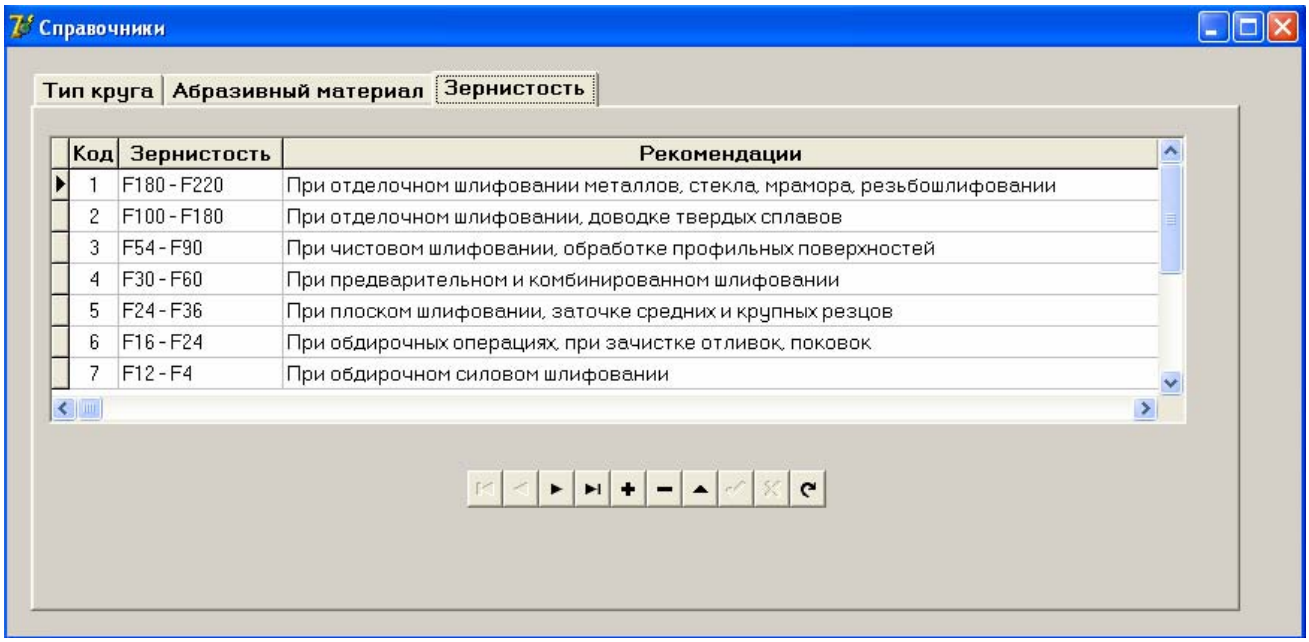



Рисунок 13.10 - Пример вида формы «Справочники» на закладке «Зернистость»

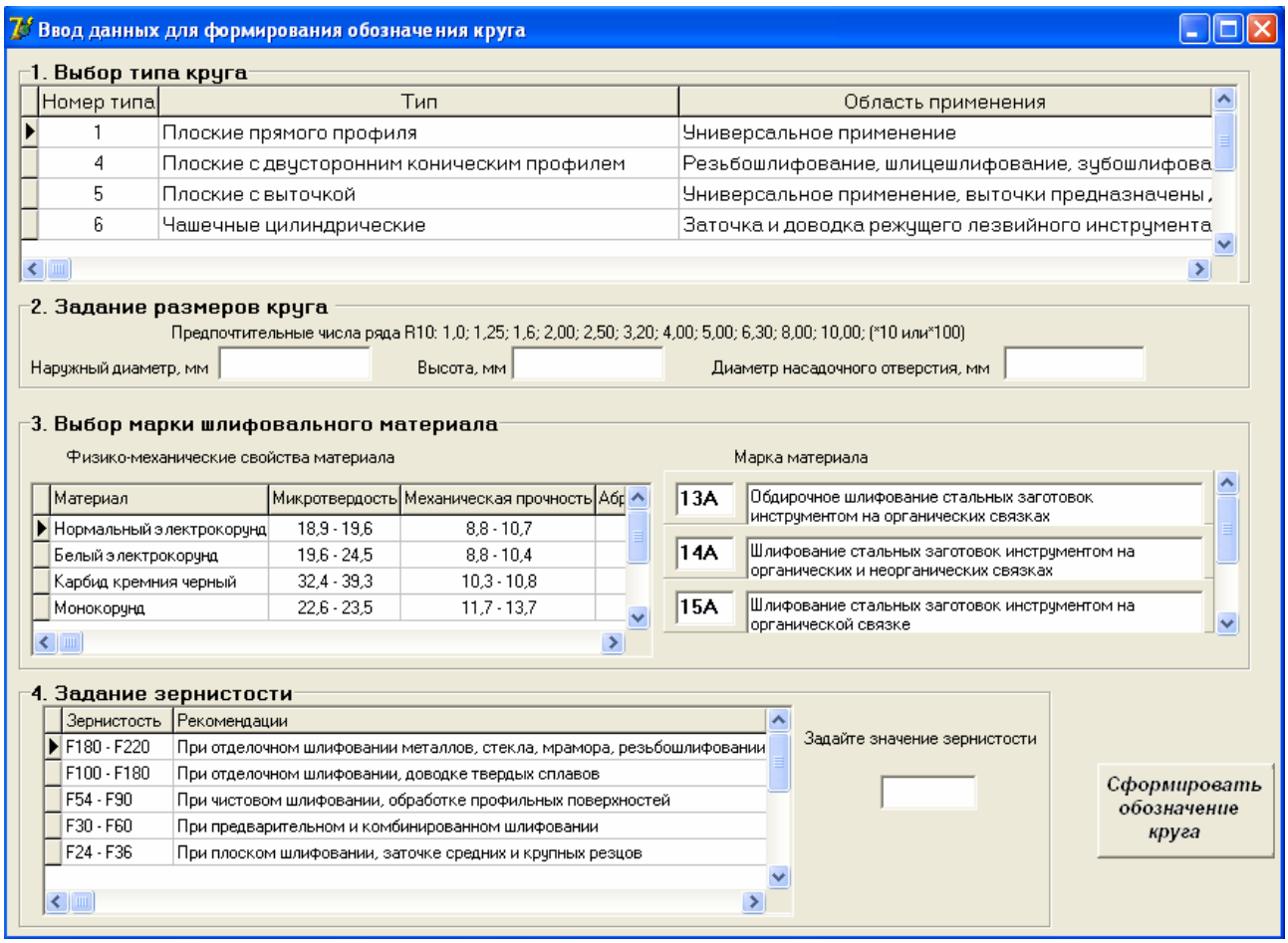



Рисунок 13.11 - Пример вида формы **Form5**

- набор данных **Table4**, источник данных **DataSource4**, сетку **DBGrid3** (задать значения их свойств для отображения данных из таблицы «Зернистость»); метку **Label7**, компонент редактирования **Edit4** на прямоугольной рамке **GroupBox4**;

- кнопку **Button1** или **BitBtn1** для перехода на форму **Form6**, на которой будет отображаться обозначение шлифовального круга.

Задать запрет на редактирование записей в используемых на форме наборах данных. Для этого изменить значения свойств компонентов **Table1** - **Table4**: свойству **ReadOnly** установить значение True, затем свойству **Active** установить значение True. Запустить приложение.

13.2.16 Добавить в программный код кнопки **Button2** «Формирование обозначения круга», расположенной на главной форме **Form1**, следующий текст:

«form5.DBGrid1.Columns[0].Visible:=False;
form5.DBGrid1.Columns[3].Visible:=False;
form5.DBGrid2.Columns[0].Visible:=False;
form5.DBGrid3.Columns[0].Visible:=False;».

При запуске приложения не будут отображаться первое и пятое поле компонента **DBGrid1**, первые поля компонентов **DBGrid2** и **DBGrid3**. Запус-

тить приложение, протестировать его работу. Откорректировать, если требуется, размеры визуальных компонентов.

13.2.17 Создать форму **Form6**. На ней разместить текстовые метки **Label1 - Label19** и компоненты статического текста **DBText1** – **DBText6**. Назначение компонентов приведено в таблице 13.6.

Таблица 13.6 – Назначение компонентов на форме **Form6**

| Имя компонента | Назначение | Значение свойства **Caption** |
|---|---|---|
| **Label1** | Заголовок | Условное обозначение шлифовального круга |
| **DBText1** | Значение номера типа круга | |
| **Label2**<br>**Label14** | Значение наружного диаметра круга | |
| **Label3**<br>**Label5** | Знак умножения | x |
| **Label4**<br>**Label15** | Значение высоты круга | |
| **Label6**<br>**Label16** | Значение диаметра насадного отверстия | |
| **DBText2** | Значение марки абразивного материала | |
| **Label7**<br>**Label19** | Значение зернистости | |
| **Label8** | Заголовок | Расшифровка характеристик круга |
| **Label9** | Имя параметра круга | Тип: |
| **DBText3** | Значение типа круга | |
| **DBText4** | Значение обозначения типа круга | |
| **Label10** | Надпись | Размеры: |
| **Label11** | Имя параметра круга | - наружный диаметр, мм |
| **Label12** | Имя параметра круга | - высота, мм |
| **Label13** | Имя параметра круга | - диаметр насадочного отверстия, мм |
| **Label17** | Имя параметра круга | Абразивный материал: |
| **DBText5** | Значение абразивного материала | |
| **DBText6** | Значение марки абразивного материала | |
| **Label18** | Имя параметра круга | Зернистость: |
| **Label19** | Значение зернистости | |

13.2.18 Для кнопки **BitBtn1** «Сформировать обозначение круга», размещенной на форме **Form5**, задать следующий программный код:
«procedure TForm5.BitBtn1Click(Sender: TObject);

```
begin
Form6.show;
Form6.DBText1.DataSource:=Form5.DataSource1;
Form6.DBText1.DataField:='Tip_nomer';
Form6.Label2.Caption:=Form5.Edit1.Text;
Form6.Label4.Caption:=Form5.Edit2.Text;
Form6.Label6.Caption:=Form5.Edit3.Text;
Form6.Label7.Caption:=Form5.Edit4.Text;
Form6.Label14.Caption:=Form5.Edit1.Text;
Form6.Label15.Caption:=Form5.Edit2.Text;
Form6.Label16.Caption:=Form5.Edit3.Text;
Form6.Label19.Caption:=Form5.Edit4.Text;
Form6.DBText2.DataSource:=Form5.DataSource3;
Form6.DBText2.DataField:='Marka';
Form6.DBText3.DataSource:=Form5.DataSource1;
Form6.DBText3.DataField:='Tip';
Form6.DBText4.DataSource:=Form5.DataSource1;
Form6.DBText4.DataField:='Tip_nomer';
Form6.DBText5.DataSource:=Form5.DataSource2;
Form6.DBText5.DataField:='Material';
Form6.DBText6.DataSource:=Form5.DataSource3;
Form6.DBText6.DataField:='Marka';
end;».
```

Если задать характеристики шлифовального круга на форме **Form5** «Ввод данных для формирования обозначения круга» и нажать кнопку «Сформировать обозначение круга», то на форме **Form6** «Условное обозначение шлифовального круга» осуществится формирование обозначения круга. Вид формы **Form6** приведен на рисунке 13.12.

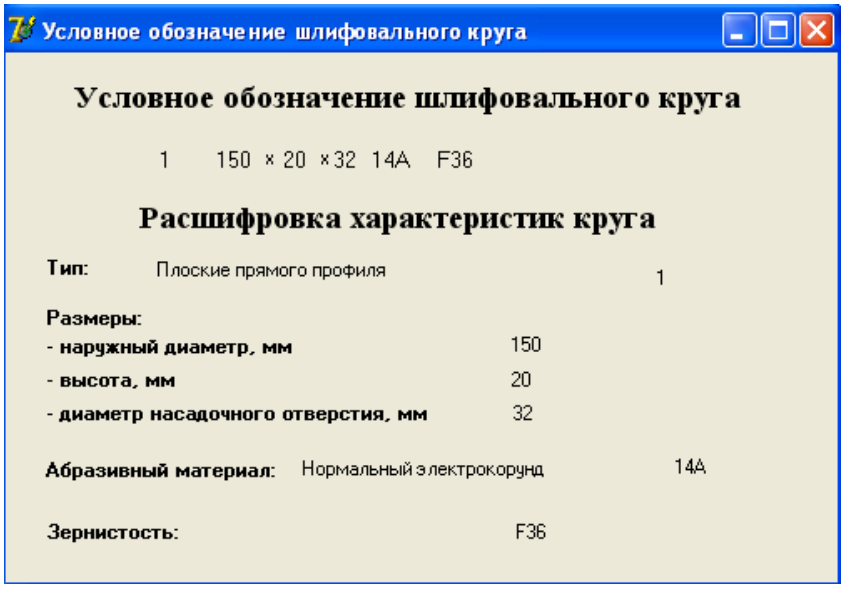



Рисунок 13.12 – Вид формы **Form6**

13.2.19 Запустить созданное приложение, протестировать работу всех его форм.

13.2.20 Оформить отчет по лабораторной работе.

### 13.3 Содержание отчёта

13.3.1 Название работы.

13.3.2 Цель работы.

13.3.3 Структуры таблиц «Тип круга», «Абразивный материал», «Марка абразивного материала», «Зернистость».

13.3.4 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

13.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

### 13.4 Тесты и контрольные вопросы

13.4.1 На какой странице расположен компонент **BitBtn**:

а) **Data Controls**; б) **Data Access**; в) **Standard**;

г) **Additional**; д) **BDE**?

13.4.2 Какое свойство определяет растровое изображение кнопки с картинкой **BitBtn**:

а) **Glyph**; б) **Caption**; в) **Picture**;

г) **Kind**; д) **Font**?

13.4.3 Какое свойство кнопки с картинкой **BitBtn** позволяет задать вид кнопки:

а) **Glyph**; б) **Caption**; в) **Picture**;

г) **Kind**; д) **Font**?

13.4.4 Что такое «глиф»:

а) текст, отображаемый на кнопке с рисунком **BitBtn**;

б) растровый рисунок, отображаемый на кнопке **BitBtn**;

в) векторный рисунок, отображаемый на кнопке **BitBtn**;

г) растровый рисунок в формате BMP?

13.4.5 На какой странице расположен компонент **Timer** :

а) **Data Controls**; б) **Data Access**; в) **Standard**;

г) **Additional**; д) **BDE**?

13.4.6 Какой компонент имеет пиктограмму  :

а) **BitBtn**; б) **Timer**; в) **Label**;

г) **Memo**; д) **Button**?

13.4.7 Какая пиктограмма используется для задания многостраничного блокнота **PageControl**:

а)  ; б)  ; в)  ; г)  ; д)  ?

13.4.8 Чем является объект **TabSheet** в многостраничном блокноте **PageControl**:

а) сеткой; б) страницей; в) закладкой;
г) панелью; д) источником данных?

13.4.9 Какой компонент сочетает в себе удобства обоих способов управления записями, представленных в виде таблицы и формы:

а) **Table**; б) **DBCtrlGrid**; в) **DBGrid**;
г) **DBEdit**; д) **Label**?

13.4.10 Какое свойство компонента **DBCtrlGrid** задает число одновременно видимых панелей:

а) **RowCount**; б) **ColCount**; в) **PanelCount**;
г) **PanelHeight**; д) **PanelWidth**?

13.4.11 Для чего используется кнопка **BitBtn**?

13.4.12 Какое свойство кнопки **BitBtn** позволяет задать количество подключаемых пиктограммм?

13.4.13 Какое свойство определяет растровое изображение кнопки?

13.4.14 В каком формате должно быть сохранено растровое изображение?

13.4.15 Какое свойство кнопки **BitBtn** задает предопределенный вид отображаемого глифа?

13.4.16 Какое свойство позволяет задать текст, отображаемый на кнопке **BitBtn**?

13.4.17 Что представляет собой заставка?

13.4.18 Какие элементы можно поместить на заставку?

13.4.19 Что можно задать при помощи компонента **Timer**?

13.4.20 Для чего предназначен компонент **Image**?

13.4.21 Какому свойству компонента **Image** необходимо установить значение True, чтобы изображение пропорционально растягивалось или уменьшалось до размеров компонента?

13.4.22 При помощи какого свойства компонента **Image** осуществляется визуализация изображения?

13.4.23 Какое свойство компонентов **Label** и **Memo** позволяет изменить шрифт надписи, начертание, размер, цвет?

13.4.24 Для чего предназначен компонент **Memo**?

13.4.25 Как можно задать время отображения заставки?

13.4.26 Каким образом можно создать шаблон формы **About box**?

13.4.27 Что представляет собой шаблон формы **About box**?

13.4.28 Что отображается с помощью компонента **PageControl**?

13.4.29 Какое свойство компонента **PageControl** указывает число страниц многостраничного блокнота?

13.4.30 Каким свойством определяется название закладки каждой страницы многостраничного блокнота?

13.4.31 Для чего предназначен компонент **DBCtrlGrid**?

13.4.32 Что представляет собой компонент **DBCtrlGrid**?

13.4.33 Какие свойства компонента **DBCtrlGrid** задают число одновременно видимых строк и столбцов сетки?

13.4.34 Для чего можно использовать компонент **DBText**?

13.4.35 Для чего можно использовать компонент **DBEdit**?

# 14 Создание базы многомерных данных в Delphi

Целью работы является создание в системе программирования Delphi приложения по работе с базой многомерных данных, которое можно использовать в системах принятия решений.

## 14.1 Общие положения

### 14.1.1 Понятие многомерных данных

Хранимые в базе данные имеют определенную логическую структуру, иными словами, описываются некоторой *моделью представления данных* (моделью данных), поддерживаемой СУБД. В системах принятия решений используются многомерные данные.

Многомерный подход к представлению данных в базе появился практически одновременно с реляционным, но реально работающих многомерных СУБД (МСУБД) до настоящего времени было очень мало. С середины девяностых годов прошлого столетия интерес к ним стал приобретать массовый характер. Толчком послужила в 1993 году программная статья одного из основоположников реляционного подхода Э. Кодда. В ней сформулированы 12 основных требований к системам класса OLAP (OnLine Analytical Processing - оперативная аналитическая обработка), важнейшие из которых связаны с возможностями концептуального представления и обработки многомерных данных. Многомерные системы позволяют оперативно обрабатывать информацию для проведения анализа и принятия решения.

Многомерность модели данных означает не многомерность визуализации цифровых данных, а многомерное логическое представление структуры информации при описании и в операциях манипулирования данными [9, 12]. По сравнению с реляционной моделью многомерная организация данных обладает более высокой *наглядностью* и *информативностью.* Для иллюстрации на рисунке 14.1 приведены реляционное (а) и многомерное (б) представления одних и тех же данных об объемах продаж технических средств.

Если речь идет о многомерной модели с мерностью больше двух, то не обязательно визуально информация представляется в виде многомерных объектов (трех, четырех и более мерных гиперкубов). Пользователю и в этих случаях более удобно иметь дело с двухмерными таблицами или графиками. Данные при этом представляют собой «вырезки» (точнее, «срезы») из многомерного хранилища данных, выполненных с разной степенью детализации.

Рассмотрим основные понятия многомерных моделей данных, к числу которых относятся измерение и ячейка.

*Измерение* (Dimension) — это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются «Дни», «Месяцы», «Кварталы» и «Годы».

В качестве географических измерений широко употребляются «Города», «Районы», «Регионы» и «Страны». В многомерной модели данных изме-

| Товар | Месяц | Объём |
|---|---|---|
| Монитор | июнь | 12 |
| Монитор | июль | 24 |
| Монитор | август | 5 |
| Принтер | июнь | 2 |
| Принтер | июль | 18 |
| Плоттер | июль | 19 |

а

| Товар | Июнь | Июль | Август |
|---|---|---|---|
| Монитор | 12 | 24 | 5 |
| Принтер | 2 | 18 | No |
| Плоттер | No | 19 | No |

б

а – реляционное представление; б – многомерное представление
Рисунок 14.1 - Реляционное и многомерное представление данных

рения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

*Ячейка* (Cell) или показатель - это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой. В зависимости от того, как формируются значения некоторой ячейки, обычно она может быть переменной (значения изменяются и могут быть загружены из внешнего источника данных или сформированы программно), либо формулой (значения, подобно формульным ячейкам электронных таблиц, вычисляются по заранее заданным формулам).

В примере на рисунке 14.2 каждое значение ячейки «Объем продаж» однозначно определяется комбинацией временного измерения («Месяц продаж») и «Тип товара». На практике зачастую требуется большее количество измерений. В примере трехмерной модели данных, приведенной на рисунке 14.2, измерениями являются: «Месяц» со значениями «Июль», «Август», «Сентябрь»; «Менеджер» со значениями «Петров», «Смирнов», «Яковлев»; «Тип товара» со значениями «Плоттер», «Принтер», «Монитор». Показателем является «Объем продаж». При возникновении необходимости обработки многофакторной информации применяют или многомерные базы данных (они остаются за рамками рассмотрения), или в реляционных СУБД реализуют такие механизмы организации данных и доступа к ним, которые резко повышают эффективность анализа многофакторных данных. Такие механизмы есть и в Delphi.

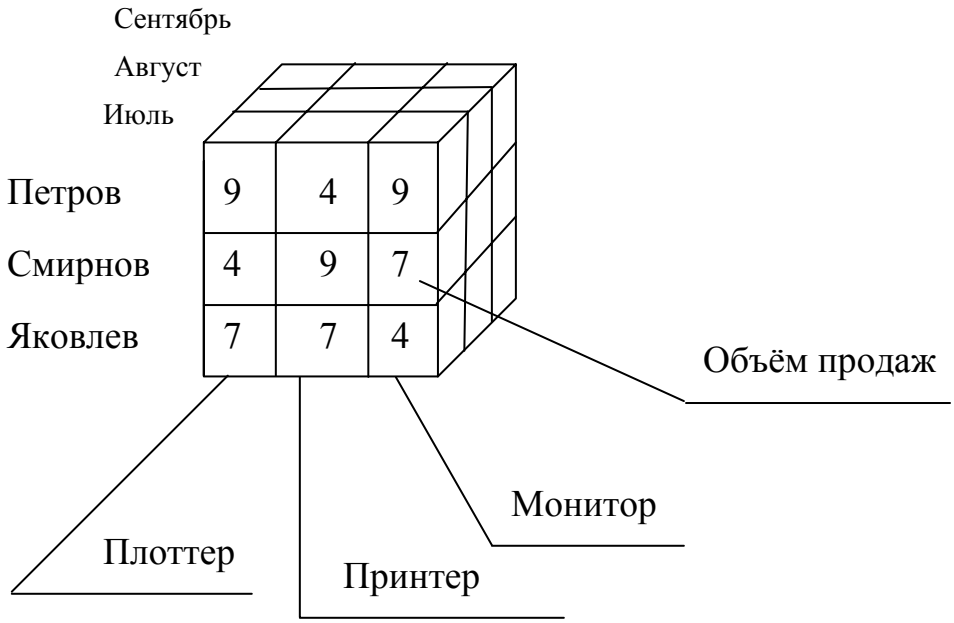



Рисунок 14.2 - Пример трёхмерной модели

Сущность подхода, при помощи которого Delphi позволяет оперировать многофакторными данными, состоит в следующем. Данные представляются в виде так называемого *метакуба* ( или *многомерного куба*), где каждому фактору соответствует своё измерение. В конкретной ячейке, как правило, представляются агрегированные данные – сумма, среднее, максимальное значение – или новые многомерные данные (кубы).

При помощи расположенных на странице **Decision Cube** компонентов Delphi в источнике данных фиксируются поля - измерения метакуба и поля, по которым должно производиться агрегирование данных (суммирование, подсчет среднего и другие). Затем определяется, какие измерения показываются как столбцы, какие – как строки. После этого пользователю предоставляется таблица многомерных данных. В форме приложения может быть расположен план метакуба, то есть список измерений, в котором каждому измерению соответствует своя колонка. Нажимая нужную кнопку, пользователь активизирует показ данных по тому или иному измерению куба.

### 14.1.2 Обзор компонентов Delphi для работы с многомерными данными

Компоненты для работы с многомерными данными расположены в Delphi на странице **Decision Cube**. Здесь расположены шесть компонентов, на основе которых можно выполнять многомерный анализ данных и расширять имеющиеся базовые возможности программным путем.

Компонент **TDecisionQuery** (  ) служит для определения набора данных, на основании которого будет создаваться многомерный куб. Он позволяет формировать произвольные запросы к базам данных. Компонент разработан специально для указанных целей, поэтому его использование при разработке

систем принятия решений является более предпочтительным, чем использование **TTable** и **TQuery**. Однако и эти компоненты могут применяться для хранения данных, на которых строится метакуб.

Компонент **TDecisionCube** (  ) выполняет все вычислительные функции, связанные с построением многомерных кубов данных. Он соединяется с набором данных при помощи свойства **DataSet**.

Компонент **TDecisionSource** (  ) является аналогом компонента **TDataSource**, но адаптирован для работы с многомерными данными. Он является промежуточным звеном между элементами отображения данных на форме и компонентом построения аналитических кубов, выполняя ряд промежуточных преобразований при передаче информации. Многомерный источник данных соединяется с компонентом **TDecisionCube** при помощи свойства **DecisionCube**.

Компонент **TDecisionPivot** (  ) служит для формирования пользовательского интерфейса, позволяет открывать и закрывать измерения метакуба. Для этого пользователю следует нажать (или отжать нажатую) кнопку, соответствующую конкретному измерению. Компонент связывается с **TDecisionSource** при помощи свойства **DecisionSource**. Применение этого компонента необязательно, поскольку компонент **TDecisionGrid** содержит собственные средства для открытия и закрытия данных по измерениям куба.

Компонент **TDecisionGrid** (  ) в системах многомерных данных имеет то же функциональное предназначение, что и компонент **TDBGrid** при работе с обычными наборами данных. В **TDecisionGrid** показываются данные из многомерного куба. Компонент связывается с **TDecisionSource** с помощью свойства **DecisionSource**.

Компонент **TDecisionGraph** (  ) предназначен для показа графиков, источником которых служат многомерные данные. Этот источник указывается в свойстве **DecisionSource** компонента.

### 14.1.3 Применение компонентов Delphi

#### 14.1.3.1 Компонент **TDecisionQuery**

Рассмотрим пример построения многомерной базы на основании реляционной базы данных.

Пусть имеются три таблицы, входящие в базу данных с псевдонимом com_sklad. Таблица Tovar.db содержит сведения о товарах. В состав таблицы входят поля: Name_t (Наименование товара), Type (Тип товара), Texxar (Технические характеристики), Zena (Цена). Первичный ключ построен по полю Name_t.

Таблица Pokup.db содержит сведения о покупателях товара. В ее состав входят поля Name_p (Наименование покупателя), City (Город, в котором расположена организация - покупатель), Telefon (Телефон). Первичный ключ построен по полю Name_p.

В третьей таблице Rashod.db содержатся сведения о расходе товара со склада. Она содержит поле N_nakl (Номер накладной), Den (День), Mes (Месяц), God (Год даты расхода), Name_t (Наименование товара), Name_p (Наименование покупателя) и Kolvo (Количество единиц отпущенного товара). Первичный ключ построен по полю N_nakl. Таблица находится в отношении «многие-к-одному» с таблицами Tovar.db и Pokup.db. Для реализации целостности построены внешние индексы по полям Name_t и Name_p.

Для определения набора данных необходимо на форме разместить компонент **TDecisionQuery** (  ) с именем DecisionQuery1. Установить в его свойство **DatabaseName** псевдоним базы данных (в данном случае - это com_sklad), а в свойстве **SQL** определить оператор **SELECT**, в котором произведено внутреннее соединение таблиц Rashod, Tovar и Pokup:

```
«SELECT P.City, R.Name_p, T.Type, R.Name_t, R.Mes, Sum(R.Kolvo*T.Zena),
AVG(R.Kolvo*T.Zena)
FROM "Pokup.db" P,
        INNER JOIN "Rashod.db" R
        ON  (P.Name_p = R.Name_p)
        INNER JOIN "Tovar.db" T
        ON  (T.Name_t = R.Name_t)
GROUP BY P.City, R.Name_p, T.Type, R.Name_t, R.Mes».
```

Существуют следующие правила объявления измерений в многомерном кубе:

- поля, по которым должны строиться измерения многомерного куба, перечисляются после ключевого слова **SELECT**;

- поля, по которым измерений строить не нужно, в операторе **SELECT** в качестве возвращаемых полей результирующего набора данных не перечисляются;

- только после полей-источников для измерений куба перечисляются агрегатные выражения (сумма, среднее, число повторений); агрегированные данные затем будут выводиться по указанным ранее измерениям;

- обязательно использование раздела **GROUP BY**, в котором необходимо перечислить все поля, по которым строятся измерения куба, причем в том же порядке, в котором они следуют после ключевого слова **SELECT**.

Поясним приведенный выше оператор **SELECT**. Данные берутся из таблиц Rashod.db (ее краткое имя - R), Pokup.db (P) и Tovar.db (T). Таблицы R и P связаны по полю Name_p, а T u R - по полю Name_t.

Измерения будут построены по полям P.City, R.Name_p, T.Type, R.Name_t и R.Mes. В качестве данных по измерениям будет также выводиться (по выбору) результат одной из следующих агрегаций: суммарная стоимость отпущенного товара (SUM) или его средняя стоимость (AVG).

Компонент **TDecisionQuery** похож на **TQuery** и не имеет отличных от него свойств, методов и событий. Установим значение True в его свойство **Active**. Теперь набор многомерных данных активен в приложении.

Для построения SQL-оператора выборки многомерных данных можно использовать редактор многомерного запроса, являющийся средством компонента **TDecisionQuery**. Щелкнуть по компоненту **TDecisionQuery** правой кнопкой мыши и в локальном меню выбрать опцию Decision Query Editor. Появится окно редактора, как показано на рисунке 14.3.

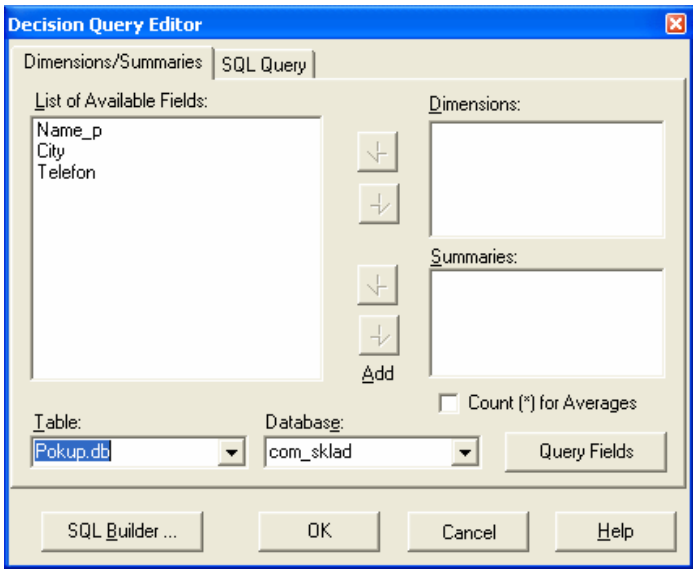



Рисунок 14.3 - Окно редактора многомерного запроса

В списке List of Available Fields перечислены поля таблицы БД, имя которой выбрано с помощью списка Table. Используя кнопки с изображением стрелок, поля, по которым должны строиться измерения в многомерном кубе, перемещаются в список Dimensions. Тем же способом в список Summaries перемещаются поля, по которым необходимо производить агрегацию. Тип агрегации запрашивается тут же. Это SUM (сумма), COUNT (счетчик повторений) и AVERAGE (среднее значение).

На странице **SQL Query** можно просмотреть и, если необходимо, отредактировать текст SQL-оператора, построенного на основе выполненных действий. Расположенная на этой странице кнопка **SQL Builder…**, позволяет перейти в режим запроса по образцу (Query By Example) для построения оператора **SELECT**.

### 14.1.3.2 Компоненты **TDecisionCube** и **TDecisionSource**

Для представления результатов SQL – запроса в виде многомерного куба можно использовать компоненты **TDecisionCube** ( ) и **TDecisionSource** ( ).

В рассмотренном ранее примере разместить на форме компонент **TDecisionCube** с именем DecisionCube1. В его свойство **DataSet** поместить имя набора данных DecisionQuery1. Определить свойства куба. Для этого в Инспекторе Объектов в свойстве **DimensionMap** нажать кнопку (...). Появится окно редактора многомерного куба Decision Cube Editor. Оно показано на рисунке 14.4.

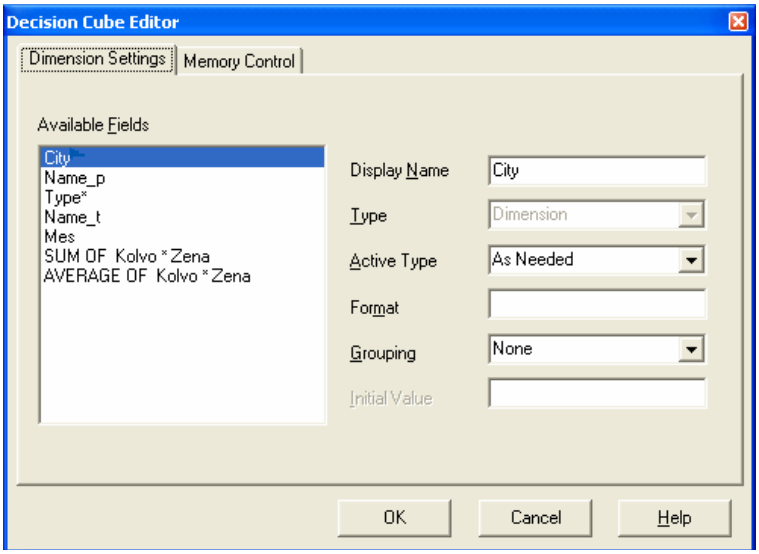



Рисунок 14.4 – Окно редактора многомерного куба

В списке Available Fields перечислены поля, по которым строятся измерения куба, и формулы для расчета агрегированных значений. Для текущего поля или формулы слева показываются параметры:

- Display Name, являющимся меткой, которая будет показываться для измерения в компонентах **TDecisionGrid**, **TDecisionPivot, TDecisionGraph**;
- Type, который является типом поля (Dimension, Sum, Average, Count);
- Active Type, который определяет показываемую информацию: As Needed (показывается, когда необходимо); Active (показывается всегда); Inactive (не показывается никогда);
- Format, который является форматом представления значений;
- Grouping, который определяет группировку по году, кварталу, месяцу и отдельному значению;
- Initial Value (начальное значение).

Закладка Memory Control позволяет определить максимальное число измерений, сумм, ячеек (группа Cube Maximums) и показывать ли только заголовки данных во время разработки приложения для экономии времени и ресурсов компьютера (группа Designer Data Options).

Компонент **TDecisionSource** выполняет в системах представления многомерных данных те же функции, что и компонент **TDataSource** в системах представления обычных данных, то есть служит источником данных для визуальных компонентов **TDecisionGrid** и **TDecisionGraph**.

### 14.1.3.3 Компонент **TDecisionGrid**

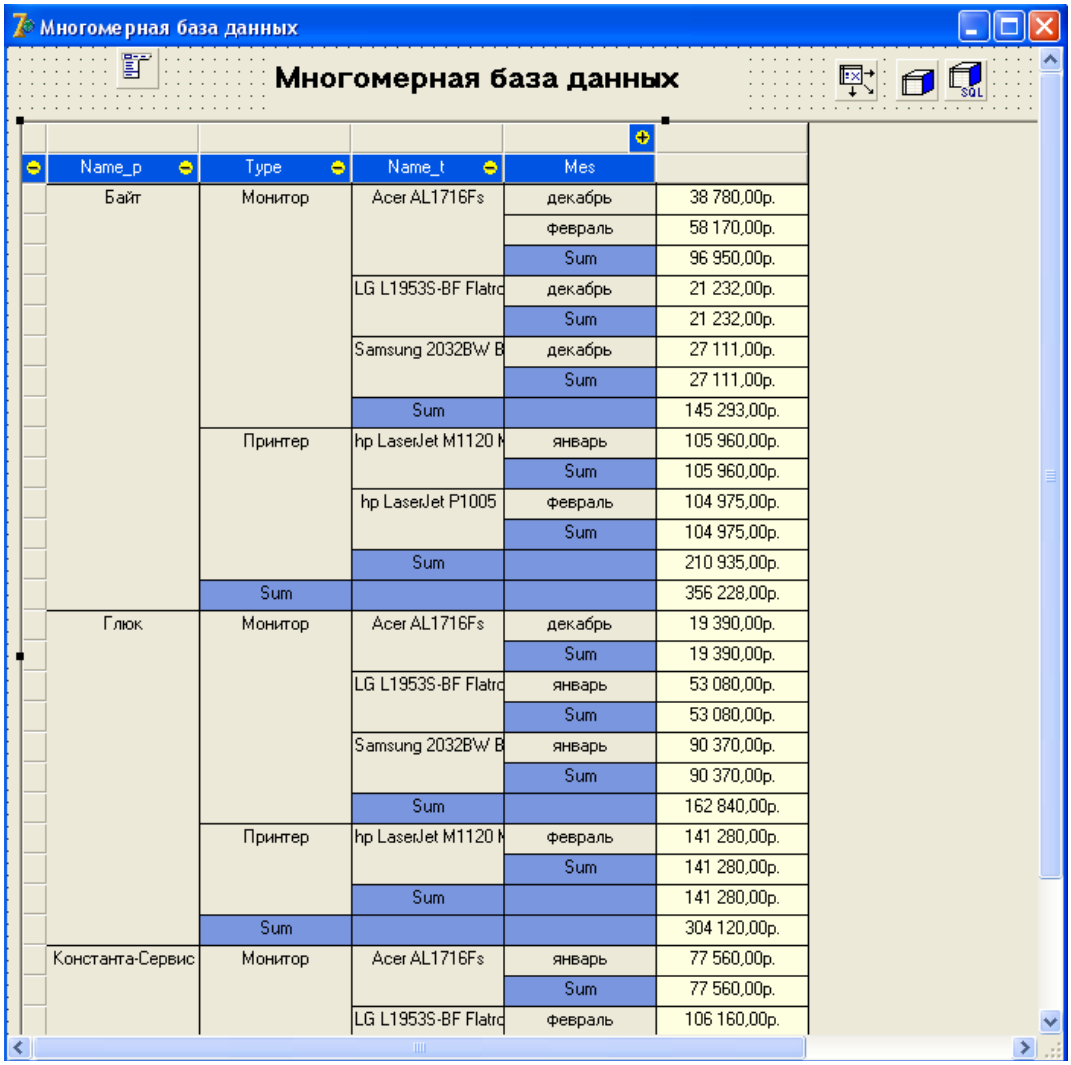
Компонент **TDecisionGrid** (  ) служит для представления многомерных данных в виде таблицы, в которой измерениям соответствуют строки и столбцы. Ячейки компонента заполняются данными, полученными на основе расчета агрегатных функций.

Разместить в форме компонент **TDecisionGrid** с именем TDecisionGrid1. Установить в его свойство **DecisionSource** имя компонента DecisionSource1. После этого произойдет визуализация многомерных данных (рисунок 14.5).

В описанном операторе **SELECT** (подпункт 14.1.3.1) определены измерения и агрегация данных в многомерном кубе. В кубе имеются измерения по полям P.City, R.Name_p, T.Type, R.Name_t и R.Mes, а в качестве данных по измерениям используются результаты агрегатных функций SUM и AVG.

Одни измерения показываются по горизонтали, построчно («Наименование покупателя», «Тип товара», «Наименование товара»), другие - вертикально, как столбцы («Город»). По умолчанию в ячейках показываются данные той агрегатной функции, которая в списке агрегатных функций в операторе **SELECT** указана первой. В данном случае - это суммарная стоимость отпущенного товара Sum(R.Kolvo*T.Zena).

Каждому измерению в сетке данных соответствует знак «+» или «-». Он указывается в ячейке, в которой содержится название предыдущего измерения. Например, знак для измерения «Тип товара» находится в ячейке, в которой показывается название предыдущего измерения - «Покупатель».




| Name_p | Type | Name_t | Mes | |
|---|---|---|---|---|
| Байт | Монитор | Acer AL1716Fs | декабрь | 38 780,00р. |
| | | | февраль | 58 170,00р. |
| | | | Sum | 96 950,00р. |
| | | LG L1953S-BF Flatro | декабрь | 21 232,00р. |
| | | | Sum | 21 232,00р. |
| | | Samsung 2032BW B | декабрь | 27 111,00р. |
| | | | Sum | 27 111,00р. |
| | | Sum | | 145 293,00р. |
| | Принтер | hp LaserJet M1120 M | январь | 105 960,00р. |
| | | | Sum | 105 960,00р. |
| | | hp LaserJet P1005 | февраль | 104 975,00р. |
| | | | Sum | 104 975,00р. |
| | | Sum | | 210 935,00р. |
| | Sum | | | 356 228,00р. |
| Глюк | Монитор | Acer AL1716Fs | декабрь | 19 390,00р. |
| | | | Sum | 19 390,00р. |
| | | LG L1953S-BF Flatro | январь | 53 080,00р. |
| | | | Sum | 53 080,00р. |
| | | Samsung 2032BW B | январь | 90 370,00р. |
| | | | Sum | 90 370,00р. |
| | | Sum | | 162 840,00р. |
| | Принтер | hp LaserJet M1120 M | февраль | 141 280,00р. |
| | | | Sum | 141 280,00р. |
| | | Sum | | 141 280,00р. |
| | Sum | | | 304 120,00р. |
| Константа-Сервис | Монитор | Acer AL1716Fs | январь | 77 560,00р. |
| | | | Sum | 77 560,00р. |
| | | LG L1953S-BF Flatro | февраль | 106 160,00р. |

Рисунок 14.5 – Визуализация многомерных данных

Знак «+» свидетельствует о том, что информация по соответствующему измерению в сетке данных не показывается. Знак «-», наоборот, свидетельствует о том, что данные по измерению показываются. Если щелкнуть по значку мышью, значок переходит в противоположное состояние, при этом данные по измерению прячутся, если они показывались в сетке, и наоборот.

Недостатком использования значков «+» и «-» является следующее: сворачивание и раскрытие данных в случае использования этих значков затрагивает и соседние измерения, расположенные справа, то есть сворачивание данных по какому-либо измерению ведет к одновременному сворачиванию данных по всем измерениям, расположенным правее от него.

Для каждого значения данных по измерению показываются промежуточные суммы. Их показ можно устранить. Для этого нужно щелкнуть по компоненту **TDecisionGrid** правой кнопкой мыши и во вспомогательном меню выбрать **Subtotals on/off**. Повторный выбор данного элемента меню приведет к включению промежуточных сумм в сетку. Подобного же эффекта, но для конкретного измерения, можно добиться, если выбрать мышью ячейку заголовка конкретного измерения и нажать правую кнопку. Выбор элемента **Display Data and Subtotals** приводит к показу промежуточных сумм. Выбор элемента **Display Data Only** устраняет показ промежуточных сумм для измерения.

Те же действия осуществимы и для конкретного значения по измерению. Если щелкнуть правой кнопкой мыши по ячейке данных, соответствующих нужному измерению, то появится меню, в котором к элементам **Display Data and Subtotals** и **Display Data Only** добавится элемент **Display Subtotals Only**. Выбор этого элемента приводит к показу только промежуточных сумм. Установка одного из названных режимов показа относится не к измерению в целом, а к данным, касающимся того значения измерения, которое показывается в ячейке.

Все описанные возможности доступны как во время разработки приложения, так и во время его выполнения. То или иное измерение может быть исключено из сетки данных или вновь добавлено в нее как во время разработки, так и во время выполнения приложения.

Чтобы скрыть данные по измерению, можно воспользоваться двумя способами.

Первый - щелкнуть правой кнопкой мыши по ячейке данных измерения, которое показывается построчно, и во вспомогательном меню выбрать элемент **Drill in to this value**.

Второй способ - нажать правую кнопку мыши:

- над заголовком измерения;
- над пустой ячейкой в ряду ячеек сетки, расположенных над рядом, в котором показываются названия измерений;
- над крайним левым (пустым) столбцом сетки и в списке измерений снять отметку с необходимого измерения.

Чтобы включить измерение в состав показываемых, нужно воспользоваться вторым способом и установить отметку у необходимого измерения.

### 14.1.3.4 Компонент **TDecisionPivot**

Компонент **TDecisionPivot** (  ) позволяет динамически определять текущие измерения куба, а также выбирать содержимое ячеек данных. Каждому измерению куба в компоненте соответствует своя кнопка. Когда кнопка нажата, измерение в составе куба является открытым, то есть данные, соответствующие измерению, показываются в кубе. Когда кнопка, соответствующая измерению, не нажата (отжата), данные, соответствующие измерению, в кубе не показываются. Агрегатным функциям соответствует отдельная кнопка.

Разместить на форме компонент **TDecisionPivot** (имя DecisionPivot1). Установить в его свойство **DecisionSource** имя компонента DecisionSource1. Общий вид окна программы показан на рисунке 14.6.

Кнопки в компоненте **TDecisionPivot** размещены в трех областях.

В левой области размещена кнопка «Стоимость», название которой задается в окне редактора многомерного куба Decision Cube Editor для измерения Sum(R.Kolvo*T.Zena) . Это кнопка предназначена для выбора агрегатных функций. По умолчанию в качестве текущей агрегатной функции берется первая из функций, определенных в операторе **SELECT** (компонент **TDecisionQuery***).* Чтобы сменить агрегатную функцию, нужно нажать кнопку и из появившегося списка выбрать нужную функцию.

Средняя область компонента **TDecisionPivot** содержит кнопки измерений, данные по которым показываются построчно. В данном случае - это кнопки, соответствующие измерениям «Наименование покупателя», «Тип товара», «Наименование товара», «Месяц»: «Покупатель», «Товар», «Наименование», «Месяц».

Правая часть компонента содержит кнопки измерений, данные по которым показываются в виде столбцов. В данном случае - это кнопка, соответствующая измерению «Город».

Чтобы перевести кнопку из одной области в другую, следует щелкнуть по ней правой кнопкой мыши и во вспомогательном меню выбрать опцию Move to Row Area (при переводе измерения в построчный показ) или Move to Column Area (при переводе измерения в показ по столбцу). Изменить вид показа измерения можно также, если перетащить кнопку с измерением в рамках объекта **TDecisionPivot** между двумя панелями. Первая из них (  ) означает, что данное измерение (ось) должно располагаться по вертикали, вторая – по горизонтали (  ). Вид окна программы с изменением в показываемых измерениях приведен на рисунке 14.7.

Обычно по измерению показываются все его данные. Например, по измерению «Наименование покупателя» показываются все покупатели. Однако часто бывает необходимо показывать данные не по всем, а по конкретному покупателю. Для этого следует переместить указатель мыши на кнопку соответствующего измерения, нажать правую кнопку мыши и отметить опцию **Drilled in** во вспомогательном меню.

Кнопка «Покупатель» изменит свой заголовок на «Покупатель.All Values». Теперь, чтобы зафиксировать конкретного покупателя, следует нажать правую кнопку мыши и в появившемся списке выбрать название конкретного покупа-

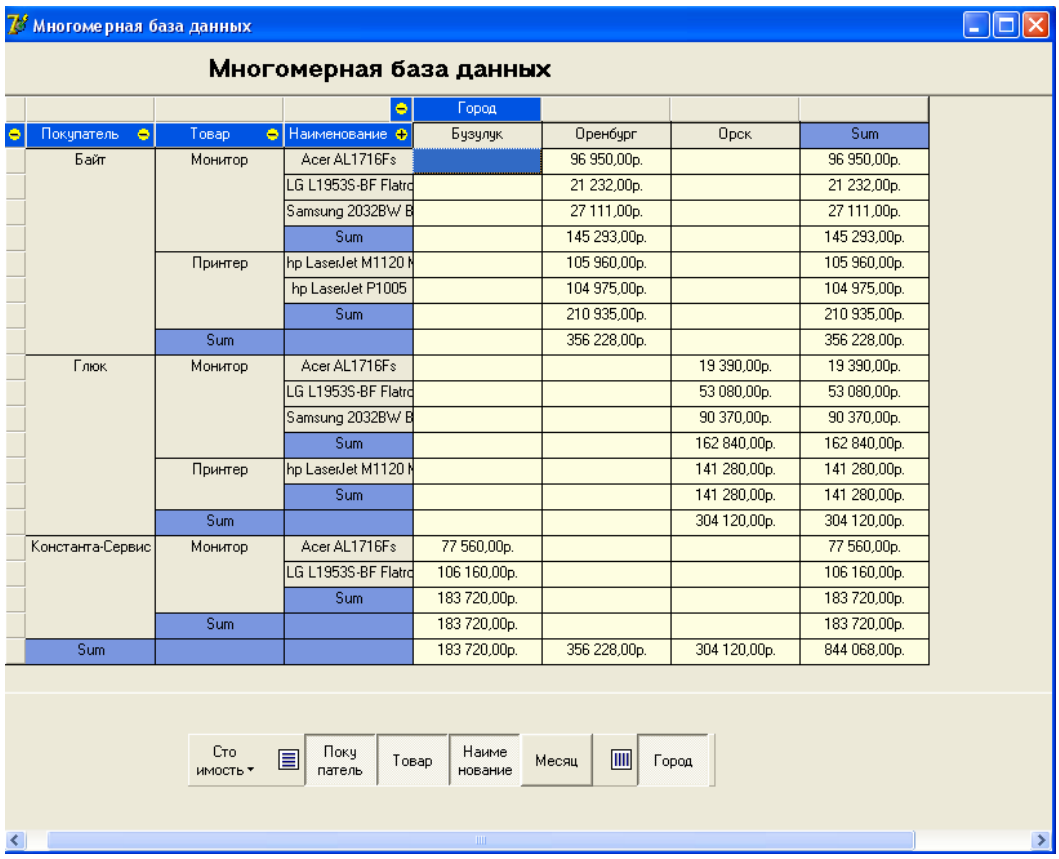



Рисунок 14.6 - Форма с компонентом **TDecisionPivot**

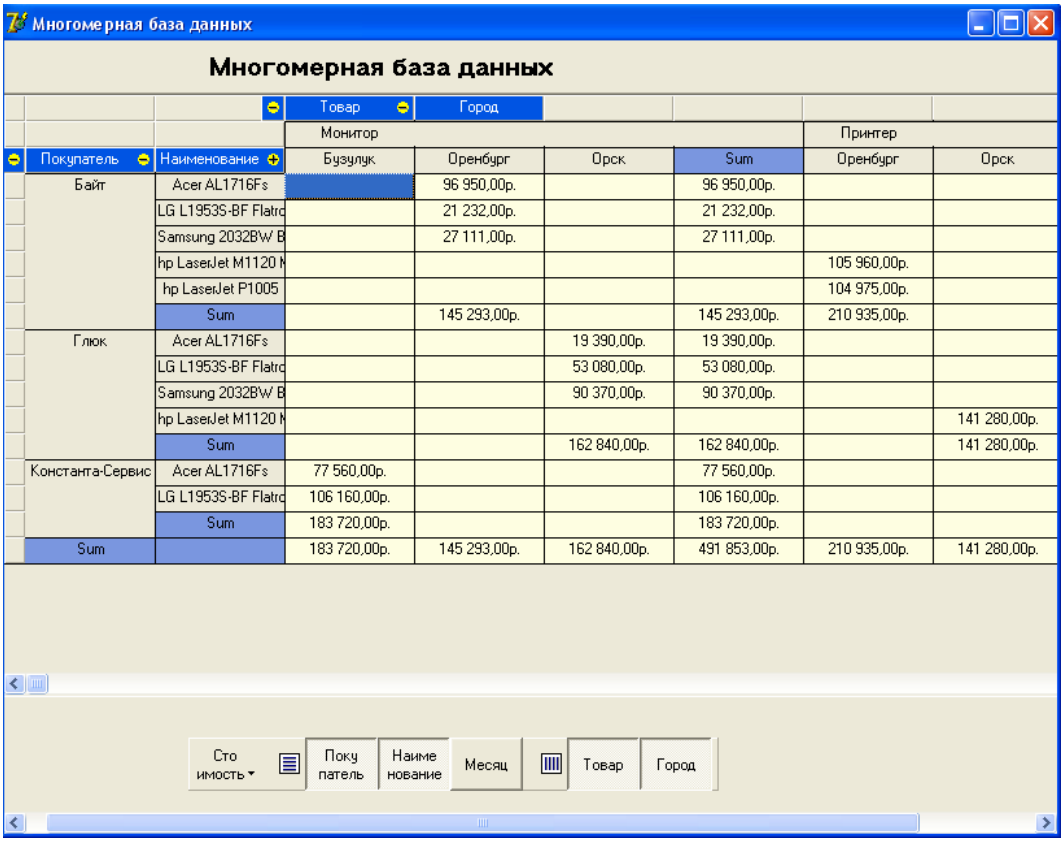



Рисунок 14.7 - Форма с измененным компонентом **TDecisionPivot**

теля. При этом его имя будет показываться в кнопке, соответствующей измерению «Покупатель», а данные в кубе будут сечением по координате измерения, соответствующей конкретному покупателю (рисунок 14.8).

Чтобы вновь показывать все значения по измерению (например, всех покупателей по измерению «Покупатель»), нужно во вспомогательном меню снять отметку с опции **Drilled in**.

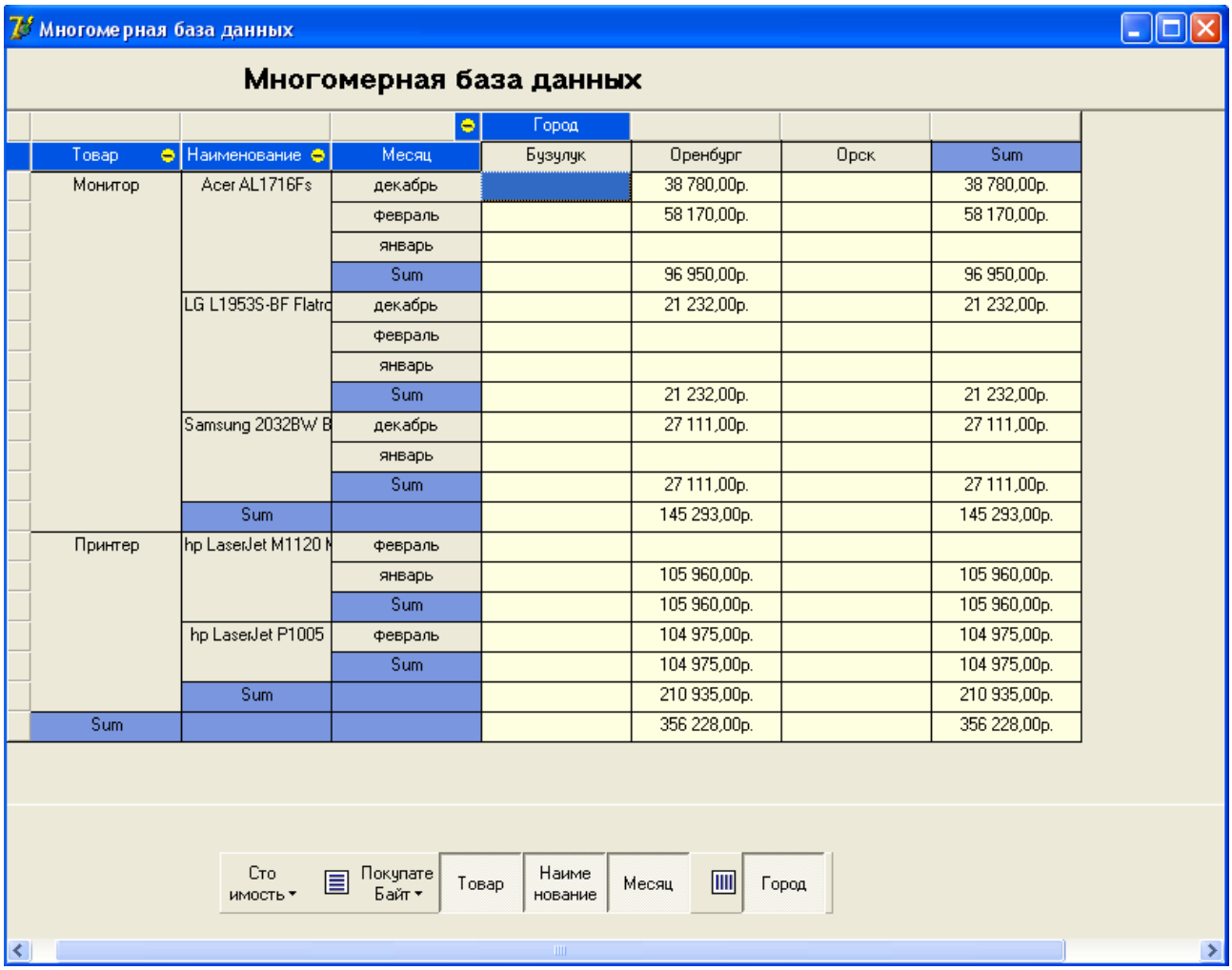



Рисунок 14.8 - Сечение по конкретному покупателю

### 14.1.3.5 Компонент **TDecisionGraph**

Компонент **TDecisionGraph** (  ) предназначен для графического отображения многомерной информации. Он является наследником компонента **TChart** - пользовательской диаграммы. Поэтому настройка внешнего вида графика осуществляется стандартным способом, как и у компонента **TChart**. С помощью Редактора диаграмм **Editing DecisionGraph**, который вызывается двойным нажатием на компонент **TDecisionGraph**, можно установить название **Titles**, оси **Axis**, легенду **Legend**, трехмерное представление **3D** и другие параметры.

После размещения компонента **TDecisionGraph** на форме и задания в свойстве **DecisionSource** ссылки на объект **DecisionSource1** на графике сразу появится изображение, наглядно показывающее, к примеру, каковы суммарные стоимости товаров по каждому городу. Если сменить координатные оси с помощью объекта **TDecisionPivot1**, можно узнать, каковы суммарные стоимости товаров в городах по каждому из товаров (рисунок 14.9).

Число товаров можно добавить и как поле для анализа (в список **Summaries**), и как третье измерение, благодаря чему можно посмотреть, например, в какой город поступили максимальные и минимальные объемы  и для каких видов товара.

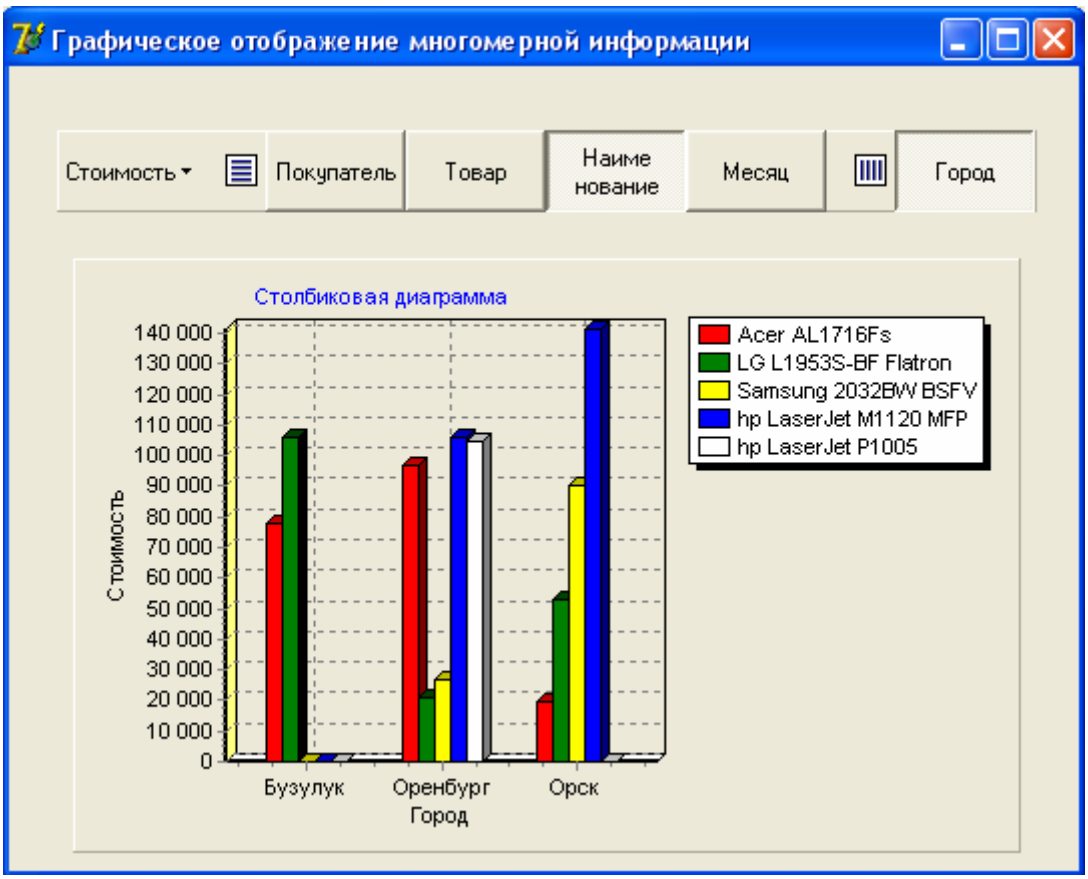



Рисунок 14.9 - Графическое отображение многомерной информации

## 14.2 Задание на выполнение работы

14.2.1 Создать на диске D:\ в папке с именем группы каталог, в котором будут храниться все файлы, относящиеся к лабораторной работе.

14.2.2 При помощи утилиты **BDE Administrator** создать псевдоним базы данных и запомнить его в созданном каталоге.

14.2.3 Запустить утилиту **Database Desktop**, установить умалчиваемый псевдоним. Задать структуру таблиц «Товар», «Покупатель», «Расход» в соответствии с таблицами 14.1 - 14.3. Для таблиц использовать тип Paradox 7. Сохранить созданную структуру в соответствующих файлах. Определить ссылочную целостность между таблицами в соответствии с типами связей, существующих между таблицами.

Таблица 14.1 – Товар

| Наименование товара | Тип товара | Технические характеристики | Цена |
|---|---|---|---|
| Acer AL1716Fs | Монитор | 17" (LCD, 1280x1024) | 3878 |
| LG L1953S-BF Flatron | Монитор | 19" <Black> (LCD, 1280x1024) | 5308 |
| hp LaserJet M1120 MFP | Принтер | A4, 19 с./мин, 32Mb, принтер +сканер +копир USB2.0 | 7064 |
| Intel Celeron D 336 | Процессор | 2.8 ГГц/ 256K/ 533МГц , 775-LGA | 1091 |
| Intel Celeron Dual-Core E1400 | Процессор | 2.0 ГГц/ 512K/ 800МГц, 775-LGA | 1894 |
| Samsung 2032BW BSFV | Монитор | 20" <HG Black> (LCD, Wide, 1680x1050, +DVI) | 9037 |
| hp LaserJet P1005 | Принтер | A4, 14 с./мин 2Мб USB2.0 | 4199 |
| Intel Core 2 Duo E6550 | Процессор | 2.33 ГГц/ 4Мб/ 1333МГц, 775-LGA | 5197 |
| Acer V203Wb | Монитор | 20" <Black> (LCD, Wide, 1680x1050) | 5378 |
| Acer B223Wymdr | Монитор | 22" <Dark Grey> с поворотом экрана (LCD, Wide, 1680x1050, +DVI) | 8645 |
| hp LaserJet P1505 | Принтер | A4, 23 с./мин 2Мб USB2.0 | 6513 |
| hp LaserJet P2014 | Принтер | A4, 23 с./мин 32Mb USB2.0/LPT | 9625 |
| Intel Pentium 4 631 | Процессор | 3.0 ГГц/ 2Мб/ 800МГц, 775-LGA | 2144 |
| Samsung ML-1640 | Принтер | A4, 8Mb, лазерный, 16 с./мин, 1200dpi, USB 2.0 | 3567 |
| XEROX WorkCentre 3119 | Принтер | A4, цифр.копир, 18коп/мин+лаз. принтер 18с./мин+сканер, USB | 5570 |
| NEC 2090UXi-BK | Монитор | 20.1" <Black> с поворотом экрана (LCD, 1600x1200, +DVI-I, DVI-D) | 33819 |
| Samsung T190N WASU | Монитор | 19" <Rose Black> (LCD, Wide, 1440x900) | 8125 |
| Acer AL2016W Csd | Монитор | 20" <Silver> (LCD, Wide, 1680x1050, +DVI) | 7537 |

Таблица 14.2 – Покупатель

| Наименование | Город | Телефон |
|---|---|---|
| Центр | Оренбург | 774711 |
| Байт | Оренбург | 369582 |
| Амиго | Орск | 15236 |
| Глюк | Орск | 63524 |
| Джаз | Оренбург | 770268 |
| Константа - Сервис | Бузулук | 225612 |

Таблица 14.3 - Расход

| Номер накладной | День | Месяц | Год | Товар | Покупатель | Количество |
|---|---|---|---|---|---|---|
| | | | | | | |

14.2.4 Запустить Delphi. На **Form1** поместить:

- компоненты набора данных **Table**, источника данных **DataSource**, сетки **DBGrid** для отображения записей таблиц «Товар», «Покупатель», «Расход», задать их свойства;
- метки **Label** для задания заголовков;
- кнопку **Button**, предназначенную для открытия формы **Form2**, задать значения ее свойств.

Вид формы **Form1** на этапе разработки показан на рисунке 14.10. Сохранить форму. Запустить приложение. Занести в таблицы соответствующие записи. Для таблиц «Товары» и «Покупатели» использовать значения в таблицах 14.1 и 14.2. Для таблицы «Расход» занести свои данные.

14.2.5 На форме **Form1** для кнопки **Button** задать процедуру открытия формы **Form2**.

14.2.6 Создать форму **Form2** «Многомерная база данных», на которой разместить компоненты для работы с многомерными данными **TDecisionQuery**, **TDecisionCube**, **TDecisionSource**, **TDecisionGrid**, **TDecisionPivot**, задать значения их свойств. Сохранить сделанные изменения, запустить приложение. Протестировать его работу.

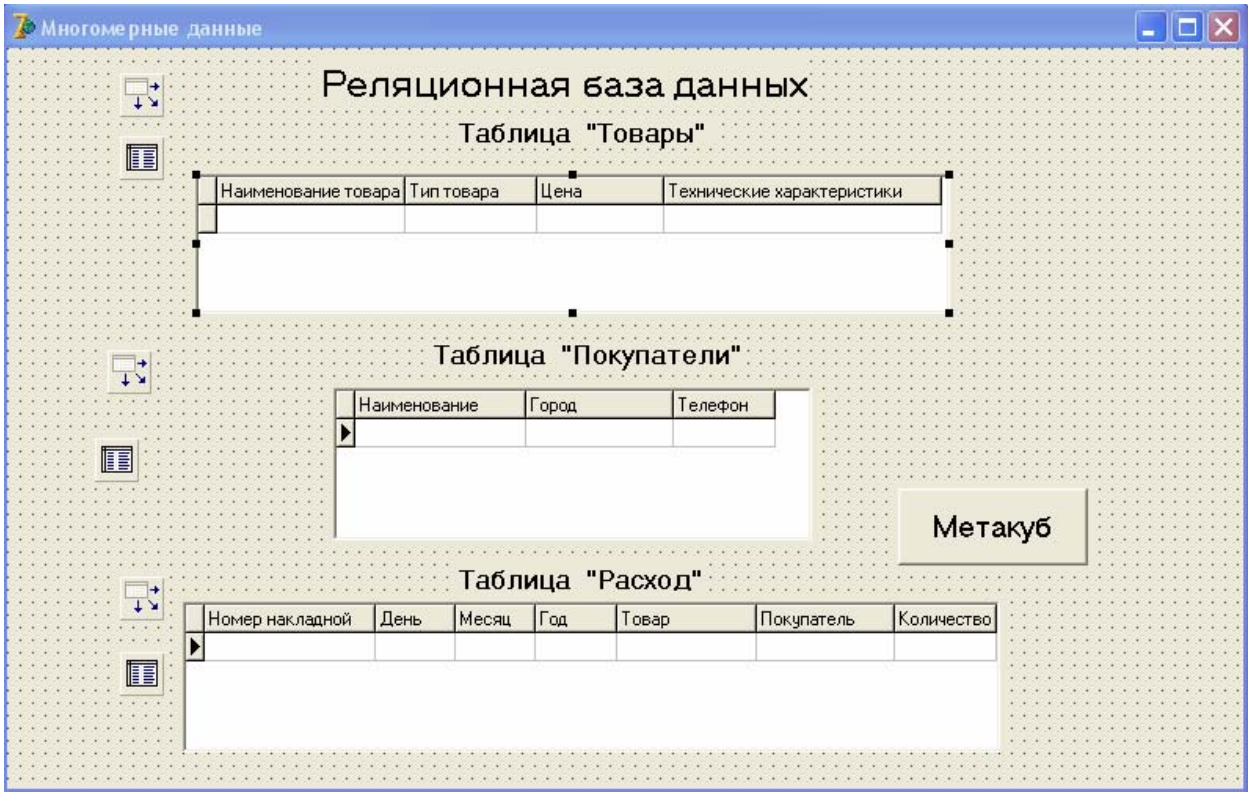



Рисунок 14.10 - Вид формы **Form1** на этапе разработки

14.2.7 На форму **Form2** поместить кнопку для открытия формы **Form3**, задать ее программный код.

14.2.8 Создать форму **Form3** «Графическое отображение многомерной информации». На ней разместить компоненты **TDecisionQuery**, **TDecisionCube**, **TDecisionSource**, **TDecisionGraph**, **TDecisionPivot**, задать значения их свойств. Сохранить сделанные изменения, запустить приложение. Протестировать его работу, меняя координатные оси с помощью объекта **TDecisionPivot** .

14.2.9 Оформить отчет по лабораторной работе.

### 14.3 Содержание отчёта

14.3.1 Название работы.

14.3.2 Цель работы.

14.3.3 Логические схемы данных созданных таблиц.

14.3.4 Перечень компонентов, использованных при создании приложения, с указанием их назначения и перечнем значений определяемых свойств.

14.3.5 Перечень команд, использованных при выполнении лабораторной работы, с указанием их назначения.

14.3.6 Формулировка SQL-запроса.

### 14.4 Тесты и контрольные вопросы

14.4.1 Многомерные модели данных предназначены для информационных систем:

а) оперативной обработки данных;

б) аналитической обработки данных;

в) функциональной обработки данных;

г) логической обработки данных.

14.4.2 Агрегируемость данных в многомерной базе данных – это:

а) рассмотрение информации на различных уровнях ее обобщения;

б) обеспечение высокого уровня статичности данных и их взаимосвязей;

в) задание функций прогнозирования и применение их к различным временным интервалам;

г) ограничение области видимости свойства объекта.

14.4.3 Историчность данных в многомерной базе данных предполагает:

а) рассмотрение информации на различных уровнях ее обобщения;

б) обеспечение высокого уровня статичности данных и их взаимосвязей, а также обязательность привязки данных ко времени;

в) задание функций прогнозирования и применение их к различным временным интервалам;

г) ограничение области видимости свойства объекта.

14.4.4 Измерение в многомерной базе данных – это:

а) множество однотипных данных, образующих одну из граней гиперкуба;

б) множество однотипных атрибутов, образующих одну из граней гипер-

куба;

в) множество однотипных записей, образующих одну из граней гиперкуба;

г) множество однотипных таблиц, образующих одну из граней гиперкуба.

14.4.5 Вращение в многомерной базе данных – это:

а) изменение порядка измерений при физическом представлении данных;

б) изменение порядка измерений при логическом представлении данных;

в) изменение порядка измерений при визуальном представлении данных;

г) изменение порядка измерений при функциональном представлении данных.

14.4.6 К достоинствам многомерной модели данных относится:

а) контроль целостности связей;

б) допустимость образования произвольных связей;

в) возможность определения функций обработки данных;

г) возможность аналитической обработки данных.

14.4.7 На какой странице Delphi расположены компоненты, позволяющие работать с многомерными данными:

а) **Data Controls**; б) **Data Access**; в) **Standard**;

г) **Additional**; д) **Decision Cube**?

14.4.8 Для чего предназначен компонент **TDecisionCube**:

а) для определения набора данных, на основании которого будет создаваться многомерный куб;

б) для соединения с источником данных;

в) для открытия и закрытия измерений куба;

г) для отображения данных из многомерного куба;

д) для показа графиков, источником которых служат многомерные данные?

14.4.9 Какой визуальный компонент используется для отображения многомерных данных:

а) **TDecisionQuery**; б) **TDecisionCube**;

в) **TDecisionSource**; г) **TDecisionGrid**;

д) **TDecisionPivot**?

14.4.10 Какой невизуальный компонент используется для определения набора данных, на основании которого будет создаваться многомерный куб:

а) **TDecisionQuery**; б) **TDecisionCube**;

в) **TDecisionSource**; г) **TDecisionGrid**;

д) **TDecisionPivot**?

14.4.11 Что такое многомерные базы данных?

14.4.12 Для каких целей используется многомерный подход к представлению данных в базе?

14.4.13 Какими достоинствами обладает многомерная организация данных по сравнению с реляционной моделью?

14.4.14 Что такое измерение?

14.4.15 Что такое ячейка?

14.4.16 Приведите примеры измерений.

14.4.17 Какой тип поля чаще всего используется для ячейки?

14.4.18 Какие компоненты используются при создании базы многомерных данных?

14.4.19 Назовите правила объявления измерений в многомерном кубе.

14.4.20 Для каких целей используется компонент **TDecisionQuery**?

14.4.21 Для каких целей используется компонент **TDecisionGrid**?

14.4.22 Какой компонент применяется для визуализации многомерных данных?

14.4.23 Для чего предназначен компонент **TDecisionSource**?

14.4.24 Что показывают знаки «+», «-»?

14.4.25 Как можно динамически определить текущие измерения куба?

14.4.26 Каким образом можно зафиксировать данные по конкретному покупателю?

14.4.27 При помощи какого свойства компонента **TDecisionCube** осуществляется соединение с источником многомерных данных?

14.4.28 Как можно изменить имена измерений для отображения на экране в компонентах **TDecisionGrid**, **TDecisionPivot**, **TDecisionGraph**?

14.4.29 Каким образом в компоненте **TDecisionPivot** можно перевести кнопку из одной области в другую?

14.4.30 Для чего предназначен компонент **TDecisionGraph**?

14.4.31 Как осуществляется настройка внешнего вида графика, созданного на компоненте **TDecisionGraph**?

14.4.32 Как можно сменить координатные оси при графическом представлении многомерных данных?

14.4.33 Как задать заголовок графика, созданного на компоненте **TDecisionGraph**?

14.4.34 Как задать имена координатных осей графика, созданного на компоненте **TDecisionGraph**?

14.4.35 Каким образом можно отредактировать легенду графика?

# Список использованных источников


1 **ГОСТ 20886-85** Организация данных в системах обработки данных. Термины и определения. – Взамен ГОСТ 20886-75; введ. 1986-01-07. – М.: Издательство стандартов, 1986. – 13 с.

2 **Голицына, О. Л.** Базы данных: учеб. пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2-е изд., испр. и доп. - М.: Форум, 2007. – 400 с. – ISBN 978-5-91134-098-8. – ISBN 978-5-16-002966-5.

3 **Дейт, К. Д.** Введение в системы баз данных = An Introduction to Database Systems: пер. с англ. / К. Д. Дейт. - 8-е изд. - М. : Вильямс, 2005. - 1328 с. - ISBN 5-8459-0788-8. - ISBN 0-312-19784-4.

4 **Диго, С.М.** Базы данных: проектирование и использование: учебник / С.М. Диго. – М.: Финансы и статистика, 2005. – 592 с.

5 **Карпова, Т. С.** Базы данных: модели, разработка, реализация: учеб. для вузов / Т. С. Карпова. - СПб. : Питер, 2001. - 304 с. - ISBN 5-272-00278-4.

6 **Кузин, А. В.** Базы данных : учеб. пособие / А. В. Кузин, С. В. Левонисова. - М. : Академия, 2005. - 320 с. - ISBN 5-7695-1796-4.

7 **Малыхина, М. П.** Базы данных: основы, проектирование, использование / М. П. Малыхина. - СПб.: БХВ - Санкт-Петербург, 2004. - 512 с. - ISBN 5-94157-310-3.

8 **Марков, А. С.** Базы данных: введение в теорию и методологию / А. С. Марков, К. Ю. Лисовский. – М.: Финансы и статистика, 2006. – 512 с. – ISBN 5-279-02298-5.

9 **Советов, Б. Я.** Базы данных: теория и практика: учеб. / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд., стер. - М.: Высш. шк., 2007. – 463 с. – ISBN 978-5-06-004876-6.

10 **Советов, Б. Я.** Информационные технологии: учебник для вузов / Б. Я. Советов, В. В. Цехановский. – М.: Высш. шк., 2008. – 263 с. – ISBN 978-5-06-004275-7.

11 **Швецов, В. И.** Базы данных : учеб. пособие для вузов / В.И. Швецов, А. Н. Визгунов, И. Б. Мееров. - Нижний Новгород : Изд-во ННГУ, 2004. - 271 с. - ISBN 5-85746-806-X.

12 **Хомоненко, А. Д.** Базы данных: учебник для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; под ред. А. Д. Хомоненко .- 5-е изд., доп. - М. : Бином; СПб. : Корона Принт, 2006. - 736 с. - ISBN 5-7931-0346-5.

13 **Васильев, А.** Microsoft Office 2007: новые возможности / А. Васильев. – СПб.: Питер, 2007. – 160 с. - ISBN 978-5-469-0162908.

14 **Золотова, С. И.** Практикум по Access: подготовительный курс, предваряющий более глубокое изучение технологии баз данных / С. И. Золотова. – М.: Финансы и статистика, 2007. – 144 с. – ISBN 978-5-279-02284-7.

15 Microsoft Office 2007. Все программы пакета: Word, Excel, Access, PowerPoint, Publisher, OneNote, InfoPath, Groove: самоучитель / А. М. Тихомиров [и др.]. – СПб.: Наука и техника, 2008. – 608 с.

16 **Мак-Дональд, М.** Access 2007. Недостающее руководство: пер. с англ. / М. Мак-Дональд. - М.: Русская редакция; СПб.: БХВ - Санкт-Петербург, 2007. - 784 с. - ISBN 978-5-7502-0343-3. - ISBN 978-5-9775-0093-7.

17 **Стоцкий, Ю.** Office 2007. Самоучитель / Ю. Стоцкий, А. Васильев, И. Телина. - СПб.: Питер, 2008. - 524 с. - ISBN 978-5-91180-524-1.

18 **Архангельский, А. Я.** Delphi7: справ. пособие / А. Я. Архангельский. - М. : Бином-Пресс, 2004. - 1024 с. - ISBN 5-9518-0027-7.

19 **Бобровский, С. И.** Delphi 7: учебный курс / С. И. Бобровский. - СПб.: Питер, 2008. - 736 с. - ISBN 978-5-8046-0086-1.

20 **Гофман, В. Э.** Работа с базами данных в Delphi / В. Э. Гофман.- 2-е изд. - СПб.: БХВ-Петербург, 2003. - 624 с. - ISBN 5-94157-211-5.

21 **Осипов, Д.** Delphi. Профессиональное программирование / Д. Осипов. - СПб.: Символ-Плюс, 2006. - 1056 с. - ISBN 5-93286-074-X.

22 **Понамарев, В. А.** Базы данных в Delphi 7 : самоучитель / В. А. Понамарев. - СПб. : Питер, 2003. - 224 с. - ISBN 5-314-00194-2.

23 **Сорокин, А. В.** Delphi. Разработка баз данных / А. В. Сорокин. - СПб.: Питер, 2005. - 477 с. - ISBN 5-469-00927-0.

24 **Фаронов, В. В.** Программирование баз данных в Delphi 7: учеб. курс / В. В. Фаронов. - СПб. : Питер, 2005. - 459 с. - ISBN 5-318-00100-9.

25 Delphi 7. Наиболее полное руководство / под ред. А. Хомоненко. - СПб. : БХВ-Петербург, 2004. - 1216 с. - ISBN 5-94157-267-0.

26 **Хомоненко, А. Д.** Работа с базами данных в Delphi / А. Д. Хомоненко, В. Э.Гофман. - СПб.: БХВ-Петербург, 2005. - 640 с. - ISBN 5-94157-361-8.

27 **Черноусова, А. М.** Создание и использование реляционной базы данных в MS Access: лабораторный практикум / А. М. Черноусова. - Оренбург : ОГУ, 2001. - 96 с.

28 **Черноусова, А.М.** Создание баз данных и работа с ними в среде Delphi: методические указания к лабораторному практикуму / А. М. Черноусова, С. В. Берещук. - Оренбург: ОГУ, 2002. -– 72 с.

29 **Черноусова, А. М.** Разработка приложений в MS ACCESS [Электронный ресурс] : метод. указ. к лаб. и самостоят. работам / А. М. Черноусова, А. В. Трибунский. - Оренбург : ГОУ ОГУ, 2003. - 23 с. - Режим доступа: http: //artlib.osu.ru/web/boobks/ metod03/metod178.pdf. - Загл. с экрана.

30 **Боровский, Г. В.** Справочник инструментальщика / Г. В. Боровский, С. Н. Григорьев, А. Р. Маслов; под общей редакцией А. Р. Маслова. - М.: Машиностроение, 2005. - 464 с. - ISBN 5-217-03284-7.

# Приложение А
***(рекомендуемое)***
## Карта правильных ответов к тестам

Таблица А.1

| Лабораторная работа | Номер вопроса | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | б | в | а | в | г | б | а | в | б | в |
| 2 | б | б | а | д | в | б | а | г | а | а |
| 3 | в | а | в | г | в | д | а | а | г | д |
| 4 | а | в | б | д | в | д | б | г | а | в |
| 5 | а | д | а | г | г | а | а | б | в | в |
| 6 | в | б | в | б | д | в | г | б | б | а |
| 7 | г | г | б | б | в | д | в | а | б | а |
| 8 | б | а | г | б | г | в | а | в | г | а |
| 9 | а | б | д | в | б | г | а | в | б | б |
| 10 | в | а | б | а | б | д | а | г | в | а |
| 11 | а | а | б | в | а | г | в | б | б | в |
| 12 | в | г | б | а | в | б | г | в | д | д |
| 13 | г | а | г | б | в | г | а | б | б | в |
| 14 | б | а | б | а | в | г | д | б | г | а |