ЧАСТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«СТАВРОПОЛЬСКИЙ МНОГОПРОФИЛЬНЫЙ КОЛЛЕДЖ»

М Е Т О Д И Ч Е С К И Е У К А З А Н И Я

к выполнению практических работ

по междисциплинарному курсу

« Управление и автоматизация баз данных»

для обучающихся специальности

09.02.13 «Интеграция решений с применением технологий искусственного интеллекта»

Ставрополь 2026

Методические указания разработаны в соответствии с программой междисциплинарного курса «Управление и автоматизация баз данных» для студентов специальности «Интеграция решений с применением технологий искусственного интеллекта».

В методических указаниях содержатся основные требования к структуре, содержанию, объёму и оформлению разделов пояснительной записки самостоятельной работы, объясняется порядок её выполнения и защиты.

Составитель: Брехова В.С.

Рассмотрено на заседании методического объединения укрупненных групп специальностей 09.00.00 «Информатика и вычислительная техника»; 10.00.00 «Информационная безопасность» Протокол № от г.

Рекомендовано к использованию в учебном процессе Методическим советом СМК, протокол № от г.

СОДЕРЖАНИЕ

Практическая работа №1.
Создание структуры базы данных.

База данных представляет собой единый большой объект, который объединяет такие составляющие, как таблицы, отчеты, запросы, формы и т. д., и позволяет хранить их в едином дисковом файле, имеющим тип MDB.

Основным структурным компонентом базы данных является таблица. Каждая таблица содержит записи определенного вида, например о студентах, обучающихся в высшем учебном заведении.

Каждая запись таблицы содержит всю необходимую информацию об отдельном элементе базы данных. Например, запись о студенте может содержать номер его личного дела, фамилию, имя, отчество, пол, дату рождения, номер учебной группы. Такие отдельные структурные элементы записи таблицы называются полями.

Первым этапом при создании таблицы является определение перечня полей, из которых она должна состоять, их типов и размеров.

Каждому полю таблицы присваивается уникальное имя, которое не может содержать более 64 символов, не разрешается использовать символы: точка, восклицательный знак, квадратные скобки.

Тип данных указывает Access, как обрабатывать эти данные. Можно использовать следующие типы:

- Текстовый - для текстовой информации и чисел при невыполнении матических расчетов (до 255 символов).
- Поле МЕМО – для хранения произвольного текста, комментариев (до 64 000 символов).
- Числовой – при выполнении над данными матических операций
- Денежный – специальное числовое поле используется для операций с деньгами.
- Дата/время – предназначено для хранения информации о дате и времени. (Даты и время, относящиеся к годам с 100 по 9999, включительно)
- Счетчик - специальное числовое поле, в котором Access автоматически присваивает уникальный порядковый номер каждой записи.
- Логический – может иметь только одно из двух возможных значений «Да» или «Нет».
- Поле объекта OLE – объект (например, электронная таблица Microsoft Excel или Microsoft Draw), созданный другим приложением.

Задание к лабораторной работе.

Сформируйте структуру таблицу СТУДЕНТ для хранения в ней справочных сведений о студентах, обучающихся в вузе. Имена, типы и размеры полей таблицы приведены в табл.1.

Выполнение работы.

1. Откройте приложение Microsoft Access, выполнив:

a) Пуск, Программы, Microsoft Access. Откроется окно Microsoft Access, в котором возле поля Новая база данных установите флажок. Нажмите ОК.

b) Появится следующее диалоговое окно Файл новой базы данных, в котором установите нужную папку и в поле Имя файла наберите Sess.

c) Нажмите кнопку Создать.

2. Создайте таблицу базы данных. Для этого:

a) В диалоговом окне Sess: база данных откройте вкладку Таблица, а затем щелкните по кнопке Создать

b) Откроется новое диалоговое окно Новая таблица, в котором щелкните Конструктор, а затем ОК

В результате проделанных операций открывается окно таблицы Таблица1: Таблица в режиме конструктора, в котором следует определить поля таблицы.

3. Определите поля таблицы (см. табл. 1). Для определения первого поля выполните следующие действия:

a) Введите в ячейку столбца Поле имя первого поля Номер и нажмите клавишу Enter.

b) В ячейке столбца Тип данных оставьте выводящееся по умолчанию значение Текстовый.

c) Для переключения в окно Свойства поля нажмите клавишу F6. Откорректируйте размер поля (введите 5).

d) Заполнение ячейки столбца Описание является необязательным и включает сведения о содержащихся в поле данных.

Поле	**Тип поля**	**Размер поля**
Номер	Текстовое	5
Фамилия	Текстовое	15
Имя	Текстовое	10
Отчество	Текстовое	15
Пол	Текстовое	1
Дата рождения	Дата	Краткий формат
Группа	Текстовое	3

Рис.1. Структура таблицы Студент.

4. Для определения всех остальных полей таблицы базы данных Sess в соответствии с рис.1 выполните действия, аналогичные указанным в п.3. Если значение типа Текстовый не подходит, то нажмите кнопку с черным треугольником в этом же поле (это кнопка раскрытия списка) и выберите нужный тип данных.

5. Сохраните таблицу в вашей папке под именем Студент.

6. Закройте базу данных, выполнив команду Файл, Выход.

Практическая работа №2.
Ввод и редактирование данных в режиме таблицы.

В созданную таблицу данные могут быть введены как непосредственно в табличной форме по умолчанию, так и с использованием специально разработанной пользовательской экранной формы. Редактирование записей и исправление ошибок в данных таблицы возможно также в каждом из двух указанных режимов.

Ввод данных

Вдоль верхнего края окна расположены имена полей таблицы. Каждое поле соответствует определенному столбцу в таблице. Каждая запись занимает одну строку таблицы. Ввод в определенную ячейку таблицы (выделенную курсором) осуществляется путем набора информации на клавиатуре и последующим нажатием клавиши Enter или Tab. При окончании ввода данных в последнее поле записи Access сам переходит на первое поле новой записи и ожидает ввода данных. Заполнения с клавиатуры требуют все поля, кроме тех, тип которых определен как Счетчик.

Перемещение в таблице.

Для быстрого просмотра данных, введенных в таблицу, а также необходимого позиционирования в таблице нужно обратить внимание на возможности быстрого перемещения в таблице.

Первая запись – Щелчок мышью по кнопке на панели инструментов Первая запись.

Последняя запись - Щелчок мышью по кнопке на панели инструментов Последняя запись

Первый столбец таблицы – клавиша Home.

Последний столбец таблицы – клавиша End.

Следующий столбец справа - одна из клавиш Right, Enter, или Tab.

Следующий столбец слева – клавиша Left, или Shift + Tab.

На строку вверх – Up.

На строку вниз – Down.

Вверх на 26 строк – PgUp.

Вниз на 26 строк – PgDn.

В левый верхний угол таблицы – Ctrl + Home.

В правый нижний угол таблицы – Ctrl + End.

Редактирование данных.

Редактировать данные ячейки таблицы можно как с полной, так и с частичной их заменой.

Для полной замены данных необходимо подвести курсор к редактируемой ячейке так, чтобы все ее содержимое было высвечено в реверсивном виде, а затем набрать нужную информацию.

Частичную замену данных можно осуществить двумя способами:

Во-первых, щелкнуть в нужной ячейке, и она автоматически откроется для редактирования;

Во-вторых, используя клавиши, переместиться в нужную ячейку, а затем нажать функциональную клавишу F2.

Удаление записи.

Для удаления записи ее необходимо выделить (щелкнуть по области маркировки записи) и либо нажать клавишу Delete, либо выполнить команду Правка, Удалить строки. В выводимом на экран запросе подтвердить удаление.

Задание к лабораторной работе.

1) Введите данные, представленные на рис.2, в таблицу СТУДЕНТ, созданную в предыдущем задании.

2) Пользуясь информацией, приведенной в данной лабораторной работе, ознакомьтесь с возможностями редактирования данных в табличном режиме.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess, выполнив следующие действия:

a) Пуск, Программы, Microsoft Access

b) В появившемся диалоговом окне Microsoft Access в перечне списка существующих баз данных, выберите Sess и нажмите ОК

2. Откройте таблицу СТУДЕНТ базы данных Sess. Для этого:

a) В окне Sess: База данных выберите вкладку Таблица

b) В этом же окне нажмите кнопку Открыть

3. Введите данные в таблицу в соответствии с рис.2. при вводе данных воспользуйтесь информацией, приведенной в данной лабораторной работе.

Номер	**Фамилия**	**Имя**	**Отчество**	**Пол**	**Дата рождения**	**Группа**
16493	Сергеев	Петр	Михайлович	м	1.01.76	111
16593	Петрова	Анна	Юрьевна	ж	15.03.75	112
16693	Анохин	Андрей	Борисович	м	24.02.75	112
16793	Борисова	Мария	Михайловна	ж	14.04.76	111
16893	Зайцев	Сергей	Алексеевич	м	29.07.76	111
16993	Кравцов	Алексей	Иванович	м	9.09.75	112
17093	Волкова	Светлана	Николаевна	ж	7.12.76	111

Рис.2. Пример таблицы для ввода данных в табличном режиме.

4. Познакомьтесь с возможностями быстрого перемещения в таблице, используя информацию, приведенную в данной лабораторной работе.

5. Отредактируйте введенные в таблицу данные, используя информацию, приведенную в данной лабораторной работе:

a) В поле Фамилия второй записи таблицы полностью замените Петрова

на Морозова.

b) В поле Дата рождения первой записи таблицы замените цифры года 76 на 75

c) Удалите последнюю запись таблицы

6. Сохраните таблицу и закройте таблицу, выполнив Файл, Выход.

Практическая работа №3.
Разработка однотабличных пользовательских форм.

Данные в таблицу базы данных вводить и редактировать намного удобнее, если воспользоваться экраном в виде некоторого бланка, формы. Такой способ ввода позволяет видеть на экране все данные одной записи и вводить дополнительный текст, поясняющий значение каждого поля. Можно создать форму, напоминающую печатную форму, расположить в ней окна списков, фотографии, графики и др.

Access располагает мастером по разработке форм пяти видов:

- Автоформа: в столбец – поля выводятся на экран в виде последовательности строк.
- Автоформа: табличная – поля выводятся в виде строк и столбцов.
- Диаграмма – для ее создания выбирается таблица, содержащая числовые значения, которые можно представить в графическом виде.
- Сводная таблица – объединяет в себе данные более одной таблицы базы данных. Позволяет просмотреть и изменить данные в нескольких таблицах одновременно.

I. Ввод данных с использованием формы.

При вызове на экран окна формы, в которое можно вводить новые записи, выберите пункт меню Записи, Ввод данных, а затем в появившемся подменю позиция Новая.

Access создает новую незаполненную запись после последней записи таблицы. Новая запись выводится в виде формы с пустыми полями, с курсором в первом поле. Данные вводятся в каждое поле, не определенное с типом Счетчик. Переход от одного поля к другому осуществляется нажатием клавиши Tab.

II. Перемещение в режиме формы.

Основные способы перемещения:

- переход к первой записи – щелкнуть по кнопке Первая запись;
- переход к последней записи – щелкнуть по кнопке Последняя запись;
- переход к следующей записи – щелкнуть по кнопке Следующая запись или нажать PgUp;
- переход к предыдущей записи – щелкнуть по кнопке Предыдущая запись или нажать PgDn;
- переход к определенной записи по ее номеру – щелкнуть в строке Запись

и удалить находящийся в ней номер, затем ввести с клавиатуры номер нужной записи.

Задание к лабораторной работе.

1. Создайте однотабличную пользовательскую форму для ввода и редактирования данных таблицы Студент, как это показано на рис.3.
2. Ознакомьтесь с возможностями ввода данных в форму.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess. Для этого:

c) Пуск, Программы, Microsoft Access

d) В появившемся диалоговом окне Microsoft Access в перечне списка существующих баз данных, выберите Sess и нажмите ОК

2. Создайте простую форму. Для этого:

a) В окне Sess: база данных щелкните по вкладке Формы

b) В том же окне нажмите кнопку Создать

c) В диалоговом окне Новая форма в поле Выберите в качестве источника данных таблицу или запрос введите имя таблицы Студент

d) В этом же окне щелкните Автоформа: в столбец и щелкните ОК

3. На экране появится окно с выводом данных из таблицы в виде формы. Вид полученной в результате проделанных операций формы представлен на рис.3.
4. Добавьте в таблицу запись в режиме формы, используя информацию, приведенную в данной лабораторной работе
5. Познакомьтесь с возможностями перемещения в таблице, представленной в виде формы, используя информацию, приведенную в данной лабораторной работе:
6. Сохраните созданную форму, для этого:

a) Выполните команду Файл, Сохранить

b) В диалоговом окне Сохранение в поле Имя формы наберите имя Форма1

c) Нажмите ОК

7. Закройте таблицу, выбрав команду меню Файл, Выход

Студент

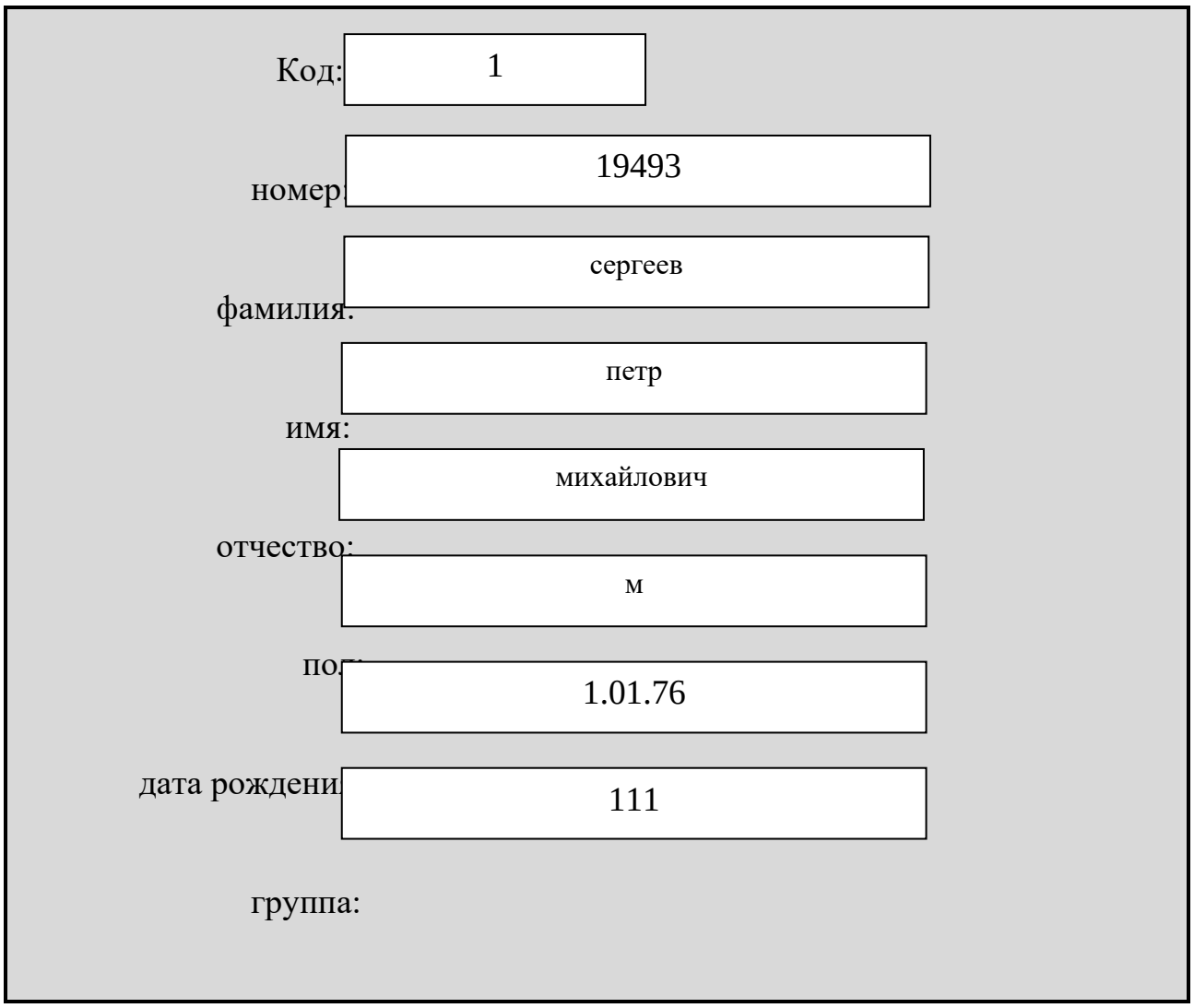


Рис.3. Пример формы.

Практическая работа №4.
Вывод данных на печать.

Access выводит информацию из базы данных в виде отчета (распечатки содержимого базы данных). Все отчеты подразделяются на три категории:

- простая распечатка содержимого базы данных из режимов таблицы или формы
- детальные отчеты – хорошо подготовленные отчеты, представленные в любом удобном для пользователя виде и включающие в себя ряд дополнительных элементов
- специальные отчеты – позволяют подготавливать почтовые наклейки и формы писем

Способ вывода на печать таблиц и форм удобен для построения быстрого чернового варианта отчета.

Преимущество таких отчетов – быстрота и простота их получения. Недостатком является вывод данных точно в таком же виде, в котором они содержатся в таблице или форме.

При выводе на печать данных из табличного режима с целью улучшения вида распечатки можно использовать следующие возможности изменения внешнего вида таблицы:

- уменьшить ширину некоторых столбцов таблицы
- скрыть некоторые столбцы
- поменять ориентацию страницы с Книжной на Альбомную

При распечатки данных с использованием формы Access выводит на странице столько записей, сколько на ней может поместиться. Возможен вариант, когда часть одной записи разместится в конце одной страницы, а другая – в конце следующей.

Задание к лабораторной работе.

Выведите на печать из режимов таблицы и формы содержимое таблицы СТУДЕНТ, созданной в предыдущих заданиях.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess.
2. Откройте таблицу Студент в табличном режиме.
3. Выведите данные таблицы СТУДЕНТ на экран из табличного режима, для чего необходимо выбрать команду Файл, Предварительный просмотр.
4. Для получения распечатки данных из таблицы выполните команду меню Файл, Печать.
5. Закройте окно таблицы в виде строк и столбцов, используя кнопку системного меню в левом верхнем углу окна таблицы. На запрос Access о сохранении изменений в таблице ответьте отрицательно.
6. Откройте таблицу СТУДЕНТ в режиме формы.
7. Выведите данные таблицы СТУДЕНТ на экран из режима формы, для чего необходимо выбрать команду Файл, Предварительный просмотр.
8. Распечатайте данные из режима формы, для этого выполните команду меню Файл, Печать.
9. Закройте базу данных, выбрав команду меню Файл, Выход.

Практическая работа №5.
Разработка детального отчета.

Для получения отчета улучшенного внешнего вида необходимо подготовить детальный отчет. Он должен иметь наглядную форму и содержать больше информации, чем простая распечатка таблицы. Целесообразно для создания обычного детального отчета использовать мастера отчетов.

Access включает следующие мастера отчетов:

- Автоотчет: в столбец
- Автоотчет: ленточный
- Мастер диаграмм
- Почтовые наклейки

При создании простого отчета выводятся все поля и записи из таблицы или запроса, причем каждое поле – на отдельной строке. При выборе мастера отчета требуется определить стиль отчета (Строгий, Доклад, Табличный), его ориентацию на странице (Книжная, Альбомная), присвоить имя отчету и ввести заголовок отчета.

Задание к лабораторной работе.

С помощью мастера создайте детальный отчет для вывода данных таблицы СТУДЕНТ. Вид отчета представлен на рис. 4.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess.
2. Создайте отчет для вывода данных таблицы СТУДЕНТ с помощью мастера. Для этого:

a) В окне Sess: база данных нажмите Отчет

b) В диалоговом окне Новый отчет в поле Выберите в качестве источника данных таблицу или запрос введите или выберите из списка имя таблицы СТУДЕНТ

c) Щелкните по кнопке Мастер отчетов

d) Нажмите кнопку ОК

e) В диалоговом окне Создание отчета в списке полей Доступные поля щелкните поле Фамилия, а затем по кнопке > для перемещения поля в список полей, выбранных для создания отчета

f) Аналогичным образом выберите для включения в отчет поля Имя, Отчество, группа. По окончанию этой операции щелкните на кнопке Далее

g) Откроется новое окно мастера отчетов, в котором будет выделено поле Фамилия. Щелкните поле Далее

h) В следующем окне мастера отчетов, где предлагается выбрать данные для сортировки, по которым будет упорядочена выводимая в отчет информация, щелкните по черному треугольнику в поле, в котором находится курсор. Из раскрывшегося списка выберите поле Фамилия и щелкните кнопку Далее

i) В следующем окне выберите Макет Табличный, Ориентация Книжная и щелкните кнопку Далее

j) В следующем окне выберите стиль отчета Строгий и щелкните кнопку Далее

k) В следующем диалоговом окне мастера отчетов введите заголовок отчета Студент и щелкните на кнопке Готово

После этого Access выходит в окно предварительного просмотра отчета

3. Выведите на печать полученный отчет, для чего нажмите в пиктографическом меню кнопку Печать. Вы должны получить отчет, вид которого показан на рис. 4.

СТУДЕНТ

Фамилия	**Имя**	**Отчество**	**Группа**
Анохин	Андрей	Борисович	112
Борисова	Мария	Михайловна	111
Зайцев	Сергей	Александрович	111
Кравцов	Алексей	Иванович	112
Морозова	Алла	Владимировна	112
Сергеев	Петр	Михайлович	111

Рис. 4. Пример отчета.

4. Сохраните созданный отчет. Для этого:

a) Выполните команду Файл, Сохранить

b) Наберите имя отчета СТУДЕНТ

c) Нажмите ОК

5. Закройте базу данных, выбрав команду меню Файл, Закрыть

Практическая работа № 6.
Поиск и отбор данных.

Access предоставляет довольно широкий спектр возможностей для поиска и отбора информации в базе данных. К таким средствам можно отнести использование команды Поиск, фильтрацию, сортировку, создание и использование запросов.

Простейшим способом поиска информации в базе данных является использование директивы Поиск. Этот поиск может проводиться как в одном из указанных полей, так и во всех полях таблицы базы данных. Возможно изменение порядка просмотра записей в таблице.

Обычно поиск по этой директиве начинается с активного места таблицы (активной записи, активного поля). Для просмотра всей таблицы необходимо перейти к первой записи, а затем начать поиск.

Для того, чтобы записи в таблице выстраивались при выводе в удобном для пользователя порядке, используется сортировка. Access может проводить сортировку по одному полю, по нескольким полям, по возрастанию или убыванию значений ключевого признака.

Для вывода только определенных записей таблицы используется фильтрация.

Задание к лабораторной работе.

Для данных, содержащихся в таблице СТУДЕНТ, в режиме формы осуществите поиск одной из записей, в режиме таблицы отсортировать записи по возрастанию значений одного из полей и отфильтровать данные в соответствии с критерием отбора.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess
2. Откройте таблицу СТУДЕНТ в режиме формы. Для этого:

a) В диалоговом окне Sess: база данных нажмите кнопку Формы, выберите форму Форма1

b) В том же окне нажмите кнопку Открыть

3. Найдите запись таблицы с информацией о студентке с фамилией Морозова. С этой целью выполните следующую группу действий:

a) Находясь в форме Форма1, щелкните в строке поля Фамилия. Щелкните на кнопке Найти или выполните команду меню Правка, Найти

b) В диалоговом окне Поиск в поле ‘ Фамилия’ введите в поле Образец слово Морозова

c) Щелкните на кнопке Найти. В форму выведется найденная запись

d) Закройте окно формы, для чего необходимо щелкнуть на кнопке Закрыть. На экране появится окно базы данных

4. Откройте таблицу СТУДЕНТ в табличном режиме. Для этого необходимо выполнить следующие операции:

a) В окне Sess: база данных перейдите на вкладку Таблица и выберите таблицу СТУДЕНТ

b) Нажмите кнопку Открыть

5. Отсортируйте записи таблицы в соответствии с алфавитным порядком фамилии студентов, сто потребует от вас следующих действий:

a) Щелкните на столбце Фамилия

b) Щелкните по кнопке пиктографического меню Сортировка по возрастанию или выберите пункт меню Записи, Сортировка, Сортировка по возрастанию

c) Записи таблицы будут выведены на экран в соответствии с алфавитным порядком фамилий

6. Используя фильтрацию для вывода на экран только записей, относящихся к студентам, родившимся после 1975 года. Для этого можно выполнить следующий порядок действий:

a) В окне с таблицей СТУДЕНТ щелкните на кнопке Изменить фильтр или выберите пункт меню Записи, Фильтр, Изменить фильтр

b) В окне фильтра выберите поле с именем Дата рождения (щелкните по этому полю)

c) Наберите выражение > 31.12.75

d) Щелкните на кнопке Применить фильтр или выберите пункт меню Фильтр, Применить фильтр

e) На экран выведутся только записи, соответствующие введенному критерию отбора

7. Удалите фильтр (выведите снова все записи таблицы), для чего щелкните по кнопке Удалить фильтр или выберите пункт меню Записи, удалить фильтр

8. Закройте базу данных, выбрав команду меню Файл, Закрыть

Практическая работа № 7.
Формирование запросов.

I. Общие сведения.

В Access поиски отбор любой нужной информации можно производить с использованием запросов, имеющих большие возможности, чем рассмотренные ранее средства. Запросы используются примерно так же, как таблицы.

Запрос представляет собой вопрос о данных, хранящихся в таблицах, или инструкцию на отбор записей, подлежащих изменению.

В Access имеется возможность самостоятельно создать запрос или воспользоваться мастером по разработке запросов. Чаще всего запрос разрабатывается самостоятельно. Однако для создания специальных запросов возможно применение мастера.

Самым распространенным типом запроса является запрос на выборку. Для подготовки запроса необходимо определить:

- поля, по которым будет проводиться поиск
- искомое значение

- поля, выводимые в результате выполнения запроса

II. Выражения в запросах.

Для указаний условий отбора данных и для создания вычисляемых полей в запросах используются выражения.

Выражения представляют собой формулы, по которым вычисляются необходимые значения. Различаются арифметические и логические выражения. Выражения могут состоять из следующих элементов:

- литералов
- операторов
- констант
- идентификаторов
- функций

Литерал – это точное значение, которое Access использует именно в том виде, как оно вводится. При записи литерала используются специальные символы-ограничители, которые указывают на тип данных литерала.

Если литерал – число, то он вводится без ограничений. Например, 465.8.

Текстовый литерал должен иметь в качестве ограничителя кавычки или апостроф («» или ‘’). Например, «Иванов» или ‘Иванов’.

В литералах типа дата используется ограничитель #. Например, #12/11/96#. В случае литерала типа поле или элемента управления вводятся ограничители [ ]. Например, [Фамилия].

Оператор указывает действие, которое должно быть выполнено с элементами выражения.

Выделяются следующие группы операторов:

- арифметические: * умножение, + сложение, - вычитание, / деление, ^ возведение в степень
- соединение частей текста &, например, =[Фамилия] & “ “ &[Имя]
- сравнения: < меньше, <= меньше или равно, > больше, >= больше или равно, = равно, <> не равно
- логические: And (И), Not (Нет), Or (Или)
- операторы SQL:
    - Link – для использования логики замены в выражениях,
    - In – для определения, содержится ли элемент данных в списке значений
    - Between…And – для выбора значений из определенного интервала

Константа – это неизменяемая величина. К наиболее часто используемым константам относятся Null (соответствует полю, не содержащему значений или символов), Истина, Ложь.

Идентификатор - это имя, введенное в выражение для резервирования места под значение, которое хранится в поле или элементе управления. На основе использования идентификаторов можно создавать выражения, которые исполь-

зуют информацию, хранящуюся в таблицах, формах, отчетах. Идентификаторы обычно заключаются в [ ]. Например, [Дата] относится к значению поля Дата таблицы СТУДЕНТ.

Функция – это специальное имя, которое используется для выполнения какой-либо операции и может применяться в выражениях. В Access встроено несколько десятков функций. Аргументы функции должны заключаться в ( ). Скобки могут быть опущены только при нулевом аргументе. Примерами функций, используемых при построении выражений в запросах, могут служить :

- Avg( ) – среднее арифметическое значение
- Count( ) – количество записей
- Sum( ) – сумма всех записей и т.д.

Задание к лабораторной работе.

Сформируйте запрос-выборку, позволяющий получить из таблицы СТУДЕНТ данные о студентах мужского пола, родившихся после 1975 года.

Выполнение работы.

1. Откройте ранее созданную базу данных Sess
2. Создайте новый запрос. Для этого:

a) В окне Sess: база данных щелкните по вкладке Запрос

b) В том же окне нажмите кнопку Создать

c) В диалоговом окне Новый запрос щелкните по полю Конструктор

d) В диалоговом окне Добавление таблицы выберите таблицу Студент и нажмите кнопку Добавить

e) После появления в окне Запрс1: запрос на выборку списка полей таблицы СТУДЕНТ щелкните по кнопке Закрыть в диалоговом окне Добавление таблицы

f) В первую ячейку строки Поле перетащить из списка полей таблицы СТУДЕНТ поле Фамилия, во вторую – Имя, в третью – Отчество, в четвертую – дата рождения, в пятую – пол из окна Студент

g) В пятую ячейку строки Условие отбора поместить выражение = «м» и убрать признак вывода на экран информацию из этого поля (щелкнуть по квадратику с галочкой в этом поле)

h) В четвертую ячейку строки Условие отбора поместить выражение >#31.12.75# и установить признак вывода на экран информации из данного поля

3. Выполнить запрос, для чего щелкнуть на кнопке пиктографического меню Запуск или выбрать пункт меню Запрос, Запуск
4. Сохранить запрос, для этого выполнить команду Файл, Сохранить. В появившемся после этих действий окне Сохранение введите имя запроса, например, можно оставить имя Запрос1, предлагаемое по умолчанию
5. Закройте базу данных, выбрав команду меню Файл, Выход

Практическая работа № 8.
: Создание многотабличной информационной базы данных (ИБД).

I. Разработка информационно-логической модели (ИЛМ), создаваемой ИБД.

Основу любой автоматизированной информационной системы (АИС) составляет ИБД, содержащая всю необходимую информацию для решения задач, стоящих перед автоматизированной системой.

ИБД формируется средствами систем управления базами данных (СУБД) на машинных носителях: винчестере, магнитных дисках (дискетах). ИБД состоит из информационных массивов, созданных в процессе проектирования ИБД. Массив состоит из отдельных записей. Запись состоит из полей. Для выделения информационных массивов необходимо построить инфологическую модель предметной области.

Предметная область (ПО) – это все предметы и события, которые составляют основу информации, необходимой для решения задач АИС.

Предметная область представляется исследователю состоящей из объектов реальных и абстрактных, называемых сущностями.

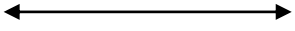
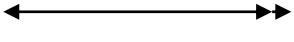
Пусть необходимо построить автоматизированную информационно-справочную систему «Студент».

Помещаем в центре предметной области сущность «студент». Сущность «студент» порождает сущности «сессия» и «стипендия» (см. рис. 5).

Каждая сущность описывается атрибутами. Атрибут – признак, описатель сущности, содержащий одну из характеристик: код сущности, наименование, тип, вид, количество и т.д.

Атрибуты бывают первичные (основные) и непервичные. Первичные входят в состав ключа, это, как правило, коды сущностей.

Сущности инфологической модели взяты в прямоугольную рамку. Атрибуты, описывающие сущности, перечислены под сущностями. Первичные атрибуты, входящие в состав ключа, подчеркнуты. Между сущностями устанавливаются связи в виде стрелок с наименованием действия.

Связь типа 1 : 1 (одна к одной) 
Связь типа 1 : М (одна ко многим) 
Связь типа М : М (многие ко многим) 

Например, связь между сущностями «стипендия» и «сессия» – это связь одна ко многим, так как данный процент стипендии может быть начислен по результатам сессии многим студентам.

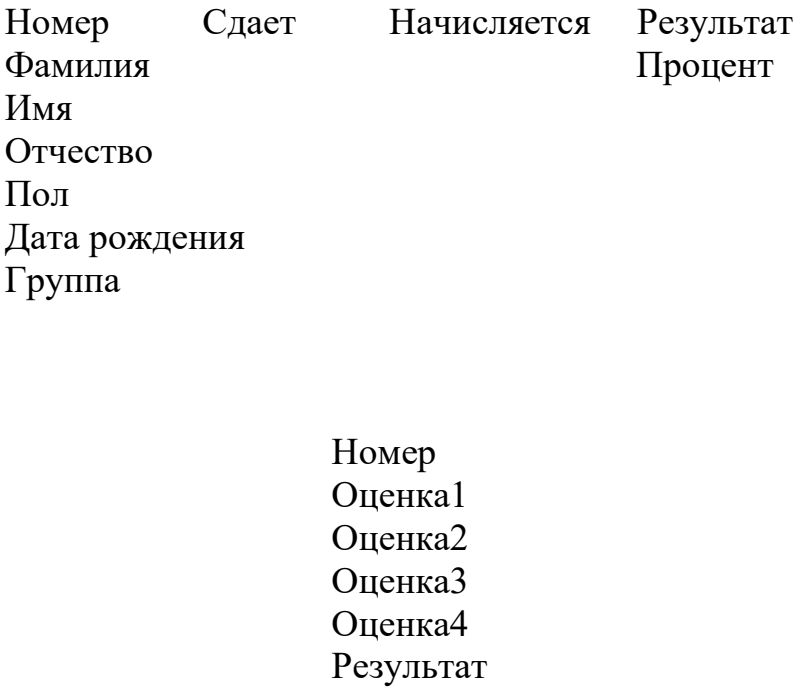


Рис. 5. Инфологическая модель ПО «Студент».

На рисунке 5 представлено графическое изображение ИЛМ. Может быть дано описательное представление ИЛМ.

СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа)
СЕССИЯ (Номер, Оценка1, Оценка2, Оценка3, Оценка4, Результат)
СТИПЕНДИЯ (Результат, Процент)

СТУДЕНТ ⟷ СЕССИЯ (1 : 1)
СТИПЕНДИЯ ⟷ СЕССИЯ (1 : M)

II. Построение модели отношений в предметной области.

Каждая сущность ИЛМ описывается таблицей атрибутов, называемой отношением. Строки таблицы – это кортеж, совокупность кортежей – это отношение, т.е. все строки таблицы. Столбец – это домен, т.е. все атрибуты одного типа в отношении. Каждое отношение имеет столько доменов, сколько атрибутов описывают сущность.

Отношение «Студент» представлено ранее на рисунке 2, аналогично строятся таблицы (отношения) «Сессия» и «Стипендия».

Чтобы реализовать отношение в ИБД необходимо для каждой таблицы построить спецификацию, т.е. описать ее структуру таким образом, как это было сделано в лабораторной работе № 1. Каждый реквизит описывается как поле, имеющее свое уникальное имя. Это имя может совпадать с именем реквизита и может быть сокращено. Для дальнейшей работы удобнее, чтобы имя поля совпа-

дало с именем реквизита. Каждая таблица должна иметь не повторяющееся ключевое поле, которое подлежит обязательному индексированию с отметкой (совпадения не допускаются).

На рисунках 6, 7, 8 приведены спецификации всех трех таблиц.

Признак ключа	**Поле**	**Тип поля**	**Размер поля**
ключ	**Номер**	Текстовое	5
	Фамилия	Текстовое	15
	Имя	Текстовое	10
	Отчество	Текстовое	15
	Пол	Текстовое	1
	Дата рождения	Дата	Краткий формат
	Группа	Текстовое	3

Рис. 6. Структура таблицы СТУДЕНТ.

Признак ключа	**Поле**	**Тип поля**	**Размер поля**
Ключ	Номер	Текстовое	5
	Оценка1	Числовое	Байт
	Оценка2	Числовое	Байт
	Оценка3	Числовое	Байт
	Оценка4	Числовое	Байт
	Результат	Текстовое	3

Рис. 7. Структура таблицы СЕССИЯ.

Признак ключа	**Поле**	**Тип поля**	**Размер поля**
Ключ	Результат	Текстовое	3

	Про-цент	Число-вое	С плавающей точ-кой (4 байт)

Рис.8. Структура таблицы СТИПЕНДИЯ.

Следующим этапом проектирования служит построение модели отношений в ПО – даталогической модели.

Рис. 9. Даталогическая модель ПО.

На рисунке 9 показаны связи по внешним ключам, которые будут реализованы в лабораторной работе № 9.

Реляционный подход к проектированию ИЛМ базируется на понятии нормализации. Теория нормализации основана на том, что определенные наборы таблиц (отношений) в наилучшей степени отражают свойства предметной области и в то же время обнаруживают лучшие качества по отношению к другим наборам таблиц в процессе манипулирования. Спроектированные в данной задаче таблицы содержат только простые, далее неделимые данные (находятся в первой нормальной форме), выполняется условие функционально-полной зависимости не ключевых атрибутов от ключа (находятся во второй нормальной форме), отсутствует транзитивная зависимость не ключевых атрибутов от ключевых или зависимости между не ключевыми атрибутами (находятся в третьей нормальной форме).

Связи между атрибутами реализуются объединением атрибутов в таблицу.

Связи между объектами в реляционной базе не хранятся, а образуются в процессе манипулирования.

III. Создание многотабличной базы данных физической модели.

СУБД Access может обрабатывать данные различных таблиц базы данных. Для этого пользователю необходимо при формировании каждой из этих таблиц базы данных установить ключ (определить ключевое поле), а затем создать связи между таблицами.

В лабораторной работе №1 рассказывается о том, что при создании любой таблицы необходимо определение состава, типа и размера полей, составляющих таблицу.

В случае, если база данных содержит несколько таблиц, необходимо также определение ключа для каждой таблицы.

Access создает индекс для ключевого поля таблицы и использует его для поиска записей и объединения таблиц в запросе. Ключевое поле не может содержать пустых и повторяющихся значений.

Таблицу, в которой не определен ключ, нельзя использовать при установке связей, кроме того, поиск и сортировка в такой таблице выполняется медленнее.

Задание к лабораторной работе.

1. В лабораторной работе №1 была описана технология создания таблицы СТУДЕНТ базы данных SESS. Структура этой таблицы полностью соответствует информации, приведенной на рис.1. В соответствии с постановкой задачи пополнить базу данных SESS еще двумя таблицами СЕССИЯ И СТИПЕНДИЯ.
2. Создайте структуру таблиц СЕССИЯ и СТИПЕНДИЯ, а в ранее созданной таблице СТУДЕНТ установите ключевое поле в соответствии с рис. 6, 7 и 8. Заполните вновь созданные таблицы СЕССИЯ и СТИПЕНДИЯ данными, как показано на рис. 9 и 10.

Выполнение работы.

1. Откройте ранее созданную базу данных SESS.MDB
2. Выведите таблицу СТУДЕНТ в режиме конструктора. Для этого:

a) В окне SESS: база данных перейдите на вкладку Таблица

b) В том же окне нажмите кнопку Конструктор

Access переходит в режим конструктора таблиц и выводит экран диалогового окна Студент: таблица с перечнем полей и их свойств.

3. Определите ключ таблицы СТУДЕНТ. Для определения ключа необходимо в окне конструктора таблиц выделить поле Номер и нажать кнопку панели инструментов Ключевое поле (значок в виде ключа), в результате чего в разделителе строк появляется меленькое изображение ключа. Окончательный состав полей таблицы и их свойств приведен на рис. 6.
4. Создайте структуру таблиц СЕССИЯ и СТИПЕНДИЯ, пользуясь описанием технологии создания новых таблиц базы данных в лабораторной работе №1.

Состав полей и их свойства приведены на рис.7 и 8.

5. Заполните данными, показанными на рис. 10 и 11, вновь созданные таблицы СЕССИЯ и СТИПЕНДИЯ, используя информацию, приведенную в лабораторной работе № 2.

Номер	**Оценка1**	**Оценка2**	**Оценка3**	**Оценка4**	**Результат**
16493	5	4	3	4	Нхр

16593	4	4	4	4	Хор
16693	5	5	5	5	Отл
16793	4	5	5	5	Хр1
16893	3	4	3	4	Нхр
16993	5	5	5	5	Отл
17093	4	4	5	4	Хор

Рис. 10. Данные таблицы Сессия.

Результат	**Процент**
Нхр	0,00
Хор	100,00
Хр1	150,00
Отл	200,00

Рис. 11. Данные таблицы СТИПЕНДИЯ.

6. Закройте базу данных, выбрав команду меню Файл, Выход.

Практическая работа № 9.
: Установление связей между таблицами.

Для сформированных таблиц с установленным ключевым полем в каждой из них возможно создание определенных взаимоотношений. Access использует эти взаимоотношения для связывания данных в каждом новом запросе, форме или отчете, включающем связанные таблицы.

Создать связь между таблицами можно, если в них есть совпадающие поля. Ключевое поле первой таблицы должно соответствовать аналогичному полю связанной таблицы. Если связанная таблица не содержит такого поля, то его необходимо добавить.

Задание к лабораторной работе.

Используя возможности Access, установите связи между созданными таблицами СТУДЕНТ, СЕССИЯ и СТИПЕНДИЯ базы данных SESS.

Выполнение работы.

1. Откройте ранее созданную базу данных SESS.MDB. В диалоговом окне Sess: база данных появляется список таблиц: СЕССИЯ, СТИПЕНДИЯ, СТУДЕНТ.
2. Выполните команду Сервис, Схема данных… . После этих действий на экран выводится окно Схема данных… .
3. Добавьте таблицы в окно Схема данных… . Для этого:

a) Выполните команду Связи, Добавит таблицу…

b) В окне Добавление таблицы из списка выбрать Студент и нажать кнопку Добавить

c) Далее из списка выбрать Сессия и нажать кнопку Добавить

d) Аналогично выберите Стипендия и нажмите кнопку Добавить. Таблицы расположите так, как это показано на рис. 12.

4. Установите связи между таблицами СТУДЕНТ и СЕССИЯ. Для этого:

a) Протащите указатель мыши от поля Номер таблицы СТУДЕНТ к полю Номер таблицы Сессия

b) В появившемся диалоговом окне Связи установите флажок Обеспечение целостности данных, выберите отношение Один-к-одному и нажмите кнопку Создать

5. Установите связь между таблицами СТИПЕНДИЯ и СЕССИЯ. Для этого:

a) Протащите указатель мыши от поля Результат таблицы СТИПЕНДИЯ к полю Результат таблицы СЕССИЯ

b) В появившемся диалоговом окне Связи установите флажок Обеспечение целостности данных, выберите отношение Один-ко-многим и нажмите кнопку Создать

c) В результате описанных действий окно Схема данных приобретает вид, показанный на рис.13.

d) В таблице Студент появилось поле код, созданное ACCESS для внутренней физической структуры связи.

6. Сохраните установленные между таблицами связи и выйдите из режима схемы данных. Для этого выберите пункт меню Файл, Сохранить, а затем Файл, Выход. На экране остается открытое окно базы данных

7. Закройте базу данных. Для этого выполните команду меню Файл, Выход

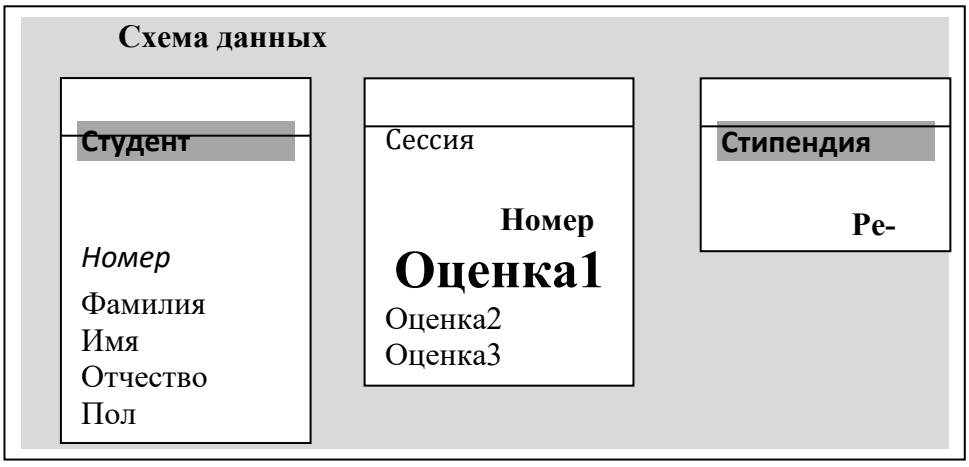


Рис. 12. Расположение таблиц в окне схемы данных.

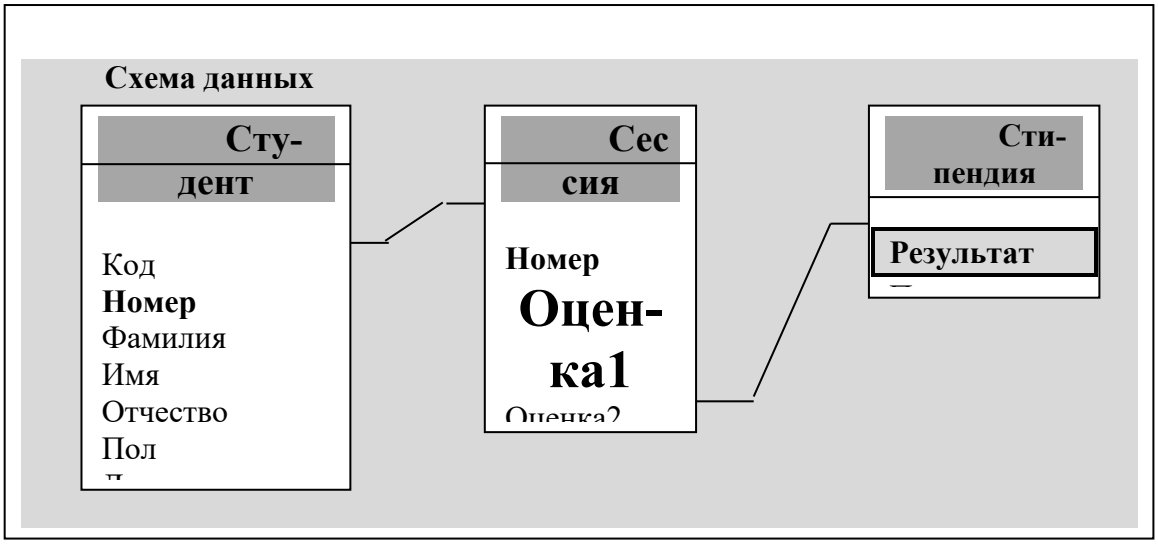


Рис.13. Установленные связи между таблицами.

Практическая работа №10.
: Формирование запросов для многотабличной базы данных.

Для получения определенных данных из базы данных пользователь может построить запрос. Результатом выполнения запроса является таблица с временным набором данных (динамический набор). Записи динамического набора могут включать поля из одной или нескольких таблиц. Запросы используются аналогично таблицам. Соответствующий динамический набор можно просмотреть в табличном представлении. На основе запроса можно построить отчет или форму. При обновлении данных в динамическом наборе возможно включение этих изменений в таблицы. Использование запросов позволяет осуществить различные формы доступа к одной и той же информации.

Access обеспечивает корректную связь между таблицами базы данных за счет ключей (значений эквивалентных полей).

При включении в запрос связанных таблиц базы данных в окно Запрос-выборка между ними автоматически возникает соединительная линия.

Если между таблицами, включенными в запрос, нет связи и она не возникает автоматически, можно соединить таблицы в окне Запрос-выборка. Для этого необходимо наличие в них полей с совпадающими данными. Однако надо учесть, что такое соединение сохраняется лишь для данного запроса и при использовании этих же таблиц в новом запросе требуется соединить их заново.

Задание к лабораторной работе.

Постройте запрос, позволяющий выводить фамилию, имя, отчество и номер группы студентов, которым может быть назначена стипендия, а также размер назначаемой стипендии. Эти данные могут быть использованы при создании проекта приказа назначения студентов на стипендию по результатам экзаменационной сессии. Информация для получения таких данных содержится в трех связанных таблицах СТУДЕНТ, СЕССИЯ и СТИПЕНДИЯ базы данных SESS.

Выполнение работы.

1. Откройте ранее созданную базу данных SESS.MDB. в диалоговом окне SESS: база данных появляется список таблиц: СЕССИЯ, СТИПЕНДИЯ, СТУДЕНТ.

2. Создайте новый запрос на основе связанных таблиц. Для этого:

a) Перейдите на вкладку Запросы и нажмите кнопку Создать

b) В диалоговом окне Новый запрос нажмите поле Конструктор, после чего появляется окно Запрос1: запрос на выборку

c) В окне Добавление таблицы выделите в списке Таблица/Запрос таблицу СТУДЕНТ и щелкните на кнопку Добавить

d) В том же списке выделите и добавьте таблицы СЕССИЯ и СТИПЕНДИЯ. Закройте диалог щелчком по кнопке Закрыть

e) Списки полей всех выбранных таблиц появляются в верхней части окна Запрос1: запрос на выборку. Между этими списками автоматически возникает соединительная линия, так как между таблицами уже установлена связь

3. Присвойте запросу имя. Для этого выберите команду меню Файл, Сохранить и в окне Сохранение введите имя ПРОЕКТ ПРИКАЗА. Нажмите ОК

4. Включите поля из трех таблиц в запрос. Из таблицы СТУДЕНТ в бланк запроса по образцу (в строку Поле) перетащите следующие поля: Фамилия, Имя, Отчество, Группа. В следующее поле в запросе перетащите поле Процент из таблицы СТИПЕНДИЯ

5. Установите условие отбора. Для отбора студентов, подлежащих назначению на стипендию, необходимо в строке Условие отбора под полем Процент ввести выражение >0.

6. Упорядочите выводимые в запросе данные по полю Фамилия в алфавитном порядке. Щелкните ячейку в строке Сортировка под полем Фамилия и в появившемся списке выберите По возрастанию

7. Посмотрите сформированную запросом информацию. Для этого щелкните по команде меню Вид, Режим таблицы. Результаты работы запроса будут выглядеть так, как показано на рис.14

8. Закройте режим запроса, выполнив команду меню Файл, Выход

9. Закройте базу данных, выполнив команду меню Файл, Выход

Фамилия	**Имя**	**Отчество**	**Группа**	**Процент**
Анохин	Андрей	Борисович	112	200,00
Борисова	Мария	Михайловна	111	150,00
Волкова	Светлана	Николаевна	111	100,00
Кравцов	Алексей	Иванович	112	200,00
Петрова	Анна	Юрьевна	112	100,00

Рис.14. Просмотр результатов запроса.

ПРАКТИЧЕСКАЯ РАБОТА №11

Разработка простого приложения базы данных

1. Цель работы

1.1.Изучить технологию доступа к данным BDE.

1.2.Научиться разрабатывать приложения базы данных, используя технологию доступа к данным BDE.

2. Приборы и оборудование

2.1. Методические указания.

2.2. Программное обеспечение Delphi.

3. Порядок выполнения работы:

3.1. Изучить основные теоретические сведения (приложение А).

3.2. Создайте новый проект в Delphi.

3.3. Создайте псевдоним Базы данных. Для чего откройте SQL Explorer (Database -> Explore). Затем запустите утилиту BDE Administrator… (Object -> BDE Administrator…). Далее в главном меню утилиты выполните команду Object -> New…

3.4. Выберите драйвер для доступа к базе данных (STANDARD). Переименуйте созданный псевдоним на номер_группы DB.

3.5. Создайте директорию, в которой будет сохранена база данных (В правой части в качестве значения свойства Path указывается путь к базе данных). После чего сохраните созданный псевдоним (Object -> Apply).

3.6. Создайте базу данных, используя утилиту Database Desktop (Tools -> Database Desktop)

3.7. Установите рабочую директорию (File -> Working Directory). В области Aliases выберите только что созданный псевдоним.

3.8. Создайте таблицу Jurnals, в соответствии с приложением 1 (Object -> New… ->Table). В появившемся окне выберите тип таблицы (Paradox 7). В столбце FieldName указываются имена полей создаваемой таблицы. Столбец Type задает их типа. Для строковых полей в столбце Size указывается их размер. А в столбце Key помечаются символом звездочки те поля, которые будут входить в состав первичного ключа.

3.9. Аналогично создайте таблицу Soderjanie.

3.10.Создайте индекс для второй таблицы, для чего в списке Table properties необходимо выбрать значение Secondary Index и нажать кнопку Define. В качестве индекса выберите атрибут kod_jurnal. Сохраните индекс с именем Jurnal_ind. Определите ссылочную целостность между таблицами, для чего в таблице Soderjanie выберите в списке Table properties элемент Referential Integrity и нажмите кнопку Define. Выберите атрибуты, по которым устанавливается целостность, и сохраните ее с именем Jurnal_ref.

3.11. Создайте модуль данных (File -> New… -> Data Module), в котором расположите два компонента TTable, которые находятся на вкладке BDE и два компонента TDataSource. Для обоих компонентов TTable в свойство Database Name выберите имя псевдонима БД и установите свойство TableName. Переименуйте компоненты TTable в соответствии с наименованиями таблиц, используя свойство Name, а компоненты TDataSource JurnalDS и SoderjanieDS соответственно. Свяжите компоненты таблицы и источники данных. При помощи свойства Dataset надо увязать Jurnal с JurnalDS, а Soderjanie с SoderjanieDS. Определите отношение между таблицами, для чего в свойстве MasterSource компонента Soderjanie необходимо указать родительский источник данных. Далее в свойстве MasterFields этого же компонента необходимо указать поля по которым устанавливается связь. Затем на основной форме нужно разместить два компонента TDBGrid, располагающихся на вкладке Data Controls.

3.12. Модуль данных надо подключить к главному модулю приложения одной строкой кода : `user DataModule;`

3.13. Установите связь компонента TDBGrid с компонентом TdataSource1 расположенным в модуле данных при помощи свойства DataSource. Разместите на форме компонент DBNavigator и свяжите его с компонентом TDBGrid через свойство DataSource.

3.14. Проделайте аналогичные действия для TdataSource2. Оформите форму. Запустите и проверьте ее работоспособность.

3.15. Оформите отчет, сделайте выводы о проделанной работе.

4. Содержание отчёта:

4.1. работы.
4.2. Цель работы.
4.3. Приборы и оборудование.
4.4. Выполнение работы.
4.5. Выводы.

5. Контрольные вопросы:

5.1. Для чего предназначен BDE?
5.2. Для чего предназначены утилиты SQL Explorer, BDE Administrator?
5.3. Как загрузить утилиту SQL Explorer, BDE Administrator?
5.4. Что такое псевдоним БД?
5.5. Как создать псевдоним БД?
5.6. Где и как используется псевдоним?
5.7. Что такое Data Module?
5.8. Как подсоединить Data Module к приложению?
5.9. Перечислите визуальные компоненты для работы с базами данных.
5.10. Перечислите невизуальные компоненты для работы с базами данных.
5.11. Перечислите основные свойства компонента Table.
5.12. Перечислите основные свойства компонента DataSource.

ПРИЛОЖЕНИЕ А

Теоретические сведения

1. СРЕДСТВА DELPHI ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ

1.1. Borland Database Engine

Традиционным для системы Delphi способом работы с базами данных является использование процессора баз данных Borland Database Engine (BDE). Разработанный фирмой Borland унифицированный программный интерфейс BDE позволяет выполнять доступ к данным как с использованием традиционного record-ориентированного (навигационного) подхода, так и с использованием set-ориентированного (реляционного) подхода, принятого в SQL-серверах баз данных. Универсальный механизм доступа к данным, которым является BDE, применяется в средствах разработки фирмы Borland (Delphi, C++ Builder), а также в некоторых других продуктах.

BDE инсталлируется вместе с Delphi, обеспечивает доступ к локальным базам данных, расположенным на том же компьютере, и к удалённым базам, расположенным на сервере. BDE предоставляет очень гибкий механизм управления базами данных, позволяющий приложениям, созданным в среде Delphi, получать информацию из баз данных наиболее популярных форматов.

BDE представляет собой набор динамических библиотек и драйверов, обеспечивающих доступ к данным. В составе BDE поставляются стандартные драйверы, обеспечивающие доступ к базам данных Paradox, dBase, FoxPro и текстовым файлам. Эти драйверы устанавливаются автоматически вместе с ядром процессора. Доступ к данным серверов SQL обеспечивает отдельная сис драйверов Borland SQL Links. Эти драйверы нужны при разработке приложений для серверов Oracle, Sybase, Informix и InterBase. Драйверы SQL Links необходимо устанавливать дополнительно. Кроме того, в BDE есть возможность подключения любых драйверов ODBC.

В Delphi реализовано достаточно большое количество технологий доступа к данным, однако общие подходы и последовательность действий при разработке приложений баз данных почти одинаковы.

1.2. Утилиты для работы с базами данных в Delphi

1.2.1. BDE Administrator

Утилита BDE Administrator (bdeadmin.exe) предназначена для конфигурирования BDE, позволяет устанавливать параметры псевдонимов баз данных, драйверов и параметры, общие для всех баз данных. Настройки BDE сохраняются в файле idapi32.cfg. Окно программы содержит две области (рисунок 1). В левой области расположены страницы Databases и Configuration. Правая область используется для вывода сведений об объекте, выбранном слева. На странице Configuration приведены сведения о драйверах баз данных и установках BDE. На странице Databases приведены псевдонимы имеющихся на компьютере баз данных. Здесь же можно создавать и редактировать псевдонимы.

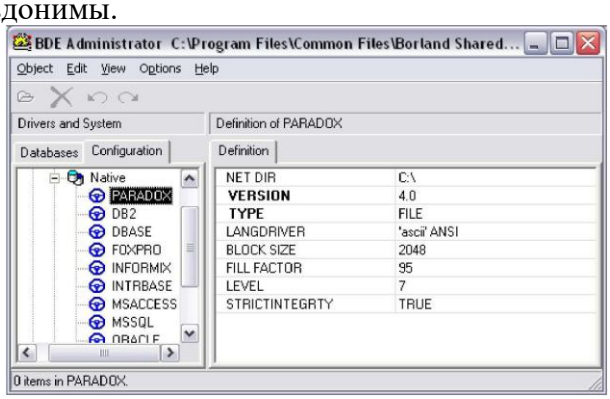


Рисунок 1 – Окно утилиты BDE Administrator

1.3. Псевдонимы

При работе с базами данных во многих случаях удобнее пользоваться псевдонимами, а не просто указывать путь доступа к таблицам базы данных. Псевдоним (alias - алиас) - это известное разработчику и BDE имя базы данных. В BDE с псевдонимом ассоциируются параметры, используемые для соединения с базой данных: формат БД, путь к её файлам, языковой драйвер, имя сервера, имя пользователя, режим открытия и т.п.

Псевдоним сохраняется в отдельном конфигурационном файле на диске и позволяет исключить из программы прямое указание пути доступа к базе данных. Такой подход даёт возможность располагать данные в любом месте, не перекомпилирую при этом программу.

Для создания псевдонима в утилитах BDE Administrator и SQL Explorer необходимо выполнить следующие действия:

- на левой панели выбрать страницу Database;
- через всплывающее меню или меню Object выбрать команду New;
- в окне New Database Alias (рисунок 2) выбрать драйвер для работы с БД и нажать OK. При работе с БД Paradox выбрать Standard;

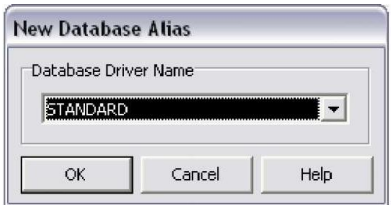


Рисунок 2 – Окно для выбора драйвера.

- на левой панели записать имя;
- на странице Definitions (правая панель) в поле Path (рисунок 3) указать путь к файлам БД: щёлкнуть на строке Path и с помощью кнопки обзора найти нужную папку;
- через всплывающее меню для левой панели или меню Object выбрать команду Apply.

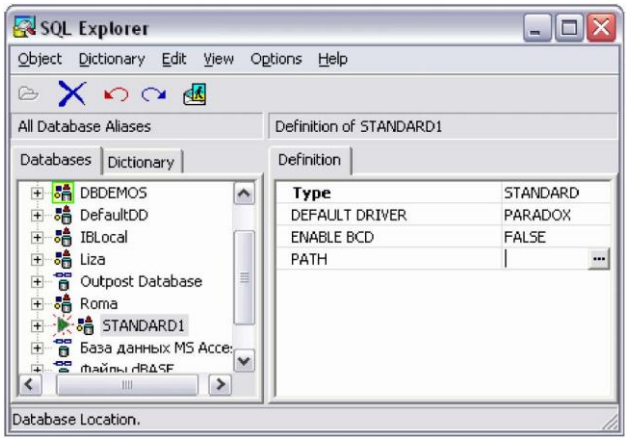


Рисунок 3 – Задание расположения БД.

Дополнительная информация, сообщаемая при создании псевдонима, зависит от типа выбранной базы данных. После создания нового псевдонима его имя вносится в общий список псевдонимов.

1.4. Обзор компонентов для работы с базами данных

Компоненты, используемые для работы с БД, находятся на страницах:

Data Access – невизуальные компоненты, предназначенные для организации доступа к данным;

Data Controls – визуальные компоненты для отображения данных;

dbExpres – компоненты для создания приложений, использующих технологию dbExpress;

BDE – компоненты для создания приложений, использующих BDE;

ADO (Delphi 7) или ***dbGo*** (Delphi 2005 и Delphi 2006) – компоненты для создания приложений по технологии ADO;

InterBase – компоненты для работы с сервером InterBase.

Таким образом, в Delphi предусмотрены специальные наборы компонентов, обеспечивающие доступ к данным при использовании разных технологий, и наборы компонентов, отображающие данные. Компоненты доступа к данным позволяют осуществлять соединения с БД, производить выборку, копирование данных и т.п.

Компоненты для отображения данных позволяют выводить данные в виде таблиц, полей, списков. Отображаемые данные могут быть текстового, графического или произвольного форматов.

Компоненты для работы с базами данных можно разделить на три группы: множества данных (data sets); визуальные компоненты баз данных (data- aware controls) и источники данных (data sources).

Множества данных - это невизуальные компоненты, которые взаимодействуют с BDE и обеспечивают доступ к данным в таблицах. Наиболее важные из них - компоненты Table и Query.

Визуальные компоненты баз данных - это управляющие элементы пользовательского интерфейса для просмотра и редактирования данных. Многие из них дублируют обычные управляющие компоненты: DBEdit, DBCheckBox, DBRadioGroup, DBImage и др.

Источники данных - это невизуальные компоненты, исполняющие роль трубопроводов между множествами данных и визуальными компонентами баз данных. Используя введённые понятия, можно уточнить структуру приложения, осуществляющего доступ к данным через BDE.

1.5. Модули данных

Модуль данных - это контейнер для невизуальных компонентов доступа к базе данных. Для создания модуля данных надо выполнить команду File|New|Other и в окне New Items выбрать Data Module.

Модуль данных является объектом класса `TDataModule`, в него можно помещать только невизуальные компоненты и задавать для компонентов доступа к данным обработчики событий. Для модуля данных определено всего несколько свойств `(Name, Tag)` и событий `(OnCreate, OnDestroy)`, так как в отличие от формы его непосредственным предком является класс `TComponent`. Использование модуля данных позволяет отделить логику обработки данных от логики работы пользовательского интерфейса.

Для форм и модулей данных, создаваемых в приложении, Delphi использует сквозную нумерацию. Для подсоединения модуля данных используется команда File|Use Unit.

При разработке приложений целесообразно поместить множества данных и источники данных в модуль данных, а визуальные компоненты - на формы.

1.6. Невизуальные компоненты для работы с данными

1.6.1. Компонент Table

Компонент Table обеспечивает доступ к таблицам базы данных, создавая набор данных, структура полей которого повторяет таблицу БД. Набором данных называют записи одной или нескольких таблиц, переданные в приложение в результате активизации компонента доступа к данным.

С помощью компонента Table можно организовать доступ к любой записи таблицы или их подмножеству. Компонент Table содержит все необходимые свойства, события и методы для создания, удаления, модификации, сортировки, фильтрации и поиска записей в таблице.

1.6.2. Компонент DataSource

Компонент DataSource[1] обеспечивает взаимодействие набора данных с компонентами для отображения данных. С каждым компонентом доступа к данным должен быть связан как минимум один компонент DataSource. С одним компонентом DataSource может быть связано несколько визуальных компонентов.

1.7. Визуальные компоненты для работы с данными

Большинство визуальных компонентов для работы с данными похожи на соответствующие компоненты для обычных приложений. Основное отличие заключается в том, что компоненты для работы с базами данных содержат свойства, позволяющие указать источник данных `(DataSource)` и поле, из которого компонент получает данные `(DataField)`.

Названия компонентов начинаются с букв DB. Большинство компонентов предназначено для отображения текущего значения одного поля: DBEdit, DBCheckBox, DBRadioGroup, DBText и др. Для вывода и редактирования данных поля Memo используются компоненты DBMemo и DBRichEdit. Для просмотра изображений предназначен компонент DBImage. Отображать данные графически позволяет компонент DBChart.

Кроме того, имеются компоненты, предназначенные для использования только в приложениях баз данных. Это DBNavigator и компоненты DBLook- upComboBox, DBLookupListBox для синхронного просмотра данных из связанных таблиц.

1.7.1. Компонент DBNavigator

Компонент DBNavigator является удобным средством перемещения по записям таблицы. Представляет собой панель с кнопками, построенную по типу панелей управления электронными устройствами. DBNavigator обеспечивает:

- прокрутку записей поодиночке вперёд и назад;
- переход к первой или последней записи;
- вставку новой записи;
- удаление текущей записи;
- переход в режим редактирования текущей записи;
- внесение изменений в таблицу;
- отмену сделанных в текущей записи изменений;
- обновление отображаемых значений. Это необходимо в тех случаях, когда данные в таблице БД могут изменяться другими приложениями.

Не все из перечисленных возможностей навигатора бывают нужны. Можно оставить только те кнопки, которые реализуют требуемую функциональность. Для управления видимостью кнопок используется свойство `visible- Buttons:` для каждой кнопки отдельно выбирается `true` или `false.`

Свойство `flat` управляет внешним видом кнопок: предусмотрены объёмное `(false)` и плоское `(true)` изображения. Свойства `Hints` и `ShowHint` предназначены для вывода подсказок. По умолчанию подсказки к кнопкам выводятся на английском языке и содержат текст, приведённый в свойстве `Hints`. Если требуется изменить подсказку к отдельной кнопке или создать подсказки на русском языке, то надо открыть строковый редактор свойства `Hints` и записать текст подсказок.

1.7.2. Компонент DBGrid

---

[1] *Полный перечень свойств и методов компонентов можно посмотреть в справочной системе.*

Визуальный компонент DBGrid предназначен для организации табличного просмотра и редактирования данных. Внешний вид данных, отображаемый DBGrid, по умолчанию соответствует структуре набора данных. Компонент DBGrid часто называют сеткой.

Для перемещения по записям используются полосы прокрутки и клавиши управления курсором. Для изменения данных достаточно установить курсор в нужную ячейку и ввести другое значение. Новая пустая строка создаётся в позиции указателя нажатием на клавишу Insert. Чтобы изменения, сделанные при редактировании и добавлении записи, были внесены в таблицу, необходимо нажать на клавишу Enter или перейти на другую строку. До того как данные были переданы в таблицу, можно клавишей Esc отменить изменения. Для удаления записи используется комбинация клавиш Ctrl+Delete.

Свойство `Options` является составным. Его значения определяют особенности поведения и внешнего представления компонента на экране. По умолчанию свойство `Options` содержит комбинацию значений `[dgEditing, DgTitles, dglndicator, DgColumn-Resize, DgColLines, DgRowLines, DgTabs, DgConfirmDelete, DgCan-celOnExit],`

По умолчанию для каждого поля исходной таблицы в DBGrid создаётся отдельный столбец. Такие столбцы называются динамическими. Характеристики динамического столбца определяются свойствами поля, для отображения которого этот столбец используется. Например, название столбец получает по названию поля, а ширина столбца определяется типом данных поля. Это не всегда удобно. При необходимости в DBGrid можно перейти к статическим столбцам, параметры которых задаются независимо от свойств поля. Для формирования статических столбцов используется Редактор колонок.

Компонент DBGrid обычно используют для просмотра данных. Для ввода и изменения данных используют компоненты, работающие с одним полем, так, чтобы на экране были видны поля только одной записи.

ПРИЛОЖЕНИЕ А

Таблица Jurnal

	Field Name	**Type**	**Size**	**Key**
1	Kod_jurnal	Numer-ic		*
2	Name	Alfa	20	
3	Number	Short		
4	Year	Short		

Таблица Soderjanie

	Field Name	**Type**	**Size**	**Key**
1	Auto	+		*
2	Kod_jurnal	Numer-ic		
3	Name_st	Alfa	200	
4	Opisanie	Memo	4	
5	Str	Alfa	3	

ПРАКТИЧЕСКАЯ РАБОТА №12

Изучение компонентов ado. связь с access через ado

1. Цель работы

1.1. Изучить возможности технологии доступа к данным ADO.

1.2. Научиться создавать связь приложения базы данных с базой данных Access, используя ADO.

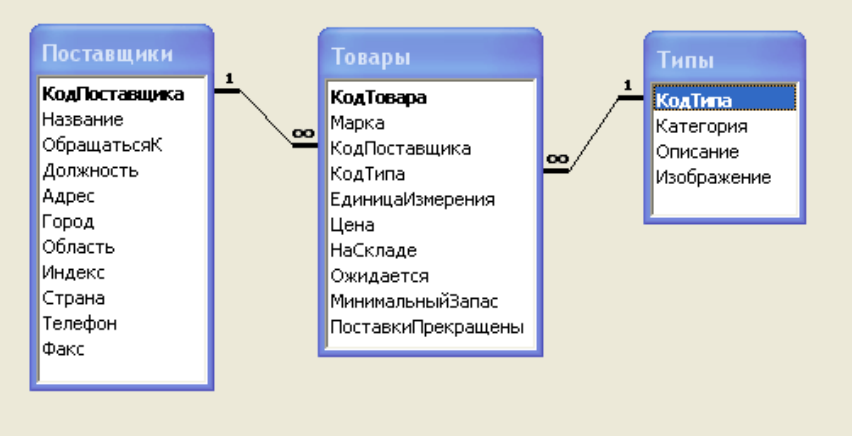
2. Приборы и оборудование

2.1. Методические указания.

2.2. Программное обеспечение Delphi.

3. Порядок выполнения работы

3.1. Создайте базу данных в Access в соответствии с рисунком:



3.2. В новом проекте Delphi создайте отдельный модуль данных. В нем расположите компонент TADOConnection и настроить соединение с базой данных:

- установите соединение с провайдером, для чего для свойства ConnectionString откройте построитель (три точки справа от поля свойства);

- в появившемся окне необходимо нажать на кнопку Build. На вкладке Provider выбрать Microsoft Jet 4.0. OLE DB Provider. На вкладке Connection необходимо указать путь к базе данных. Нажмите на кнопку Test connection. Закройте окно в случае положительного тестирования;

- свойству LoginPrompt присвойте значение true и активируйте соединение (для свойства Connection установить значение true).

3.3. Расположите 3 компонента TADOTable и свяжите их с компонентом TADOConnection при помощи свойства Connection. В свойстве TableName следует выбрать имя таблицы, а затем активизировать компонент. А так же расположите компонент TDataSource и свяжите его с компонентом TADOConnection.

3.4. Расположите на главной форме компонент TDGrid и свяжите его с компонентом TDataSource. А так же TDBNavigator и свяжите его TDataSource.

3.5. Создайте формы для каждой таблицы. И заполните таблицы записями.

3.6. Создайте главную форму для запуска всех форм.

3.7. Оформите отчет, сделайте выводы о проделанной работе.

4. Содержание отчёта:

4.1. работы.

4.2. Цель работы.

4.3. Приборы и оборудование.

4.4. Выполнение работы.

4.5. Выводы.

5. Контрольные вопросы:

5.1. Как организовать доступ к данным по технологии OLE DB, если для нужной СУБД нет OLE DB-драйвера, но есть ODBC-драйвер?

5.2. Приведите схему доступа к данным с применением ADO.

5.3. Почему активно используется доступ к данным с использованием ADO?

5.4. Какие объекты используются в технологии ADO?

5.5. Какие компоненты Delphi используются для организации доступа к данным по технологии ADO?

5.6. Как задаются параметры соединения при разработке в Delphi приложения, использующего технологию ADO?

ПРИЛОЖЕНИЕ А

Теоретические сведения

ADO

Компанией Microsoft был предложен механизм доступа к данным ActiveX Data Objects (ADO), построенный на использовании интерфейсов OLE DB. ADO это набор библиотек, содержащих COM-объекты, реализующие прикладной программный интерфейс для доступа к данным и используемые в клиентских приложениях. Технология ADO использует библиотеки OLE DB, предоставляющие низкоуровневый интерфейс для доступа к данным.

ADO становится всё более популярным способом доступа к данным, так как включен в ядро операционных систем семейства Windows, и входит в состав таких популярных продуктов, как MS Office.

Согласно терминологии ADO любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым приложение взаимодействует при посредстве провайдера.

Приложение обращается к данным не напрямую, а через объект OLE DB, который «умеет» работать с разнообразными данными. Технология ADO включает в себя набор объектов и механизмов, обеспечивающих взаимодействие объектов с данными и приложениями. Очень важную роль играют провайдеры ADO, координирующие работу приложений с хранилищами данных различных типов.

Набор объектов и соответствующий провайдер могут быть созданы для любого хранилища данных без внесения изменений в структуру ADO.

Для каждого используемого типа хранилища данных должен существовать провайдер ADO. Провайдер знает, какие данные и где расположены, умеет обращаться к данным с запросами и интерпретировать возвращаемую служебную информацию и результаты запросов для передачи приложениям. При установке соединения через соответствующие компоненты становится доступен список установленных в операционной системе провайдеров.

В технологии ADO используются объекты-перечислители, источники данных, сессии, транзакции, наборы рядов, команды.

Объекты-перечислители выполняют поиск объектов ADO, осуществляющих доступ к источнику данных.

Для соединения с хранилищем данных используются два типа объектов: источники данных и сессии.

Объект-набор рядов обеспечивает работу с данными.

Объект-команда объединяет текстовую команду и механизмы обработки команд. Команды позволяют использовать для работы с данными язык SQL.

МЕХАНИЗМЫ ДОСТУПА К ДАННЫМ, ПОДДЕРЖИВАЕМЫЕ DELPHI

Имеющиеся в Delphi классы и компоненты позволяют быстро и эффективно разрабатывать приложения баз данных. Для удобства использования компоненты разбиты на группы:

1. Компоненты для доступа к данным, реализующие:

— доступ через процессор баз данных BDE, используя ODBC- драйверы или внутренние драйверы BDE;

— доступ через ADO-объекты, в основе которого лежит применение технологии OLE DB;

— доступ к локальному или удалённому SQL-серверу InterBase;

— доступ посредством драйверов dbExpress;

— доступ к БД при многозвенной архитектуре (компоненты страницы DataSnap);

2. Компоненты для связи источников данных с визуальными компонентами, предоставляющими интерфейс пользователя;
3. Визуальные компоненты, реализующие интерфейс пользователя;
4. Компоненты для визуального проектирования отчётов.

Разные механизмы работы с данными имеют схожие схемы. Ранее подробно была рассмотрена схема работы через BDE с использованием внутренних драйверов (с таблицами форматов Paradox, dBase, FoxPro). Работа через BDE с использованием ODBC-драйверов организуется аналогично.

В модуль данных (или в форму) добавляется компонент набора данных (объект класса `TDataSet`) и устанавливается связь с источником данных, определяемая свойством `DataDaseName`. Связь может быть указана по имени БД, каталогу или псевдониму (ограничения зависят от типа источника данных).

В модуль данных (или в форму) добавляется компонент источника данных (`TDataSourse`), являющийся связующим звеном между набором данных и элементами управления, отображающими данные. Свойство `DataSet` компонента типа `TDataSourse` указывает набор данных, формируемый компонентами таких классов, как `TTable` или `TQuery`.

В форму добавляются элементы управления для работы с данными, такие как `TDBGrid`, `TDBEdit`, `TDBCheckbox` и т.п. Они связываются с источником данных через свойство `DataSource`.

При реализации схем доступа к данным через ADO, dbExpress применяются другие компоненты, но подход остаётся тем же.

Предком всех классов наборов данных является класс `TDataSet`. В зависимости от механизма доступа, используемого приложением, базовыми классами набора данных могут быть:

`TTable`, `TQuery`, `TStoredProc` - для однозвенных или двухзвенных приложений, использующих BDE;

`TClientDataSet` - для реализации клиентского набора данных и для многозвенной архитектуры, использующей распределенный доступ;

`TADODataSet` - для приложений, использующих ADO-объекты;

`TSQLDataSet` - для доступа к базе данных посредством dbExpress. Этот класс реализует направленный набор данных. Для такого набора данных не создайся кэш памяти на клиенте, и среди методов доступа возможны только методы `Next` и `First`. Редактирование записей в направленном наборе данных возможно только явным выполнением SQL-оператора `UPDATE` или при установке соединения с клиентским набором данных через провайдера;

`TSQLTable` и `TSQLQuery` - для доступа к базе данных посредством dbExpress.

На рисунке 2 приведена иерархия классов наборов данных библиотеки VCL системы Delphi. При работе с компонентами наборов данных можно обойтись без явного использования компонентов, реализующих соединение с базой данных. Однако некоторые возможности, такие как управление транзакциями или кэшированные обновления, невозможны без компонентов типа `TDatabase` или `TADOConnection`.

Компонент база данных `TDatabase` применяется для соединения с источником данных через драйверы BDE или внешние ODBC-драйверы. Компонент `ADOConnection` используется для создания объекта-соединения при доступе через OLE DB, который реализуется посредством ADO-объектов VCL- библиотеки.

По умолчанию при переходе от одной записи набора данных к другой происходит запись всех сделанных изменений в базу данных. Для того чтобы можно было отменять сделанные изменения или выполнять обновление нескольких записей, применяют кэшированные об-

новления. Они позволяют значительно снизить сетевой трафик за счет того что все сделанные изменения хранятся во внутреннем кэше. И при переходе от одной записи к другой информация в базу данных не передаётся. Чтобы включить режим кэшированного обновления, следует установить значение свойства `CachedUpdates` равным `True` для компонента набора данных. Для присвоения кэшированного обновления вызывается метод `ApplyUpdates`, а для отмены - `CancelUpdates`.

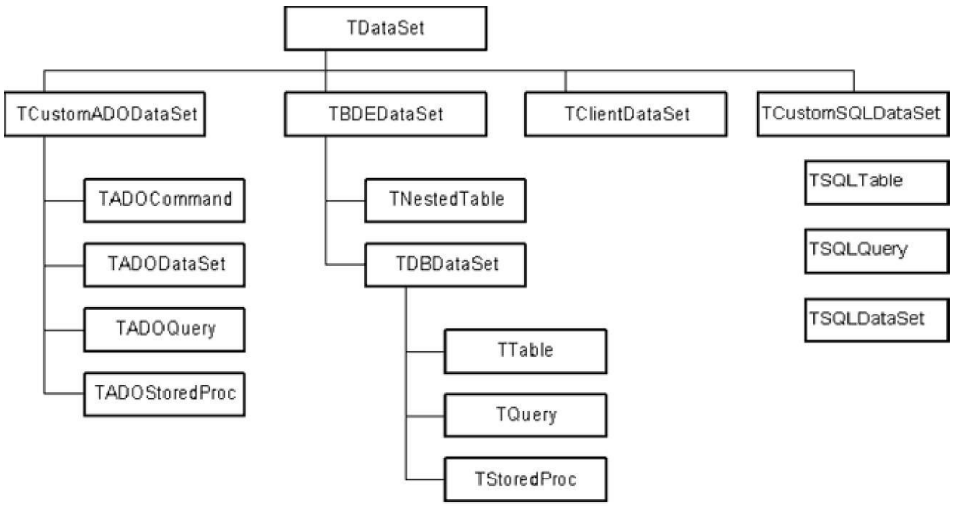


Рисунок 2 – Иерархия классов, реализующих доступ к данным

ТЕХНОЛОГИЯ ADO

Для применения технологии ADO в Delphi 7 предназначены семь компонентов, расположенных на закладке ADO палитры компонентов.

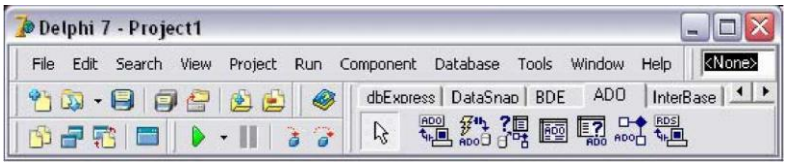


Рисунок 3 – Компоненты страницы ADO

Таблица 1 Назначение компонентов ADO

Название	Описание
ADOConnection	Функционально аналогичен компоненту **Database** закладки BDE. Позволяет указывать местоположение базы данных и работать с транзакциями
ADOCommand	Предназначен для выполнения SQL–команды без возврата результирующего набора данных
ADODataset	Предназначен для получения набора данных из одной или нескольких таблиц БД. Позволяет работать с возвращённым набором данных визуальным компонентам
ADOTable	Аналог компонента **Table**, расположенного на закладке BDE. Используется для доступа к таблице с помощью механизма ADO
ADOQuery	Аналог **Query**. Позволяет формировать запросы к БД, которые возвращают данные из базы (например, командой `SELECT`) или не формируют результирующего набора данных (например, `INSERT`)
ADOStoredProc	Предназначен для вызова процедуры, хранимой на сервере базы данных. Является потомком `TDataSet`, в отличие от BDE и InterBase позволяет возвращать набор данных, поэтому может выступать источником данных в компонентах типа **DataSource**
RDSConnection	Управляет механизмом, который позволяет клиенту получать доступ к объектам, расположенным в другом адресном пространстве или на другом компьютере

Класс `TADOConnection` обеспечивает соединение с данными, доступ к которым реализуется через ADO-объекты. Компоненты ADOConnection используют для доступа к данным OLE DB-провайдеры. Компоненты ADOCommand и ADODataSet связываются с источником данных посредством объекта ADOConnection, указывая ссылку на него как значение свойства `Connection`.

Для идентификации соединения необходимо определить значение свойства `ConnectionString` (строка соединения) компонента ADOConnection, которое может основываться на указании datalink-файла или строки соединения. Если в качестве значения свойства `ConnectionString` указано имя datalink-файла, то настройку соединения можно выполнять автономно от приложения (например, указывая имя базы данных Microsoft SQL Server на текущем ПК).

Реализацию доступа к данным через ADO проще всего рассмотреть на примерах.

Пример. Разработать в `Delphi` приложение для просмотра объектов базы данных MS Access, используя механизм ADO. Последовательность действий

1. Откроем Delphi, создадим приложение и разместим на форме три компонента:
   - ADOTable с закладки ADO;
   - DataSource с закладки Data Access;
   - BDGrid с закладки Data Controls.
2. Свяжем компоненты между собой:
   - установим значение свойства `DataSource` компонента DBGridl в DataSourcel;
   - в свойстве `DataSet` компонента DataSourcel укажем ADOTablel.
3. Зададим параметры соединения компонента ADOTablel:
   - в свойстве `ConnectionString` (рисунок 4) нажмём кнопку с многоточием, в окне редактора параметров соединения установим переключатель в положение Use Connection String и нажмём кнопку Build;

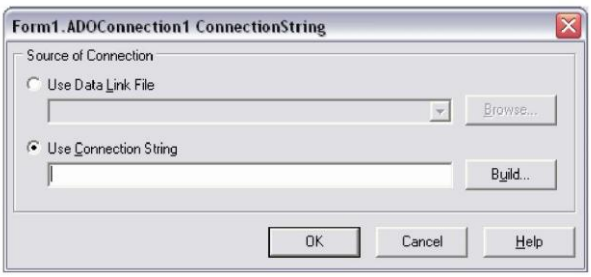


Рисунок 4 – Окно задания свойства `ConnectionString`

– в появившемся окне на вкладке Поставщик данных выберем свойство Microsoft Jet 4.0 OLE DB Provider (рисунок 5);

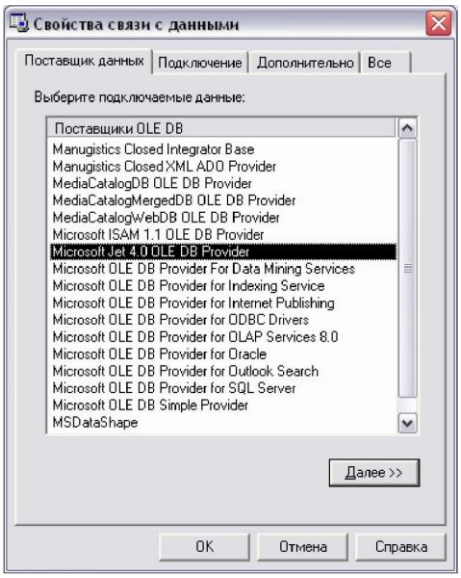


Рисунок 5 – Выбор OLE DB – провайдера.

— перейдём на вкладку Подключение, укажем путь к БД, введём имя пользователя и при необходимости зададим пароль (рисунок 6);

— нажмём кнопку Проверить подключение;

— после завершения проверки подключения перейдём на вкладку Дополнительно и поставим галочки напротив свойств ReadWrite, Share Deny None;

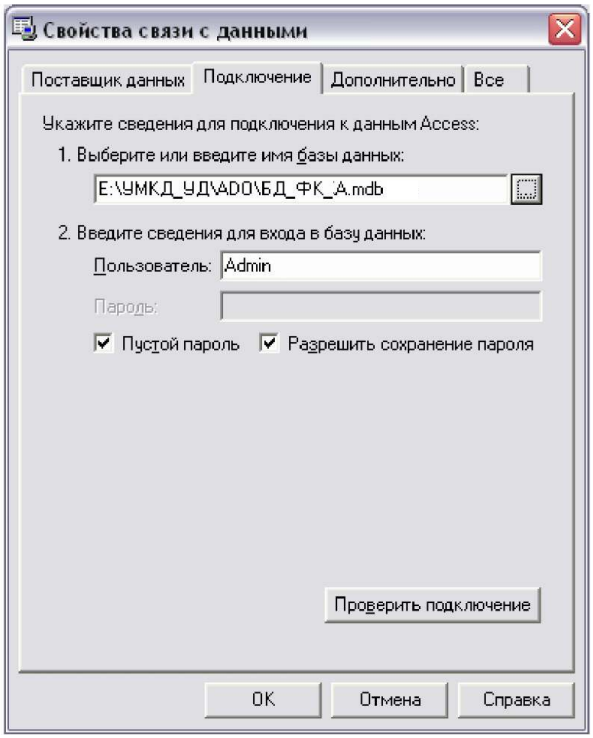


Рисунок 6 – Задание источника данных.

— выберем вкладку Все и проверим сделанные установки. При необходимости изменить значение выберем нужную строчку, нажмём кнопку Изменить значение, в появившемся окне выберем значение и нажмём ОК;

— нажмём два раза ОК;

— в свойстве `TableName` компонента ADOTablel укажем нужную таблицу;

— в свойстве `Active` установим значение `true`.

Если всё сделано правильно, то после задания таблицы компонент DBGridl заполнится данными.

ПРАКТИЧЕСКАЯ РАБОТА №13

Реализация com в delphi

1. Цель работы

1.1. Изучить возможности технологии доступа к данным COM.

1.2. Научиться создавать локальный сервер COM.

2. Приборы и оборудование

2.1. Методические указания.

2.2. Программное обеспечение Delphi.

3. Порядок выполнения работы

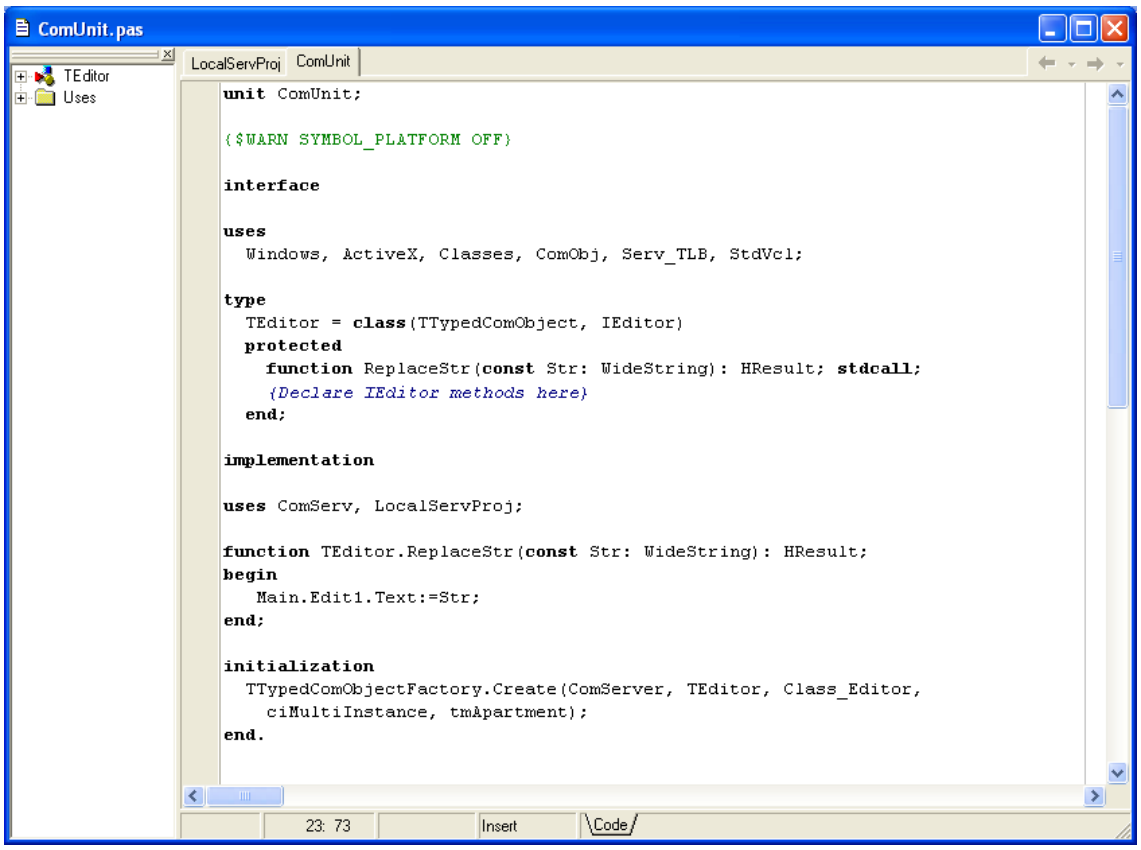
3.1. Создайте новый проект и сохраните его под именем Serv.

3.2. На главной форме расположите компонент TEdit. Сам модуль надо сохранить с именем LocalServProj.

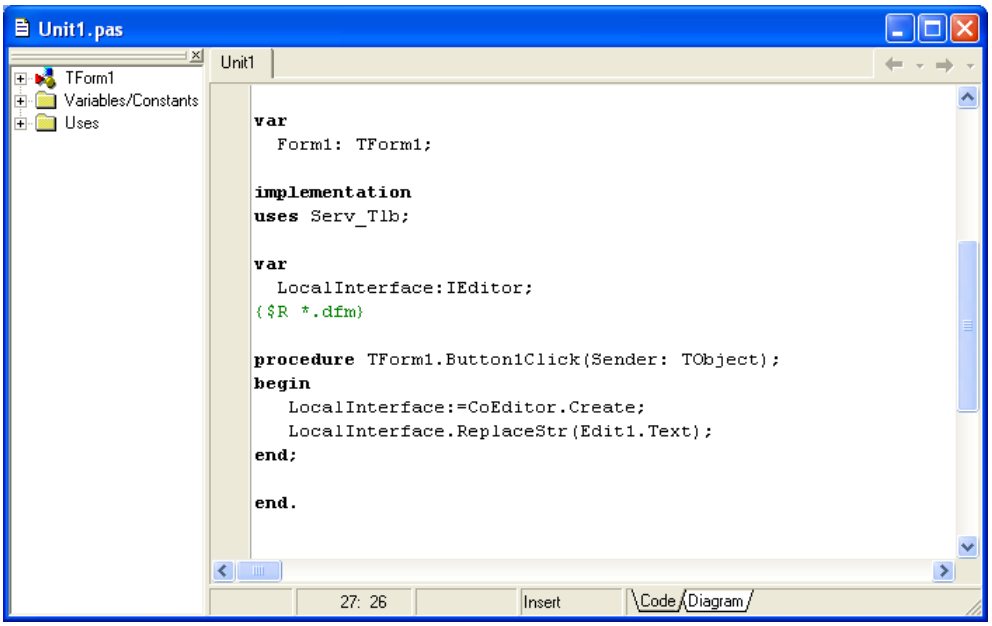
3.3. Добавьте объект COM, для чего выполните команду File->New…->Other и на вкладке ActiveX репозитория объектов следует выбрать значок COM Object. В поле ClassName окна мастера создания объекта нужно ввести значение Editor. В списке Instancing потребуется выбрать значение Multiple Instance, а в списке Threading Model – значение Apartment. Модуль созданного объекта сохранить с именем ComUnit.

3.4. Выполнить пункт меню View->Type Library. К интерфейсу IEditor добавьте метод ReplaceStr, для чего нажмите кнопку New Method на панели инструментов. На вкладке Parametrs нужно установить значение параметров метода. В поле Name установить значение Str, в поле Type выбрать значение BSTR. В поле Modifier установить значение in, а в списке Return Type выбрать значение HRESULT. После чего нажмите кнопку Refresh Implementation для формирования шаблона метода и его описания.

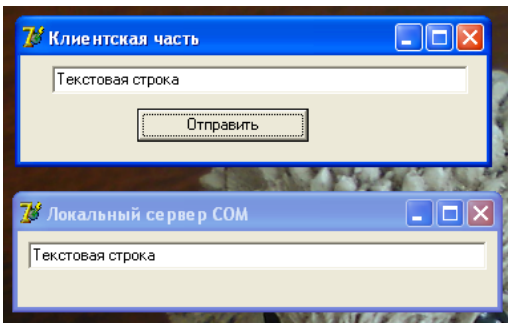
3.5. Откройте модуль ComUnit и модифицируйте его в соответствии с ниже приведенным кодом:



3.6. Разработайте клиентское приложение. Создайте новый проект. На главной форме расположите компонент TEdit и одну кнопку. Подключите библиотеку Serv_TLB. И создайте обработку события для кнопки. В результате должен быть получен следующий программный код:



3.7.В результате запуска программы должен получиться следующий результат:



3.8.Оформите отчет, сделайте выводы о проделанной работе.

4. Содержание отчёта:

4.1. работы.
4.2.Цель работы.
4.3.Приборы и оборудование.
4.4.Выполнение работы.
4.5.Выводы.

5. Контрольные вопросы:

5.1.Каким классом описывается объект COM?
5.2.Где описываются все свойства и методы объекта?
5.3.Какая приставка добавляется к имени класса объекта COM?
5.4.Как создается объект COM?
5.5.Как создаётся внутренний сервер COM? Его назначение.
5.6.Как создаётся локальный сервер COM? Его назначение.

ПРИЛОЖЕНИЕ А

Теоретические сведения

РЕАЛИЗАЦИЯ COM В DELPHI

Объект COM описывается обычным классом TCOMObject, который порожден непосредственно от класса TObject. Все свойства и методы объекта описываются в объявлении класса. При создании объекта COM с ним связывается вспомогательный класс CoClass, который описывает все его интерфейсы. При создании объекта к имени его класса добавляется приставка Co и создается CoClass этого объекта.

Описание CoClass содержится в библиотеке типов. Стандартное объявление класса обеспечивает создание кода объекта, который будет скомпилированв двоичный код. CoCI ass обеспечивает представление экземпляра класса в соответствии со спецификой COM и гарантирует корректное обращение клиента к объекту.

Класс TComObject

Класс TComObject является базовым классом, на основе которого создаются простые классы COM, такие как, например, расширения оболочек. Класс TComObject является COM-объектом, который поддерживает интерфейсы IUnknown и I Support Error Info. Класс реализует простые объекты, обладающие базовым списком возможностей:

— Объект обладает идентификатором класса CLSID, который используется для создания экземпляров класса с использованием фабрики класса,

— Объект поддерживает агрегирование при помощи методов интерфейса IUnknown.

— Объект поддерживает безопасные вызовы и обработку исключительных ситуаций OLE, используя интерфейс IProvideErrorlnfo.

— Объект использует в работе интерфейс lErrorlnfo.

Класс TComObject может использоваться как базовый класс для создания классов объектов COM, которые должны иметь идентификатор класса (CLSID). Идентификатор CLSID, как было сказано ранее, используется для регистрации класса в реестре и для создания его экземпляра извне при помощи вызова фабрики класса.

Объект COM, как н любой другой объект, создается конструктором Create. Конструктор создает объект как самостоятельный экземпляр класса, не входящий в агрегат. Под *агрегатом* следует понимать совокупность объектов COM, предоставляющих свои интерфейсы и имеющих один общий — IUnknown. Для создания объекта COM и включения его в агрегат используется метод CreateAggregated, который создает новый объект как часть агрегата. В параметре Controller указывается общий управляющий интерфейс IUnknown и передается свойству Controller.

Метод CreateFroraFactory используется для создания объекта и его инициализации. В свойстве Factory указывается фабрика класса объекта. После создания

объекта метод Initialize позволяет произвести его инициализацию. В ходе инициализации счетчик ссылок на объект увеличивается на единицу.

В свойстве RefCount содержится число ссылок на объект. Свойство RefCount определяет, когда созданный объект может быть уничтожен. Когда свойству присваивается пулевое значение, объект уничтожается, так как с ним не работает ни один клиент. Значение свойства увеличивается методом AddRef интерфейса IUnknown и реализуется методом ObjAddRef. Уменьшение значения свойства производится при помощи метода ObjRelease, реализуя метод Release интерфейса IUnknown.

Метод ObjQuerylnterface позволяет выяснить, имеет ли данный объект COM интерфейс с идентификатором, заданным IID. Данный метод является реализацией метода Querylnterface интерфейса lUnknown

Класс TTypedComObject

Класс TTypedComObject является прямым наследником класса TComObject. Он Предназначен для создания объектов COM с использованием библиотеки типов. Класс TTypedComObject имеет интерфейс IProvideClasslnfo, в состав которого входит единственный метол, предназначенный для получения указателя на CoClass объекта.

Метод GetClassInfo является реализацией метода GetClassInfo интерфейса IProvideClassInfo.

Интерфейс lUnknown

Интерфейс lUnknown является базовым для всех интерфейсов. Он является прямым потомком класса Interface, на основе которого в Delphi строятся все интерфейсы.

В состав интерфейса входят методы AddRef, Release и Duerylnterface, которые рассматривались ранее.

В коде Delphi интерфейс lUnknown подменяет собой класс Interface.

Класс TComObjectFactory

Класс TComObjectFactory является фабрикой класса для объектов COM. порожденных от класса TComObject. Класс TComObjectFactory обеспечивает функционирование интерфейсов lUnknown, IClassFactory и IClassFactory2. Интерфейс IClassFactory создает объект, принимая в качестве параметра его идентификатор CLSID. Интерфейс IC1 assFactory2 используется для обеспечения лицензирования объекта COM. Создание объекта фабрики класса может быть инициировано извне при помощи функции API CoCreateClassObject. Для создания объекта также могут быть использованы функции CreateComObject и Create- OleObject. Если создается несколько объектов одного класса, эффективнее вызывать фабрику класса, получая указатель на его интерфейс IClassFactory и используя его собственные методы. Для управления фабриками классов на сервере COM используется специальный класс TComClassManager, доступ к которому можно получить, используя функцию ComClassManager. Метод CreateComObject инициирует вызов конструктора класса TComObjectFactory. Конструктор Create создает фабрику класса во время запуска сервера. Конструктор фабрики класса описывается в сек-

ции инициализации модуля, включающего сервер COM. В параметре ComServer указывается сервер COM, в составе которого будет функционировать объект. В параметре ComClass указывается тип класса, который используется методом GetFactoryFromClass менеджера классов для идентификации фабрики. Параметр ClasslD задает идентификатор класса, создаваемого этой фабрикой, а параметр Instancing задает способ создания объекта. Этот способ регламентируется одним из перечисленных ниже значений:

— Значение ciInternal указывает, что объект создается в процессе как сервер COM. Другие приложения не могут создать экземпляр объекта напрямую, но могут использовать методы приложения для создания экземпляра документа.

— Значение ciSinglelnstance указывает, что для каждого клиента, обратившегося к серверу COM, создается собственный экземпляр объекта. Все объекты создаются в единственном экземпляре сервера.

— Значение ciHultiInstance указывает, что для каждого клиента создается собственный экземпляр сервера, в котором содержится экземпляр объекта.

В параметре ThreadingModel задается способ взаимодействия сервера и клиента. Значения параметра перечислены ниже:

— Значение traSingle указывает, что сервер последовательно выполняет запросы клиентов.

— Значение tmApartment свидетельствует, что вызов объекта осуществляется только в том потоке, в котором он создан. Различные объекты некоторого сервера могут быть вызваны в различных потоках, при этом один объект может обслуживать одновременно только одного клиента.

— Значение tmFree указывает, что объект может одновременно использоваться произвольным числом клиентов.

— Значение traBoth указывает, что клиент может использовать модели tmApartment или tmFree.

— Значение tmNeutral указывает, что к данному объекту одновременно могут обращаться несколько клиентов в различных потоках. Но защиту от возможных конфликтов обязан обеспечить программист, используя критические секции, моникеры и иные соответствующие средства. Данная модель доступна только для технологии COM+. При использовании COM применяется модель tmApartment.

Метод RegisterClassObject регистрирует класс создаваемого объекта. Приложения, содержащие сервер COM, вызывают этот метод при запуске соответствующего сервиса. Метод UpdateRegistry вызывается для регистрирования объекта COM или удаления его регистрации. Регистрация объекта производится при первом запуске приложения. В свойстве ClasslD указывается идентификатор класса, а свойство ClassName определяет имя класса объекта, В свойстве ErrorHO содержится GUID интерфейса, в котором произошла ошибка. Свойство ShowErrors включает или отключает показ сообщений об ошибках при создании объекта. Если свойство имеет значение True, то сообщение о возникающих ошибках выводится в информационном окне.

Класс TTypedComObjectFactory

Класс TTypedComObjectFactory является прямым наследником класса TComObjectFactory. Он используется для создания объектов класса TypedComObject.

Свойство Classlnfo позволяет получить информацию о типе без необходимости применения загрузки библиотеки типов. Для получения информации об объекте используется интерфейс ITypelnfo.

Метод GetInterfaceTypeInfo возвращает информацию об объекте COM, созданном данной фабрикой.

Класс TComClassManager

Класс TComClassManager используется для управления фабриками классов. Экземпляр класса TComClassManager возвращается функцией ComClassManager, которая содержится в модуле ComObj.pas. Этот экземпляр управляет фабриками классов объектов, владельцем которых является данный сервер COM. Экземпляр класса получает от сервера COM список фабрик н обновляет его при создании нового объекта или уничтожении старого. Разработчику предоставляются методы, позволяющие управлять фабриками классов. Функция ComClassManager возвращает ссылку на экземпляр класса, А метод ForEachFactory последовательно выполняет определенные действия над всеми фабриками классов данного сервера COM. В параметре ComServer указывается сервер COM, а в параметре FactoryProc — ссылка на метод, выполняющий необходимые действия.

Метод GetFactoryFromClass возвращает фабрику класса для класса, указанного в параметре ComClass. А метод GetFactoryFromClassID возвращает фабрику класса, принимая в качестве входного параметра идентификатор класса ClassID.

Класс TComServer

Класс TComServer предназначен для создания экземпляров серверов COM. При создании объекта COM него модуль автоматически добавляется модуль ComServ, содержащий реализацию класса TComServer. Класс TComServer содержит всю необходимую информацию о сервере. В нем упоминаются библиотека типов, имя, файл справки и многие другие параметры. Также сервер содержит информацию о том, как он должен быть загружен и как он должен быть выгружен из адресного пространства памяти. Класс сервера применяется для создания экземпляров фабрик классов объектов, используя их идентификаторы CLSID.

Создается объект сервера COM методом Create. Метод Initialize вызывается при создании сервера. При первом запуске он производит регистрацию всех связанных объектов COM в системном реестре.

Режим запуска сервера определяется в свойстве StartMode. Его возможные значения перечислены далее:

— Значение smAutomation указывает, что сервер запускается как сервер автоматизации в ответ на запрос от контроллера автоматизации.

— Значение smRegServer указывает, что сервер запускается с целью регистрации в системном реестре (первый запуск) либо с использованием ключа /regserver.

— Значение smStandalone указывает, что сервер запускается пользователем как отдельное приложение.

— Значение smUnregServer указывает, что сервер запускается с целью удаления регистрации из системного реестра (используется ключ /unregserver).

Свойство IsInprocServer указывает, является ли данный сервер COM внутренним сервером или нет. Для внутреннего сервера используется значение True. В свойстве ServerKey указывается, является ли данный сервер COM внутренним или локальным. Свойство содержит строку InprocServer32, если сервер является внутренним и оформлен в виде библиотеки DLL. Если сервер является локальным и существует в форме EXE-файла, то используется значение LocalServer32.

Для загрузки библиотеки типов следует вызвать метод LoadTypeLib. В случае возникновения ошибки будет возвращен объект класса исключения E01e-SysError С информацией об ошибке.

Получить доступ к библиотеке типов, связанной сданным сервером, можно через свойство Typelib, возвращающее интерфейс ITypeLib. Выяснить количество объектов, запущенных в данном сервере COM, можно при помощи свойства ObjectCount. Значение свойства увеличивается на единицу, когда новый объект создается фабрикой класса, и уменьшается, когда он уничтожается. В свойстве ServerName содержится имя сервера. Задать его можно методом SetServerName. Если сервер использует библиотеку типов, то метод не будет выполнен, так как свойство получит значение автоматически из библиотеки типов.

В свойстве ServerFileName содержится полный путь к файлу сервера и его имя. А путь к соответствующему файлу справки содержится в свойстве BelpFileName.

Создание внутреннего сервера COM и работа с ним

Для создания серверов COM Delphi предоставляет широкий выбор мастеров, автоматизирующих выполнение трудоемких задач. Мастера доступны на вкладке ActiveX репозитория объектов. Перед созданием внутреннего сервера COM в виде DLL необходимо создать специальную библиотеку, которая имеет несколько функции и описывается по правилам COM.

Репозитории объектов можно активировать при помощи команды меню File ► New ► Other. Для дальнейшей работы потребуется вкладка ActiveX. Нужно создать новый проект, выбрав значок ActiveX Library.

Созданный проект нужно сохранить с именем TestServ. В листинге приведен код созданной библиотеки.

Листинг. Код библиотеки TestServ

```
library TestServ:
uses
    CofflServ;
exports
```

DllGetClassObject.
DllCanUnloadNow.
DllRegisterServer.
DllUnregisterServer:
{$R *.RES}
begin
end.

Созданная библиотека экспортирует функции DIlGetClassObject, DllCanUnloadNow, DllRegisterServer и DllUnregisterServer, необходимые для работы с COM-технологией. Эти функции объявляются в модуле ComServ:

— Метод DllGetClassObject возвращает фабрику класса для объекта COM.

— Метод DllCanUnloadMow производит проверку на предмет того, возможно ли выгрузить ИЗ памяти DLL-библиотеку данного сервера COM. Функция возвращает значение S 0K, если выгрузка возможна. В противном случае возвращается значение S FALSE.

— Метод DllRegisterServer регистрирует сервер COM в системном реестре.

— Метод DllUnregisterServer удаляет регистрацию данного сервера COM из системного реестра.

После этого необходимо создать объект COM. Для этого в репозитории объектов на вкладке ActiveX следует выбрать значок COM Object. В результате появится диалоговое окно, в котором необходимо заполнить несколько ключевых полей. Объект будет создаваться с учетом данных, введенных разработчиком.

В поле ClassName указывается имя создаваемого класса. Под этим именем он будет зарегистрирован в реестре, и оно же будет использовано для названия класса CoClass. На рисунке видно, что было выбрано имя TestCom. В списке Instancing определяют способ создания объекта. Следует установить значение Multiple Instance. В списке Threading Model указывается способ взаимодействия сервера и клиента. В данном случае требуется использовать значение Apartament. В поле Implemented Interface указываются интерфейсы, входящие в состав создаваемого объекта COM. По умолчанию для объекта создается собственный интерфейс. Также интерфейс можно выбрать из списка, получаемого при нажатии кнопки List. Активируемое диалоговое окно показано на рисунке 3. Установка значка Include Type Library приводит к включению в сервер библиотеки типов. Если флажок установлен, то объект наследуется от класса TTypedComObject. если снят — от класса TCOMObject. Установка флажка Mark interface Oleautomation указывает, что сервер COM создается как совместимый с технологией OLE.

Созданный модуль нужно сохранить с именем ComUnit. После того как к серверу будет добавлен объект, в секции Uses библиотеки появится указатель на него и на библиотеку типов сервера, как показано в листинге 1.

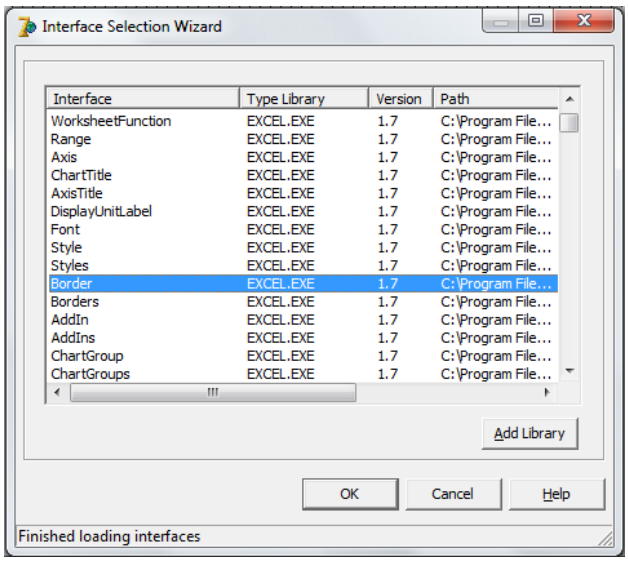


Рисунок 3 – Список интерфейсов, зарегистрированных на компьютере.

Создание локального сервера COM и работа с ним

Для разработки локального сервера потребуется создать новый проект и сохранить его иод именем Serv. На главной форме нужно расположить компонент TEdit, Сам модуль надо сохранить с именем LocalServProj. Теперь к проекту нужно добавить объект COM. Для этого на вкладке ActiveX репозптормя объектов следует выбрать значок COM Object и добавить соответствующий объект в проект. В поле ClassName окна мастера создания объекта COM нужно ввести значение Editor. В списке Instancing потребуется выбрать значение Multiple Instance, а в списке Threading Model - значение Apartament. Модуль объекта COM нужно сохранить с именем ComUnit.

Выполнив пункт меню View ► Type Library, нужно вызвать библиотеку типов объекта. К интерфейсу IEditor следует добавить метод ReplaceStr. На вкладке Parametrs нужно установить значения параметров метода. В поле Name должно оказаться значение Str, в поле Type нужно выбрать значение BSTR. В поле Modifier нужно установить значение In, а в списке Return Type выбрать значение HRESULT. После этого нужно нажать кнопку Refresh Implementation для формирования шаблона метода и его описания.

Данный COM-объект будет взаимодействовать с формой приложения. С помощью метода ReplaceStr будет изменяться содержимое редактора TEdit. Код объекта приведен в листинге 4.8. Для того чтобы зарегистрировать сервер в системе, достаточно просто запустить его.

Листинг. Код локального сервера COM

```
unit ComUnit;
{$WARN SYMBOLJLATFORM OFF}
interface
uses
Windows, ActiveX, Classes, ComObj, Serv_TLB. StdVcl;
type
```

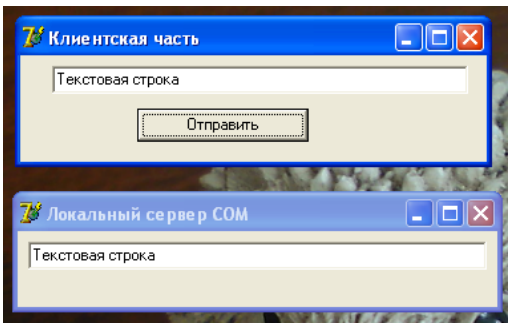
```
TEditor = class(TTypedComObject, IEditor)
protected
function ReplaceStr (const Str: WideString): HResult; stdcall;
{Declare IEditor methods here}
end;
implementation
uses ComServ, LocalServProj;
function TEditor.ReplaceStrt(const Str: WideString): HResult;
begin
Main.Editl.Text:=Str;
end;
initialization
TTypedComObjectFactory.Create(ComServer, TEditor, Class_Editor, ciMulti-
Instance. tmApartment);
end.
```

Теперь нужно разработать клиентскую часть приложения. В новом проекте на главной форме нужно расположить компонент TEdit и одну кнопку. Чтобы иметь возможность обращаться к серверу, необходимо подключить библиотеку типов. Для этого в секции uses надо подключить библиотеку Serv_TLB. Если сервер располагается в ином каталоге, нежели клиентская часть, то необходимо скопировать библиотеку типа в каталог с клиентской частью. Код клиентского приложения приведен в листинге.

Листинг. Код клиентской части

```
var
Forml; TForml;
implementation
uses Serv_TLB;
var
LocalInterface : IEditor;
{$SR *.dfm}
procedure TForml.SendBtnClick(Sender; TObject);
begin
Local Interface: =CoEditor.Create;
LocalInterface.ReplaceStr(SendEdit.Text);
end;
end.
```

В методе SendBtn создается CoClass объекта и в переменную LocalInterface передается указатель на интерфейс IEditor — базовый интерфейс класса. После этого вызывается метод, в качестве параметра которому передается строка. На рисунке показана основная форма приложения и окно сервера.


Клиентская часть
Текстовая строка
Отправить
Локальный сервер COM
Текстовая строка

ПРАКТИЧЕСКАЯ РАБОТА №14

Разработка приложения. внесение и изменение данных в базе данных. Сортировка и поиск данных.

1. Цель работы

1.1. Разработка в системе программирования Delphi приложения, предназначенного для создания и обслуживания базы данных.

1.2. Знакомство с различными видами представления меню.

1.3. Знакомство с основными методами по формированию состава полей набора данных при выполнении приложения.

2. Приборы и оборудование

2.1. Методические указания.

2.2. Программное обеспечение Delphi.

3. Порядок выполнения работы

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

3.2. При помощи утилиты BDE Administrator создать псевдоним базы данных и запомнить его в созданном каталоге.

3.3. Запустить утилиту Database Desktop, установить умалчиваемый псевдоним.

3.4. 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.

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

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

3.5. Сохранить созданную структуру в соответствующих файлах. Занести в описанные таблицы соответствующие записи.

3.6. 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».

3.7. На форме «Поставщики» должен открываться список со всеми поставщиками, включающий поля «Наименование» (сортировка по наименованию), «Адрес», «Телефон», «WWW адрес» и сведения о количестве поставщиков.

3.8. На форме «Склад» должен открываться список всех имеющихся на складе товаров (поля – «Наименование типа товаров», «Цена», «Дата поступления», «Количество», «Стоимость», группировка по типу товара) и общая их стоимость. Пример формы «Склад» приведен на рисунке 12.1.

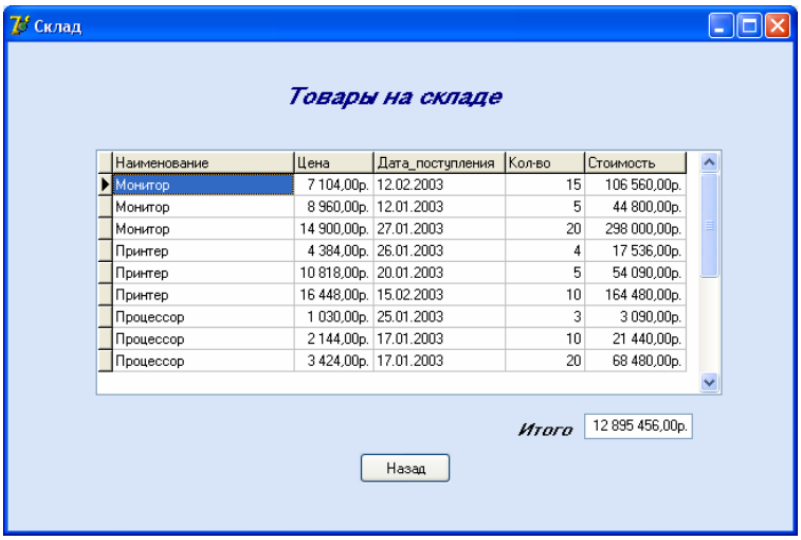


Рисунок 12.1 – Форма «Склад»

3.9. Сохраните и запустите проект.

3.10. Оформите отчет, сделайте выводы о проделанной работе.

4. Содержание отчёта:

4.1. работы.

4.2. Цель работы.

4.3. Приборы и оборудование.

4.4. Выполнение работы.

4.5. Выводы.

5. Контрольные вопросы:

5.1. Для чего предназначена утилита BDE Administrator?

5.2. Как создать псевдоним базы данных?

5.3. Для чего предназначена утилита Database Desktop?

5.4. Как установить умалчиваемый псевдоним базы данных?

5.5. Что можно задать в окне Create Table?

5.6. Что задается в окне определения структуры таблицы базы данных?

5.7. Какие типы данных используются для таблиц Paradox?

5.8. Для каких полей в столбце Key помещается звездочка?

5.9. Как запомнить структуру таблицы?

5.10. Какие страницы содержит палитра компонентов?

5.11. Как и для каких целей определяется ссылочная целостность между таблицами?

5.12. Какие визуальные компоненты со страницы Standard использовались при выполнении лабораторной работы?

5.13. Что означает «визуальные» и «невизуальные» компоненты?

5.14. Какие свойства компонента Table задаются для связи приложения с таблицей данных?

5.15. Для чего используется компонент Data Source?

5.16. Для каких целей создается форма с подчиненной формой? Какие действия выполняются для связи подчиненной формы с главной формой?

5.17. С помощью какого компонента можно задать имя таблицы?

5.18. Каким образом можно изменить названия столбцов на русские наименования?

5.19. Для чего предназначен компонент Button? Что необходимо задать для компонента Button, чтобы он действовал как кнопка?

ПРИЛОЖЕНИЕ А

Теоретические сведения

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).

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 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.

Определение ссылочной целостности между таблицами

Целостность – свойство базы данных, означающее, что она содержит полную, непротиворечивую и адекватно отражающую предметную область информацию. 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 1) 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;

— 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.

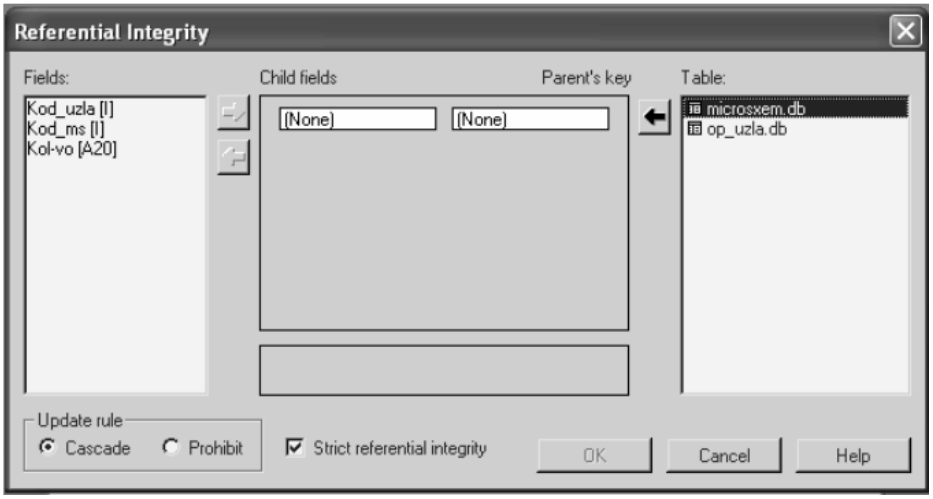


Рисунок 1 – Окно определения ссылочной целостности

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.

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 2). 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 (). 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 3 а) для изменения характеристики столбцов (перейти от неявно определяемым столбцам

к явно определяемым) щелкнуть по кнопке Add All Fields ( ) на инструментальной панели окна редактора (рисунок 3 б);

– 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 4). При этом перемещение указателя в главной таблице приводит к автоматической смене информации в отображаемых данных дочерней таблицы.

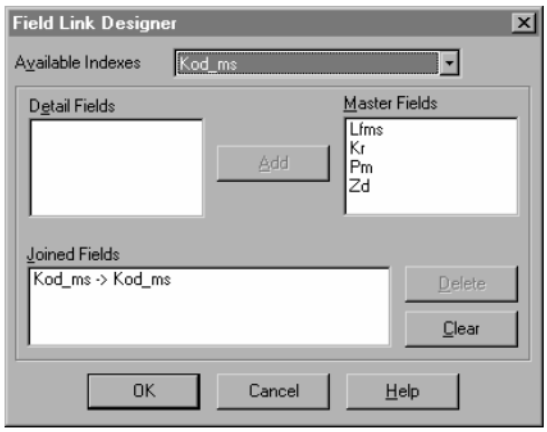


Рисунок 2 – Окно редактора связей

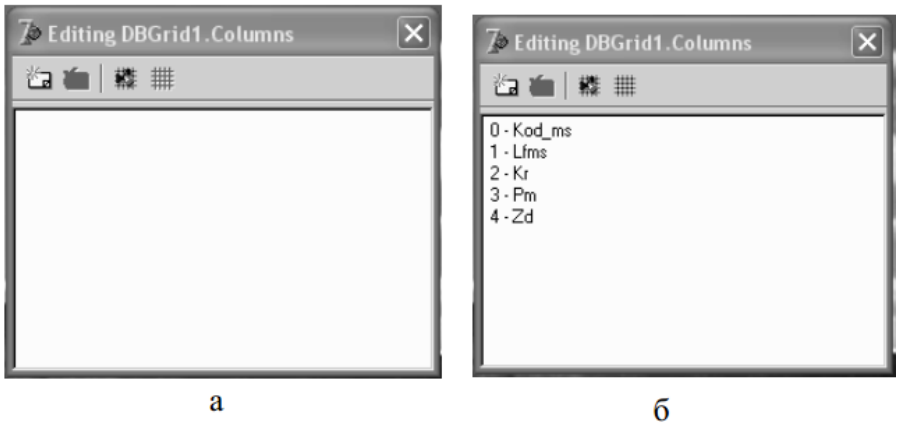


Рисунок 3 – Окно редактора столбцов
(а - пустой список столбцов; б - заполненный список)

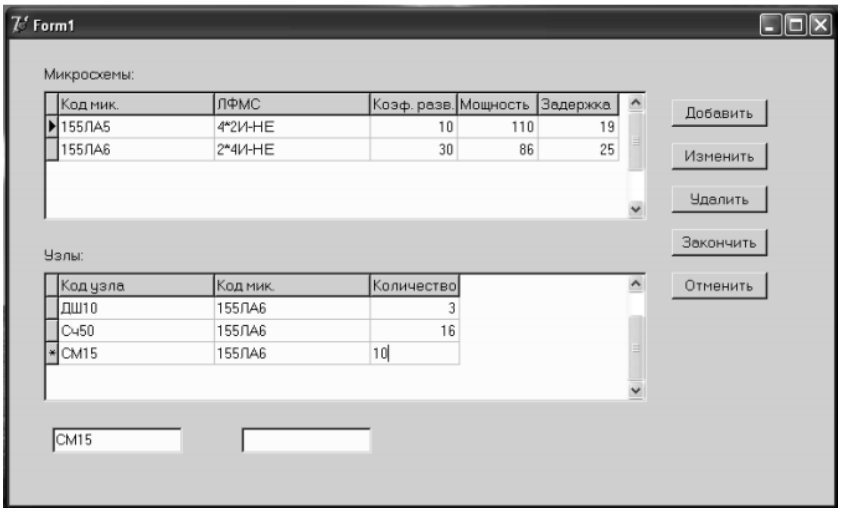


Рисунок 4 – Работа формы с главными и подчиненными наборами данных

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 несоот-

ветствия типов. На рисунке 5 показано окно визуального построителя SQL Builder с установленными связями межу таблицами.

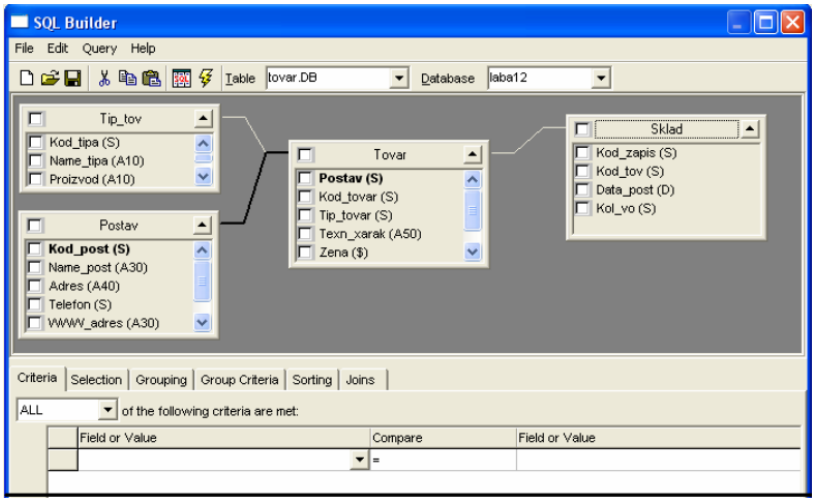


Рисунок 5 – Формирование запроса к базе данных с использованием визуального построителя

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.

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;».

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.DeleteButt onClick(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 на бор 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.CancelBut tonClick(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. Для связи компонентов с полями необходимо поместить в их свойства DataSource значения соответственно DataSource1 и 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.

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 ре жиме 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 не большую 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. 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.

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 рассматриваемом примере на форму Form3 необходимо поместить два компонента GroupBox, как показано на рисунке 5. 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 2. 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.

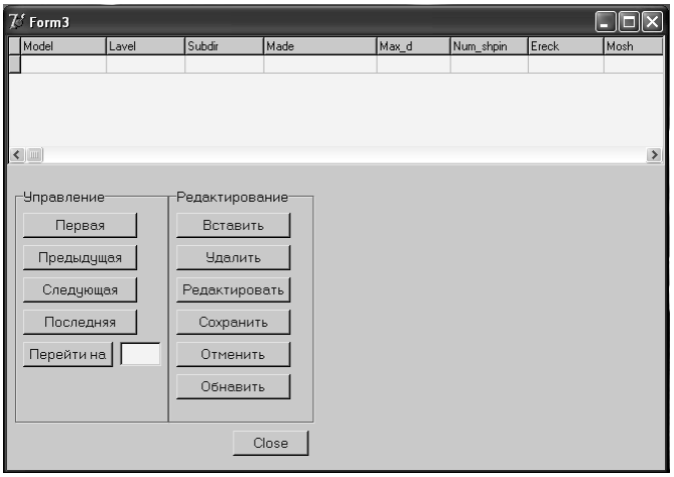


Рисунок 5 – Форма с кнопками

Для кнопки Button6 «Вставить»:

«procedure Tform3.Button6C lick(Sender: TObject);
begin
Table1.Insert;
end;»,

Для кнопки Button7 «Удалить»:

«procedure Tform3.Button7C lick(Sender: TObject);
begin
Table1.Delete;
end;».

Для кнопки Button8 «Сохранить»:

«procedure Tform3.Button8C lick(Sender: TObject);
begin
Table1.Post;
end;».

Для кнопки Button9 «Отменить»:

«procedure Tform3.Button9C lick(Sender: TObject);
begin
Table1.Cancel;
end;».

Для кнопки Button10 «Обновить»:

«procedure Tform3.Button10 Click(Sender: TObject);
begin
Table1.Refresh;
end;».

Для каждого компонента Button необходимо задать соответствующие значения в свойстве Caption.

ПРАКТИЧЕСКАЯ РАБОТА №15

Сложные запросы. Фильтрация данных.

1. Цель работы

1.1. Создание пользовательского приложения по работе с базой данных, которое позволяет управлять данными и осуществлять фильтрацию записей.

1.2. Создание приложения по работе с базой данных, которое использует структурированный язык запросов SQL в среде программирования Delphi.

2. Приборы и оборудование

2.1. Методические указания.

2.2. Программное обеспечение Delphi.

3. Порядок выполнения работы

3.1. Изучить теоретические указания.

3.2. Открыть проект, созданный в предыдущей лабораторной работе.

3.3. На основании рисунка 1, приложение А, на форме «Склад», создать фильтры по выражению «По наименованию типа товара» и «По максимальной цене». Поместить две кнопки Button: одну для выполнения фильтрации, вторую для её отмены. Создать вычисляемое поле.

3.4. Создать новую форму. Осуществить набор данных, состоящий из полей: «Наименование товара», «Производитель», «Технические характеристики», «Наименование поставщика», «Цена», вычисляемое поле «Итого». Сохраните и запустите проект.

3.5. Осуществите выборку данных из таблицы «Типы товаров», результат выборки: «Наименование товара», «Производитель» – Samsung.

3.6. 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).

3.7. Сохраните и запустите проект.

3.8. Оформите отчет, сделайте выводы о проделанной работе.

4. Содержание отчёта:

4.1. работы.

4.2. Цель работы.

4.3. Приборы и оборудование.

4.4. Выполнение работы.

4.5. Выводы.

5. Контрольные вопросы:

5.1. Что такое SQL?
5.2. В чем заключаются особенности языка SQL в Delphi?
5.3. Для чего предназначен SQL?
5.4. Для чего предназначен оператор SELECT?
5.5. Что включает синтаксис оператора SELECT?
5.6. Для чего предназначено средство SQL Builder?
5.7. В чем заключается достоинство SQL Builder?
5.8. Для чего используется компонент Query?
5.9. Какие панели используются в SQL Builder для уточнения запроса?
5.10. Для чего используется флажок в изображении таблицы в SQL Builder?
5.11. Какая команда позволяет сменить псевдоним таблицы?
5.12. Что формируется с помощью панели Criteria?
5.13. Что формируется с помощью панели Selection?
5.14. Что формируется с помощью панели Grouping?
5.15. Что формируется с помощью панели Sorting?
5.16. Как осуществить связывание таблиц в программе SQL Builder?

ПРИЛОЖЕНИЕ А

Теоретические сведения

Создание фильтрации по выражению

Рассмотрим в качестве примера обработчики событий формы, использующей фильтрацию записей набора данных по выражению. Вид формы приведён на рисунке 1.

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 происходит 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;».
```

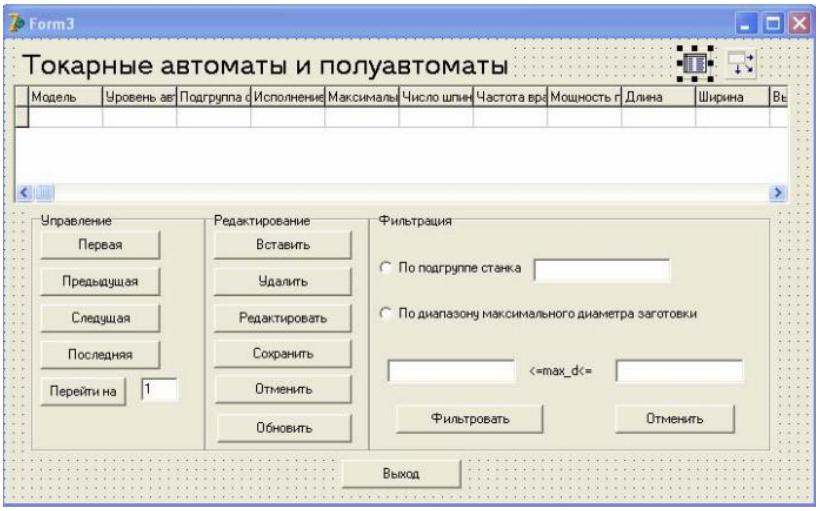


Рисунок 1 – Пример формы с фильтрацией по выражению.

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 => Про граммы => 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 (). 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 5. Для создания заголовка формы помещается компонент Label, для того не обходимо 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 2) 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)  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 обрабатывать. 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.

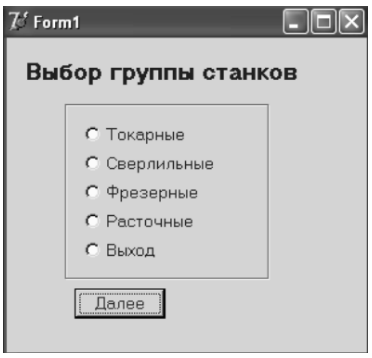


Рисунок 2 – Форма «Выбор группы станков»

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 3). 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 от дельные группы. Его применение сделает интерфейс приложения более удобным и дружественным пользователю.

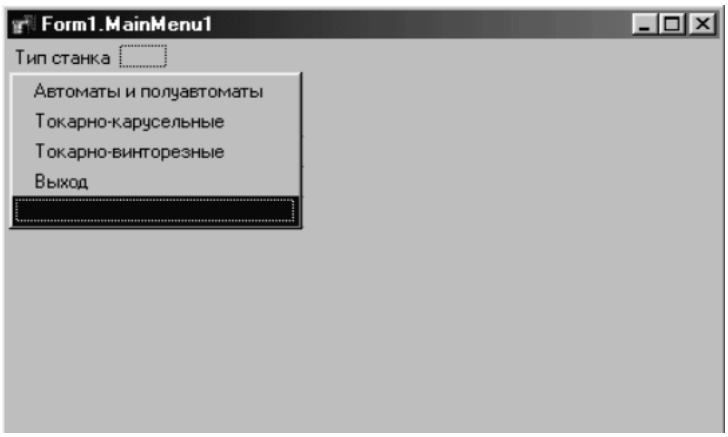


Рисунок 3 – Конструктор меню «Выбор типа станка»

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 3, 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;».
```

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» (Kolvo) 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 па литры 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.Kolvo, M.Lfms, (M.PM*O.Kolvo) 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 4. 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.

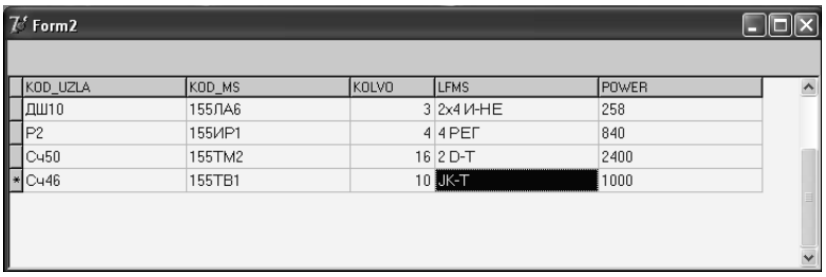


Рисунок 4 – Создание данных из разных таблиц в одном наборе данных

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 несоот-

ветствия типов. На рисунке 5 показано окно визуального построителя SQL Builder с установленными связями межу таблицами.

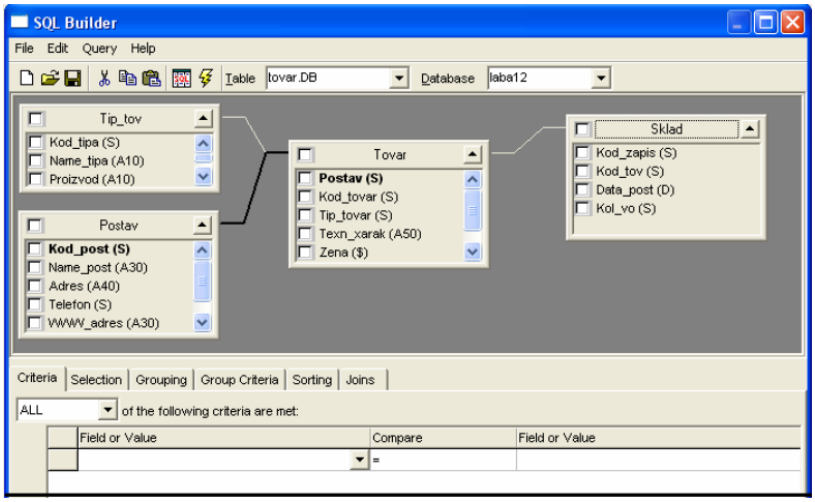


Рисунок 5 – Формирование запроса к базе данных с использованием визуального построителя

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 нажатия клавиши Enter новое название псевдонима вступает в действие. На рисунке 6 приведен выбор отображаемых полей и изменение название псевдонима таблицы.

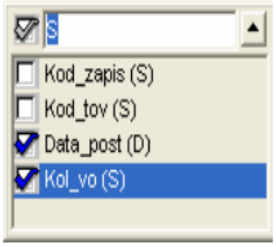


Рисунок 6 – Выбор отображаемых полей и изменение названия псевдонима таблицы

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 7 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 названия логических операций. По умолчанию список содержит значение ALL (ВСЕ), что соответствует операции логического умножения.

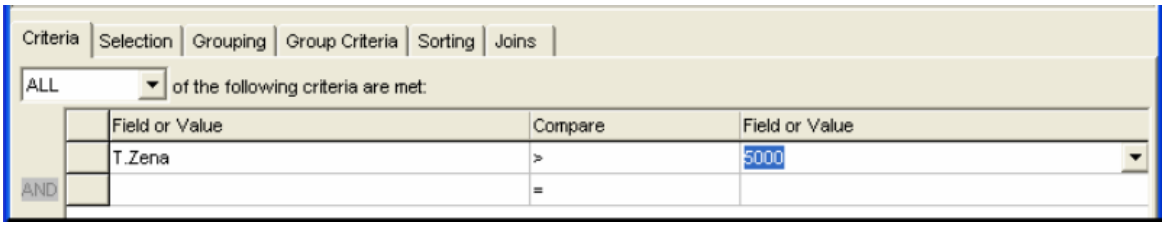


Рисунок 7 – Формирование условия на отбор записей

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).

ПРАКТИЧЕСКАЯ РАБОТА №16

Создание диаграмм и отчётов при работе с базой данных.

1. Цель работы

1.1. 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.

1. Приборы и оборудование

1.1. Методические указания.
1.2. Программное обеспечение Delphi.

2. Порядок выполнения работы

2.1. Изучить теоретические указания.
2.2. 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.
2.3. Запустить утилиту Database Desktop, установить умалчиваемый псевдоним. Задать структуру таблицы «Характеристики процессоров», содержание которой приведено в таблице 14.1. Для таблицы использовать тип Paradox 7. Сохранить созданную структуру в соответствующем файле.

Таблица 14.1 – 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

2.4. 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.
2.5. 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 14.2 приведены значения свойств и программные коды для созданных кнопок.

Таблица 14.2 – 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);
			begin
			form2.DBChart1.RefreshData;
			form2.ShowModal;
			end;

Button2	Переход к форме **Form3** для показа отчета	Отчет	Procedure TForm1.Button2Click(Sender: TObject);
			Begin
			form3.QuickRep1.Preview;
			end;

Button3	Выход из **Form1**	Выход	Procedure TForm1.Button3Click(Sender: Tobject);
			Begin
			Halt;
			end;

2.6. 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 1. Для графика задать заголовок, легенду и марки.

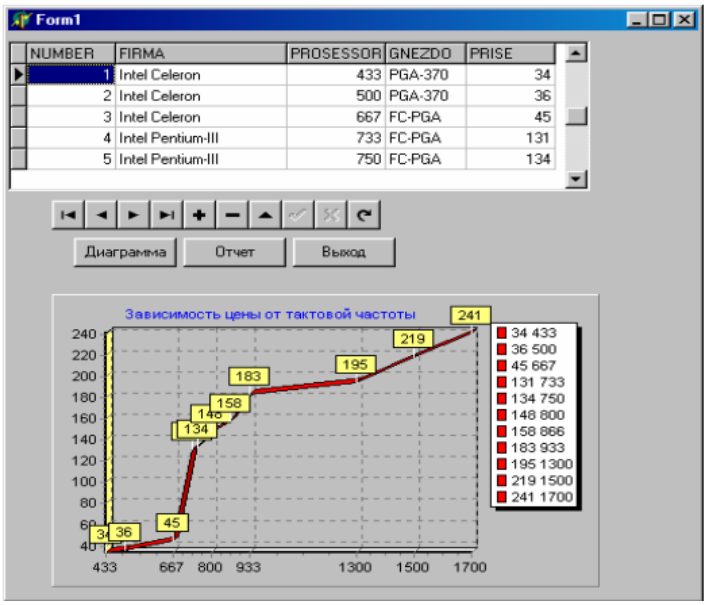
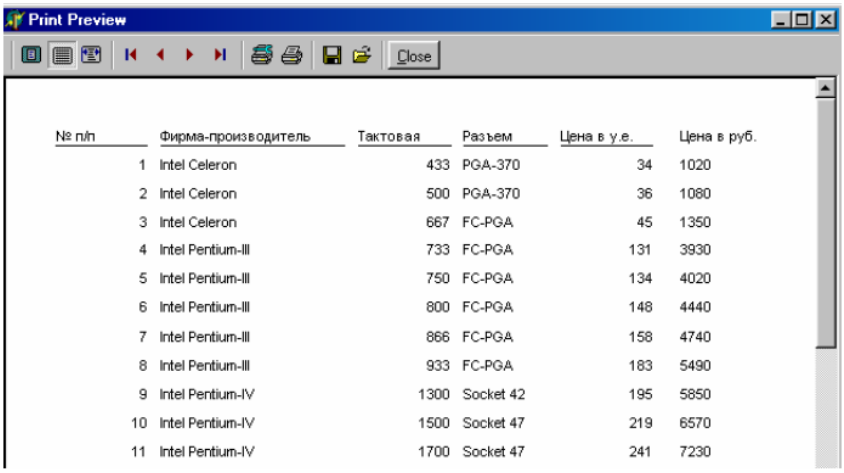


Рисунок 1 – Вид формы Form1

2.7. С помощью Мастера диаграмм TeeChart Wizard построить круговую диаграмму Pie по полю «Тактовая частота», поместив ее на форму Form2.

2.8. Для диаграммы задать заголовок, легенду и установить марки. На форме поместить кнопку Button «Выход» для возврата в форму Form1.

2.9. С помощью программы Rave Designer создать для таблицы «Характеристики процессоров» отчет. В отчет поместить все поля, задать заголовок отчета. Вид отчета приведен на рисунке 2.



№ 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 2 – Вид отчета

2.10. Сохранить и запустить созданное приложение.
2.11. Оформите отчет, сделайте выводы о проделанной работе.

3. Содержание отчёта:

3.1. работы.
3.2. Цель работы.
3.3. Приборы и оборудование.
3.4. Выполнение работы.
3.5. Выводы.

4. Контрольные вопросы:

4.1. Что такое SQL?
4.2. Что такое Мастер диаграмм?
4.3. Какие этапы выполняются при построении диаграммы с помощью
4.4. Мастера диаграмм?
4.5. Какие типы диаграмм позволяет создавать Delphi?
4.6. Что такое легенда?
4.7. Что такое марки?
4.8. Для чего предназначен компонент DBChart?
4.9. Как вызвать окно редактора диаграмм?
4.10. Какие закладки имеет редактор диаграмм?
4.11. Что устанавливается для диаграммы, построенной с помощью
4.12. компонента DBChart ?
4.13. Как выбирается источник данных для диаграмм?
4.14. Какая программа является визуальным конструктором отчетов?
4.15. Как создается отчет при помощи Мастера отчетов?
4.16. Что такое отчет?
4.17. Что такое простой отчет?
4.18. Какие компоненты программы Rave Designer использовались?

4.19. Какой компонент предназначен для создания полосы отчета, на которой располагаются другие компоненты отчета?

4.20. Из каких частей состоит окно проектировщика отчетов?

4.21. Какие способы создания простого отчета можно использовать?

4.22. Какой компонент используется для создания заголовка отчета?

4.23. Какие компоненты используются для ввода текста?

ПРАКТИЧЕСКАЯ РАБОТА №17
РАЗРАБОТКА ПРОСТОГО ПРИЛОЖЕНИЯ БАЗЫ ДАННЫХ

Цель работы

Изучить технологию доступа к данным BDE.

Научиться разрабатывать приложения базы данных, используя технологию доступа к данным BDE.

Приборы и оборудование

Методические указания.

Программное обеспечение Delphi.

Порядок выполнения работы:

Изучить основные теоретические сведения (приложение А).

Создайте новый проект в Delphi.

Создайте псевдоним Базы данных. Для чего откройте SQL Explorer (Database -> Explore). Затем запустите утилиту BDE Administrator… (Object -> BDE Administrator…). Далее в главном меню утилиты выполните команду Object -> New…

Выберите драйвер для доступа к базе данных (STANDARD). Переименуйте созданный псевдоним на номер_группы DB.

Создайте директорию, в которой будет сохранена база данных (В правой части в качестве значения свойства Path указывается путь к базе данных). После чего сохраните созданный псевдоним (Object -> Apply).

Создайте базу данных, используя утилиту Database Desktop (Tools -> Database Desktop)

Установите рабочую директорию (File -> Working Directory). В области Aliases выберите только что созданный псевдоним.

Создайте таблицу Jurnals, в соответствии с приложением 1 (Object -> New… ->Table). В появившемся окне выберите тип таблицы (Paradox 7). В столбце FieldName указываются имена полей создаваемой таблицы. Столбец Type задает их типа. Для строковых полей в столбце Size указывается их размер. А в столбце Key помечаются символом звездочки те поля, которые будут входить в состав первичного ключа.

Аналогично создайте таблицу Soderjanie.

Создайте индекс для второй таблицы, для чего в списке Table properties необходимо выбрать значение Secondary Index и нажать кнопку Define. В качестве индекса выберите атрибут kod_jurnal. Сохраните индекс с именем Jurnal_ind. Определите ссылочную целостность между таблицами, для чего в таблице Soderjanie выберите в списке Table properties элемент Referential Integrity и нажмите кнопку Define. Выберите атрибуты, по которым устанавливается целостность, и сохраните ее с именем Jurnal_ref.

Создайте модуль данных (File -> New… -> Data Module), в котором расположите два компонента TTable, которые находятся на вкладке BDE и два компонента TDataSource. Для обоих компонентов TTable в свойство Database Name выберите имя псевдонима БД и установите свойство TableName. Переименуйте компоненты TTable в соответствии с наименованиями таблиц, используя свойство Name, а компоненты TDataSource JurnalDS и SoderjanieDS соответственно. Свяжите компоненты таблицы и источники данных. При помощи свойства Dataset надо увязать Jurnal с JurnalDS, а Soderjanie с SoderjanieDS. Определите отношение между таблицами, для чего в свойстве MasterSource компонента Soderjanie необходимо указать родительский источник данных. Далее в свойстве MasterFields этого же компонента необходимо указать поля по которым устанавливается связь. Затем на основной форме нужно разместить два компонента TDBGrid, располагающихся на вкладке Data Controls.

Модуль данных надо подключить к главному модулю приложения одной строкой кода : `user DataModule;`

Установите связь компонента TDBGrid с компонентом TdataSource1 расположенным в модуле данных при помощи свойства DataSource. Разместите на форме компонент DBNavigator и свяжите его с компонентом TDBGrid через свойство DataSource.

Проделайте аналогичные действия для TdataSource2. Оформите форму. Запустите и проверьте ее работоспособность.

Оформите отчет, сделайте выводы о проделанной работе.

Содержание отчёта:

работы.
Цель работы.
Приборы и оборудование.
Выполнение работы.
Выводы.

Контрольные вопросы:

Для чего предназначен BDE?
Для чего предназначены утилиты SQL Explorer, BDE Administrator?
Как загрузить утилиту SQL Explorer, BDE Administrator?
Что такое псевдоним БД?
Как создать псевдоним БД?

Где и как используется псевдоним?
Что такое Data Module?
Как подсоединить Data Module к приложению?
Перечислите визуальные компоненты для работы с базами данных.
Перечислите невизуальные компоненты для работы с базами данных.
Перечислите основные свойства компонента Table.
Перечислите основные свойства компонента DataSource.

ПРИЛОЖЕНИЕ А

Теоретические сведения

1. СРЕДСТВА DELPHI ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ

1.1. Borland Database Engine

Традиционным для системы Delphi способом работы с базами данных является использование процессора баз данных Borland Database Engine (BDE). Разработанный фирмой Borland унифицированный программный интерфейс BDE позволяет выполнять доступ к данным как с использованием традиционного record-ориентированного (навигационного) подхода, так и с использованием set-ориентированного (реляционного) подхода, принятого в SQL-серверах баз данных. Универсальный механизм доступа к данным, которым является BDE, применяется в средствах разработки фирмы Borland (Delphi, C++ Builder), а также в некоторых других продуктах.

BDE инсталлируется вместе с Delphi, обеспечивает доступ к локальным базам данных, расположенным на том же компьютере, и к удалённым базам, расположенным на сервере. BDE предоставляет очень гибкий механизм управления базами данных, позволяющий приложениям, созданным в среде Delphi, получать информацию из баз данных наиболее популярных форматов.

BDE представляет собой набор динамических библиотек и драйверов, обеспечивающих доступ к данным. В составе BDE поставляются стандартные драйверы, обеспечивающие доступ к базам данных Paradox, dBase, FoxPro и текстовым файлам. Эти драйверы устанавливаются автоматически вместе с ядром процессора. Доступ к данным серверов SQL обеспечивает отдельная сис драйверов Borland SQL Links. Эти драйверы нужны при разработке приложений для серверов Oracle, Sybase, Informix и InterBase. Драйверы SQL Links необходимо устанавливать дополнительно. Кроме того, в BDE есть возможность подключения любых драйверов ODBC.

В Delphi реализовано достаточно большое количество технологий доступа к данным, однако общие подходы и последовательность действий при разработке приложений баз данных почти одинаковы.

1.2. Утилиты для работы с базами данных в Delphi

1.2.1. BDE Administrator

Утилита BDE Administrator (bdeadmin.exe) предназначена для конфигурирования BDE, позволяет устанавливать параметры псевдонимов баз данных, драйверов и параметры, общие для всех баз данных. Настройки BDE сохраняются в файле idapi32.cfg. Окно программы содержит две области (рисунок 1). В левой области расположены страницы Databases и Con-

figuration. Правая область используется для вывода сведений об объекте, выбранном слева. На странице Configuration приведены сведения о драйверах баз данных и установках BDE. На странице Databases приведены псевдонимы имеющихся на компьютере баз данных. Здесь же можно создавать и редактировать псевдонимы.

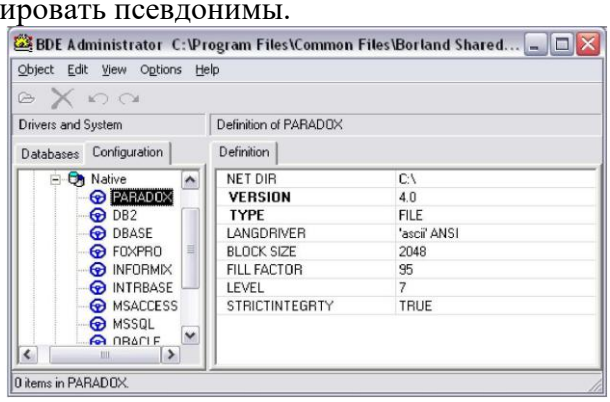


Рисунок 1 – Окно утилиты BDE Administrator

1.3. Псевдонимы

При работе с базами данных во многих случаях удобнее пользоваться псевдонимами, а не просто указывать путь доступа к таблицам базы данных. Псевдоним (alias - алиас) - это известное разработчику и BDE имя базы данных. В BDE с псевдонимом ассоциируются параметры, используемые для соединения с базой данных: формат БД, путь к её файлам, языковой драйвер, имя сервера, имя пользователя, режим открытия и т.п.

Псевдоним сохраняется в отдельном конфигурационном файле на диске и позволяет исключить из программы прямое указание пути доступа к базе данных. Такой подход даёт возможность располагать данные в любом месте, не перекомпилирую при этом программу.

Для создания псевдонима в утилитах BDE Administrator и SQL Explorer необходимо выполнить следующие действия:

- на левой панели выбрать страницу Database;
- через всплывающее меню или меню Object выбрать команду New;
- в окне New Database Alias (рисунок 2) выбрать драйвер для работы с БД и нажать OK. При работе с БД Paradox выбрать Standard;

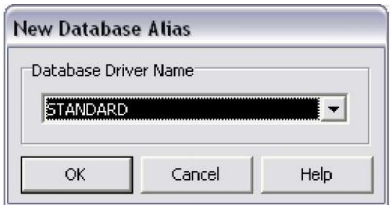


Рисунок 2 – Окно для выбора драйвера.

- на левой панели записать имя;
- на странице Definitions (правая панель) в поле Path (рисунок 3) указать путь к файлам БД: щёлкнуть на строке Path и с помощью кнопки обзора найти нужную папку;
- через всплывающее меню для левой панели или меню Object выбрать команду Apply.

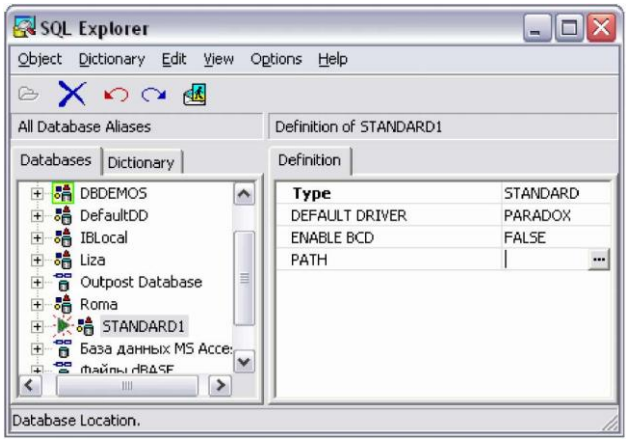


Рисунок 3 – Задание расположения БД.

Дополнительная информация, сообщаемая при создании псевдонима, зависит от типа выбранной базы данных. После создания нового псевдонима его имя вносится в общий список псевдонимов.

1.4. Обзор компонентов для работы с базами данных

Компоненты, используемые для работы с БД, находятся на страницах:

Data Access – невизуальные компоненты, предназначенные для организации доступа к данным;

Data Controls – визуальные компоненты для отображения данных;

dbExpres – компоненты для создания приложений, использующих технологию dbExpress;

BDE – компоненты для создания приложений, использующих BDE;

ADO (Delphi 7) или ***dbGo*** (Delphi 2005 и Delphi 2006) – компоненты для создания приложений по технологии ADO;

InterBase – компоненты для работы с сервером InterBase.

Таким образом, в Delphi предусмотрены специальные наборы компонентов, обеспечивающие доступ к данным при использовании разных технологий, и наборы компонентов, отображающие данные. Компоненты доступа к данным позволяют осуществлять соединения с БД, производить выборку, копирование данных и т.п.

Компоненты для отображения данных позволяют выводить данные в виде таблиц, полей, списков. Отображаемые данные могут быть текстового, графического или произвольного форматов.

Компоненты для работы с базами данных можно разделить на три группы: множества данных (data sets); визуальные компоненты баз данных (data- aware controls) и источники данных (data sources).

Множества данных - это невизуальные компоненты, которые взаимодействуют с BDE и обеспечивают доступ к данным в таблицах. Наиболее важные из них - компоненты Table и Query.

Визуальные компоненты баз данных - это управляющие элементы пользовательского интерфейса для просмотра и редактирования данных. Многие из них дублируют обычные управляющие компоненты: DBEdit, DBCheckBox, DBRadioGroup, DBImage и др.

Источники данных - это невизуальные компоненты, исполняющие роль трубопроводов между множествами данных и визуальными компонентами баз данных. Используя введённые понятия, можно уточнить структуру приложения, осуществляющего доступ к данным через BDE.

1.5. Модули данных

Модуль данных - это контейнер для невизуальных компонентов доступа к базе данных. Для создания модуля данных надо выполнить команду File|New|Other и в окне New Items выбрать Data Module.

Модуль данных является объектом класса `TDataModule`, в него можно помещать только невизуальные компоненты и задавать для компонентов доступа к данным обработчики событий. Для модуля данных определено всего несколько свойств `(Name, Tag)` и событий `(OnCreate, OnDestroy)`, так как в отличие от формы его непосредственным предком является класс `TComponent`. Использование модуля данных позволяет отделить логику обработки данных от логики работы пользовательского интерфейса.

Для форм и модулей данных, создаваемых в приложении, Delphi использует сквозную нумерацию. Для подсоединения модуля данных используется команда File|Use Unit.

При разработке приложений целесообразно поместить множества данных и источники данных в модуль данных, а визуальные компоненты - на формы.

1.6. Невизуальные компоненты для работы с данными

1.6.1. Компонент Table

Компонент Table обеспечивает доступ к таблицам базы данных, создавая набор данных, структура полей которого повторяет таблицу БД. Набором данных называют записи одной или нескольких таблиц, переданные в приложение в результате активизации компонента доступа к данным.

С помощью компонента Table можно организовать доступ к любой записи таблицы или их подмножеству. Компонент Table содержит все необходимые свойства, события и методы для создания, удаления, модификации, сортировки, фильтрации и поиска записей в таблице.

1.6.2. Компонент DataSource

Компонент DataSource[2] обеспечивает взаимодействие набора данных с компонентами для отображения данных. С каждым компонентом доступа к данным должен быть связан как минимум один компонент DataSource. С одним компонентом DataSource может быть связано несколько визуальных компонентов.

1.7. Визуальные компоненты для работы с данными

Большинство визуальных компонентов для работы с данными похожи на соответствующие компоненты для обычных приложений. Основное отличие заключается в том, что компоненты для работы с базами данных содержат свойства, позволяющие указать источник данных `(DataSource)` и поле, из которого компонент получает данные `(DataField)`.

Названия компонентов начинаются с букв DB. Большинство компонентов предназначено для отображения текущего значения одного поля: DBEdit, DBCheckBox, DBRadioGroup, DBText и др. Для вывода и редактирования данных поля Memo используются компоненты DBMemo и DBRichEdit. Для просмотра изображений предназначен компонент DBImage. Отображать данные графически позволяет компонент DBChart.

Кроме того, имеются компоненты, предназначенные для использования только в приложениях баз данных. Это DBNavigator и компоненты DBLook- upComboBox, DBLookupListBox для синхронного просмотра данных из связанных таблиц.

1.7.1. Компонент DBNavigator

Компонент DBNavigator является удобным средством перемещения по записям таблицы. Представляет собой панель с кнопками, построенную по типу панелей управления электронными устройствами. DBNavigator обеспечивает:

- прокрутку записей поодиночке вперёд и назад;
- переход к первой или последней записи;

---

[2] *Полный перечень свойств и методов компонентов можно посмотреть в справочной системе.*

- вставку новой записи;
- удаление текущей записи;
- переход в режим редактирования текущей записи;
- внесение изменений в таблицу;
- отмену сделанных в текущей записи изменений;
- обновление отображаемых значений. Это необходимо в тех случаях, когда данные в таблице БД могут изменяться другими приложениями.

Не все из перечисленных возможностей навигатора бывают нужны. Можно оставить только те кнопки, которые реализуют требуемую функциональность. Для управления видимостью кнопок используется свойство `visible- Buttons:` для каждой кнопки отдельно выбирается `true` или `false.`

Свойство `flat` управляет внешним видом кнопок: предусмотрены объёмное `(false)` и плоское `(true)` изображения. Свойства `Hints` и `ShowHint` предназначены для вывода подсказок. По умолчанию подсказки к кнопкам выводятся на английском языке и содержат текст, приведённый в свойстве `Hints.` Если требуется изменить подсказку к отдельной кнопке или создать подсказки на русском языке, то надо открыть строковый редактор свойства `Hints` и записать текст подсказок.

1.7.2. Компонент DBGrid

Визуальный компонент DBGrid предназначен для организации табличного просмотра и редактирования данных. Внешний вид данных, отображаемый DBGrid, по умолчанию соответствует структуре набора данных. Компонент DBGrid часто называют сеткой.

Для перемещения по записям используются полосы прокрутки и клавиши управления курсором. Для изменения данных достаточно установить курсор в нужную ячейку и ввести другое значение. Новая пустая строка создаётся в позиции указателя нажатием на клавишу Insert. Чтобы изменения, сделанные при редактировании и добавлении записи, были внесены в таблицу, необходимо нажать на клавишу Enter или перейти на другую строку. До того как данные были переданы в таблицу, можно клавишей Esc отменить изменения. Для удаления записи используется комбинация клавиш Ctrl+Delete.

Свойство `Options` является составным. Его значения определяют особенности поведения и внешнего представления компонента на экране. По умолчанию свойство `Options` содержит комбинацию значений `[dgEditing, DgTitles, dglndicator, DgColumn-Resize, DgColLines, DgRowLines, DgTabs, DgConfirmDelete, DgCancelOnExit],`

По умолчанию для каждого поля исходной таблицы в DBGrid создаётся отдельный столбец. Такие столбцы называются динамическими. Характеристики динамического столбца определяются свойствами поля, для отображения которого этот столбец используется. Например, название столбец получает по названию поля, а ширина столбца определяется типом данных поля. Это не всегда удобно. При необходимости в DBGrid можно перейти к статическим столбцам, параметры которых задаются независимо от свойств поля. Для формирования статических столбцов используется Редактор колонок.

Компонент DBGrid обычно используют для просмотра данных. Для ввода и изменения данных используют компоненты, работающие с одним полем, так, чтобы на экране были видны поля только одной записи.

ПРИЛОЖЕНИЕ А

Таблица Jurnal

	Field Name	**Type**	**Size**	**Key**
1	Kod_jurnal	Numer-ic		*
2	Name	Alfa	20	
3	Number	Short		
4	Year	Short		

Таблица Soderjanie

	Field Name	**Type**	**Size**	**Key**
1	Auto	+		*
2	Kod_jurnal	Numer-ic		
3	Name_st	Alfa	200	
4	Opisanie	Memo	4	
5	Str	Alfa	3	

ПРАКТИЧЕСКАЯ РАБОТА №18

ИЗУЧЕНИЕ КОМПОНЕНТОВ ADO. СВЯЗЬ С ACCESS ЧЕРЕЗ ADO

6. Цель работы

6.1. Изучить возможности технологии доступа к данным ADO.

6.2. Научиться создавать связь приложения базы данных с базой данных Access, используя ADO.

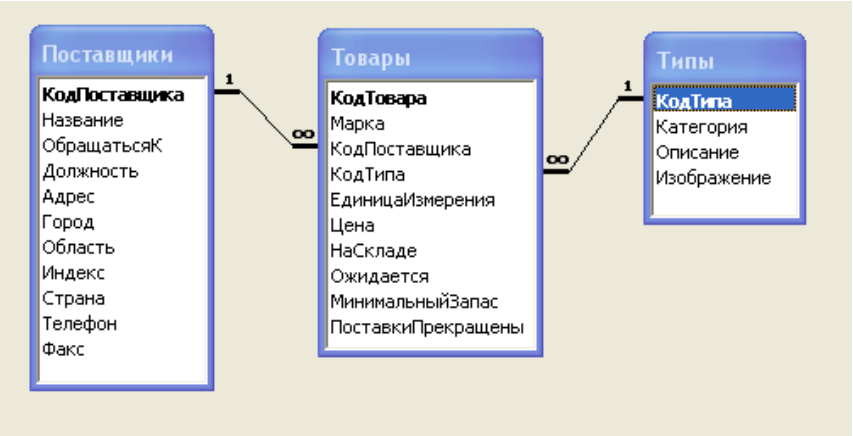
7. Приборы и оборудование

7.1. Методические указания.

7.2. Программное обеспечение Delphi.

8. Порядок выполнения работы

8.1. Создайте базу данных в Access в соответствии с рисунком:



8.2. В новом проекте Delphi создайте отдельный модуль данных. В нем расположите компонент TADOConnection и настроить соединение с базой данных:

- установите соединение с провайдером, для чего для свойства ConnectionString откройте построитель (три точки справа от поля свойства);

- в появившемся окне необходимо нажать на кнопку Build. На вкладке Provider выбрать Microsoft Jet 4.0. OLE DB Provider. На вкладке Connection необходимо указать путь к базе данных. Нажмите на кнопку Test connection. Закройте окно в случае положительного тестирования;

- свойству LoginPrompt присвойте значение true и активируйте соединение (для свойства Connection установить значение true).

8.3. Расположите 3 компонента TADOTable и свяжите их с компонентом TADOConnection при помощи свойства Connection. В свойстве TableName следует выбрать имя таблицы, а затем активизировать компонент. А так же расположите компонент TDataSource и свяжите его с компонентом TADOConnection.

8.4. Расположите на главной форме компонент TDGrid и свяжите его с компонентом TDataSource. А так же TDBNavigator и свяжите его TDataSource.

8.5. Создайте формы для каждой таблицы. И заполните таблицы записями.

8.6. Создайте главную форму для запуска всех форм.

8.7. Оформите отчет, сделайте выводы о проделанной работе.

9. Содержание отчёта:

9.1. работы.

9.2. Цель работы.

9.3. Приборы и оборудование.

9.4. Выполнение работы.

9.5. Выводы.

10.Контрольные вопросы:

10.1. Как организовать доступ к данным по технологии OLE DB, если для нужной СУБД нет OLE DB-драйвера, но есть ODBC-драйвер?

10.2. Приведите схему доступа к данным с применением ADO.

10.3. Почему активно используется доступ к данным с использованием ADO?

10.4. Какие объекты используются в технологии ADO?

10.5. Какие компоненты Delphi используются для организации доступа к данным по технологии ADO?

10.6. Как задаются параметры соединения при разработке в Delphi приложения, использующего технологию ADO?

ПРИЛОЖЕНИЕ А

Теоретические сведения

ADO

Компанией Microsoft был предложен механизм доступа к данным ActiveX Data Objects (ADO), построенный на использовании интерфейсов OLE DB. ADO это набор библиотек, содержащих COM-объекты, реализующие прикладной программный интерфейс для доступа к данным и используемые в клиентских приложениях. Технология ADO использует библиотеки OLE DB, предоставляющие низкоуровневый интерфейс для доступа к данным.

ADO становится всё более популярным способом доступа к данным, так как включен в ядро операционных систем семейства Windows, и входит в состав таких популярных продуктов, как MS Office.

Согласно терминологии ADO любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым приложение взаимодействует при посредстве провайдера.

Приложение обращается к данным не напрямую, а через объект OLE DB, который «умеет» работать с разнообразными данными. Технология ADO включает в себя набор объектов и механизмов, обеспечивающих взаимодействие объектов с данными и приложениями. Очень важную роль играют провайдеры ADO, координирующие работу приложений с хранилищами данных различных типов.

Набор объектов и соответствующий провайдер могут быть созданы для любого хранилища данных без внесения изменений в структуру ADO.

Для каждого используемого типа хранилища данных должен существовать провайдер ADO. Провайдер знает, какие данные и где расположены, умеет обращаться к данным с запросами и интерпретировать возвращаемую служебную информацию и результаты запросов для передачи приложениям. При установке соединения через соответствующие компоненты становится доступен список установленных в операционной системе провайдеров.

В технологии ADO используются объекты-перечислители, источники данных, сессии, транзакции, наборы рядов, команды.

Объекты-перечислители выполняют поиск объектов ADO, осуществляющих доступ к источнику данных.

Для соединения с хранилищем данных используются два типа объектов: источники данных и сессии.

Объект-набор рядов обеспечивает работу с данными.

Объект-команда объединяет текстовую команду и механизмы обработки команд. Команды позволяют использовать для работы с данными язык SQL.

МЕХАНИЗМЫ ДОСТУПА К ДАННЫМ, ПОДДЕРЖИВАЕМЫЕ DELPHI

Имеющиеся в Delphi классы и компоненты позволяют быстро и эффективно разрабатывать приложения баз данных. Для удобства использования компоненты разбиты на группы:

5. Компоненты для доступа к данным, реализующие:

— доступ через процессор баз данных BDE, используя ODBC- драйверы или внутренние драйверы BDE;

— доступ через ADO-объекты, в основе которого лежит применение технологии OLE DB;

— доступ к локальному или удалённому SQL-серверу InterBase;

— доступ посредством драйверов dbExpress;

— доступ к БД при многозвенной архитектуре (компоненты страницы DataSnap);

6. Компоненты для связи источников данных с визуальными компонентами, предоставляющими интерфейс пользователя;
7. Визуальные компоненты, реализующие интерфейс пользователя;
8. Компоненты для визуального проектирования отчётов.

Разные механизмы работы с данными имеют схожие схемы. Ранее подробно была рассмотрена схема работы через BDE с использованием внутренних драйверов (с таблицами форматов Paradox, dBase, FoxPro). Работа через BDE с использованием ODBC-драйверов организуется аналогично.

В модуль данных (или в форму) добавляется компонент набора данных (объект класса `TDataSet`) и устанавливается связь с источником данных, определяемая свойством `DataDaseName`. Связь может быть указана по имени БД, каталогу или псевдониму (ограничения зависят от типа источника данных).

В модуль данных (или в форму) добавляется компонент источника данных (`TDataSourse`), являющийся связующим звеном между набором данных и элементами управления, отображающими данные. Свойство `DataSet` компонента типа `TDataSourse` указывает набор данных, формируемый компонентами таких классов, как `TTable` или `TQuery`.

В форму добавляются элементы управления для работы с данными, такие как `TDBGrid`, `TDBEdit`, `TDBCheckbox` и т.п. Они связываются с источником данных через свойство `DataSource`.

При реализации схем доступа к данным через ADO, dbExpress применяются другие компоненты, но подход остаётся тем же.

Предком всех классов наборов данных является класс `TDataSet`. В зависимости от механизма доступа, используемого приложением, базовыми классами набора данных могут быть:

`TTable`, `TQuery`, `TStoredProc` - для однозвенных или двухзвенных приложений, использующих BDE;

`TClientDataSet` - для реализации клиентского набора данных и для многозвенной архитектуры, использующей распределенный доступ;

`TADODataSet` - для приложений, использующих ADO-объекты;

`TSQLDataSet` - для доступа к базе данных посредством dbExpress. Этот класс реализует направленный набор данных. Для такого набора данных не создайся кэш памяти на клиенте, и среди методов доступа возможны только методы `Next` и `First`. Редактирование записей в направленном наборе данных возможно только явным выполнением SQL-оператора `UPDATE` или при установке соединения с клиентским набором данных через провайдера;

`TSQLTable` и `TSQLQuery` - для доступа к базе данных посредством dbExpress.

На рисунке 2 приведена иерархия классов наборов данных библиотеки VCL системы Delphi. При работе с компонентами наборов данных можно обойтись без явного использования компонентов, реализующих соединение с базой данных. Однако некоторые возможности, такие как управление транзакциями или кэшированные обновления, невозможны без компонентов типа `TDatabase` или `TADOConnection`.

Компонент база данных `TDatabase` применяется для соединения с источником данных через драйверы BDE или внешние ODBC-драйверы. Компонент `ADOConnection` используется для создания объекта-соединения при доступе через OLE DB, который реализуется посредством ADO-объектов VCL- библиотеки.

По умолчанию при переходе от одной записи набора данных к другой происходит запись всех сделанных изменений в базу данных. Для того чтобы можно было отменять сделанные изменения или выполнять обновление нескольких записей, применяют кэшированные об-

новления. Они позволяют значительно снизить сетевой трафик за счет того что все сделанные изменения хранятся во внутреннем кэше. И при переходе от одной записи к другой информация в базу данных не передаётся. Чтобы включить режим кэшированного обновления, следует установить значение свойства `CachedUpdates` равным `True` для компонента набора данных. Для присвоения кэшированного обновления вызывается метод `ApplyUpdates`, а для отмены - `CancelUpdates`.

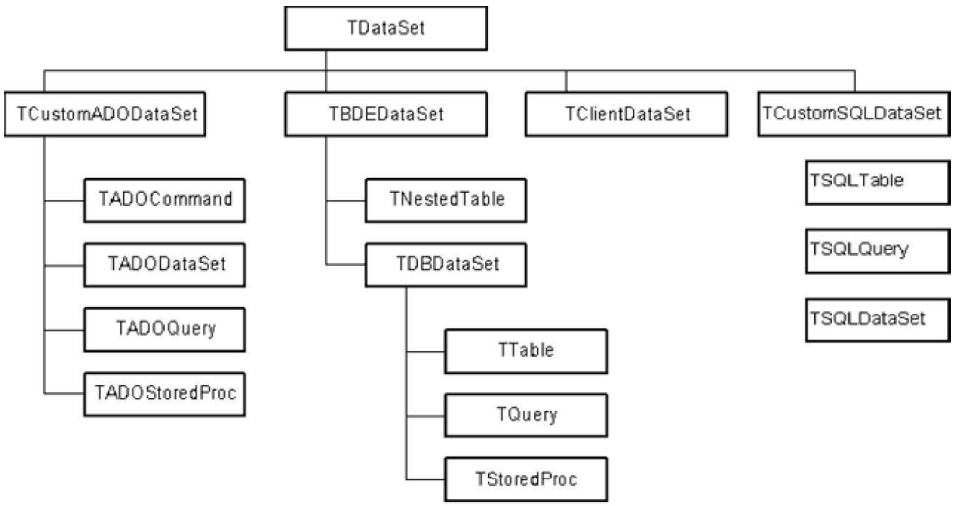


Рисунок 2 – Иерархия классов, реализующих доступ к данным

ТЕХНОЛОГИЯ ADO

Для применения технологии ADO в Delphi 7 предназначены семь компонентов, расположенных на закладке ADO палитры компонентов.

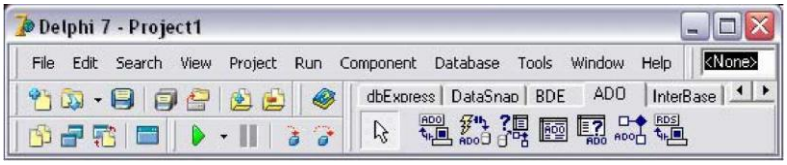


Рисунок 3 – Компоненты страницы ADO

Таблица 1 Назначение компонентов ADO

Название	Описание
ADOConnection	Функционально аналогичен компоненту **Database** закладки BDE. Позволяет указывать местоположение базы данных и работать с транзакциями
ADOCommand	Предназначен для выполнения SQL–команды без возврата результирующего набора данных
ADODataset	Предназначен для получения набора данных из одной или нескольких таблиц БД. Позволяет работать с возвращённым набором данных визуальным компонентам
ADOTable	Аналог компонента **Table**, расположенного на закладке BDE. Используется для доступа к таблице с помощью механизма ADO
ADOQuery	Аналог **Query**. Позволяет формировать запросы к БД, которые возвращают данные из базы (например, командой `SELECT`) или не формируют результирующего набора данных (например, `INSERT`)
ADOStoredProc	Предназначен для вызова процедуры, хранимой на сервере базы данных. Является потомком `TDataSet`, в отличие от BDE и InterBase позволяет возвращать набор данных, поэтому может выступать источником данных в компонентах типа **DataSource**
RDSConnection	Управляет механизмом, который позволяет клиенту получать доступ к объектам, расположенным в другом адресном пространстве или на другом компьютере

Класс `TADOConnection` обеспечивает соединение с данными, доступ к которым реализуется через ADO-объекты. Компоненты ADOConnection используют для доступа к данным OLE DB-провайдеры. Компоненты ADOCommand и ADODataSet связываются с источником данных посредством объекта ADOConnection, указывая ссылку на него как значение свойства `Connection`.

Для идентификации соединения необходимо определить значение свойства `ConnectionString` (строка соединения) компонента ADOConnection, которое может основываться на указании datalink-файла или строки соединения. Если в качестве значения свойства `ConnectionString` указано имя datalink-файла, то настройку соединения можно выполнять автономно от приложения (например, указывая имя базы данных Microsoft SQL Server на текущем ПК).

Реализацию доступа к данным через ADO проще всего рассмотреть на примерах.

Пример. Разработать в `Delphi` приложение для просмотра объектов базы данных MS Access, используя механизм ADO. Последовательность действий

4. Откроем Delphi, создадим приложение и разместим на форме три компонента:
   - ADOTable с закладки ADO;
   - DataSource с закладки Data Access;
   - BDGrid с закладки Data Controls.
5. Свяжем компоненты между собой:
   - установим значение свойства `DataSource` компонента DBGridl в DataSourcel;
   - в свойстве `DataSet` компонента DataSourcel укажем ADOTablel.
6. Зададим параметры соединения компонента ADOTablel:
   - в свойстве `ConnectionString` (рисунок 4) нажмём кнопку с многоточием, в окне редактора параметров соединения установим переключатель в положение Use Connection String и нажмём кнопку Build;

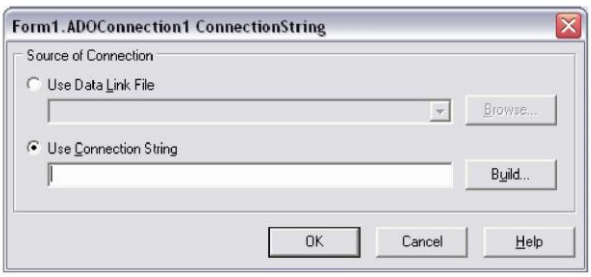


Рисунок 4 – Окно задания свойства `ConnectionString`

– в появившемся окне на вкладке Поставщик данных выберем свойство Microsoft Jet 4.0 OLE DB Provider (рисунок 5);

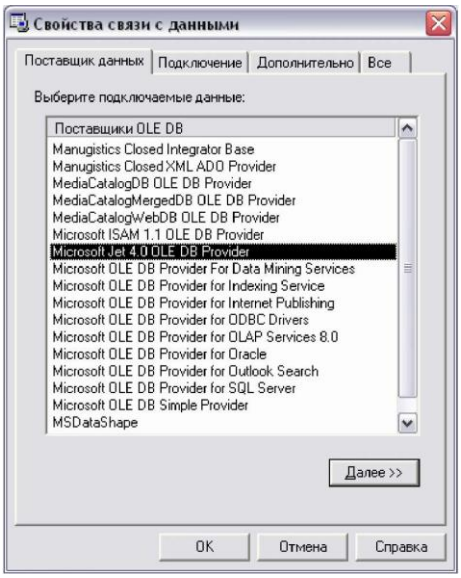


Рисунок 5 – Выбор OLE DB – провайдера.

— перейдём на вкладку Подключение, укажем путь к БД, введём имя пользователя и при необходимости зададим пароль (рисунок 6);

— нажмём кнопку Проверить подключение;

— после завершения проверки подключения перейдём на вкладку Дополнительно и поставим галочки напротив свойств ReadWrite, Share Deny None;

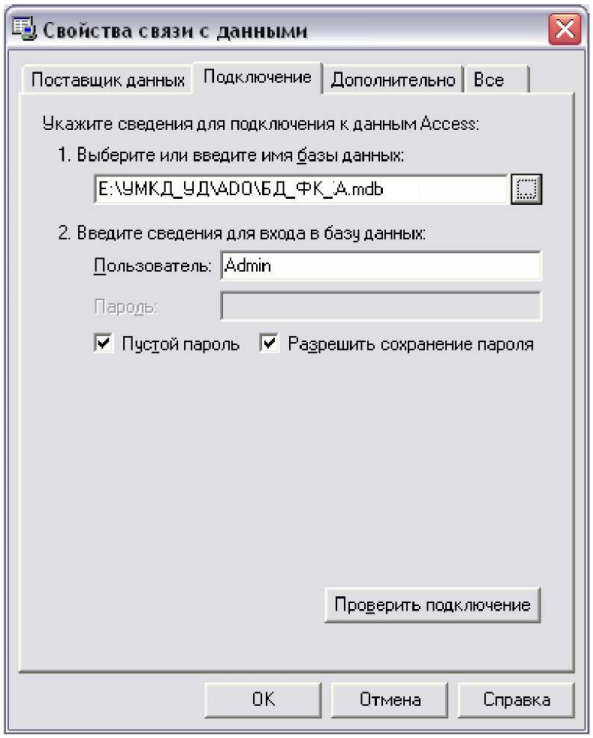


Рисунок 6 – Задание источника данных.

— выберем вкладку Все и проверим сделанные установки. При необходимости изменить значение выберем нужную строчку, нажмём кнопку Изменить значение, в появившемся окне выберем значение и нажмём ОК;

— нажмём два раза ОК;

— в свойстве `TableName` компонента ADOTablel укажем нужную таблицу;

— в свойстве `Active` установим значение `true`.

Если всё сделано правильно, то после задания таблицы компонент DBGridl заполнится данными.

ПРИЛОЖЕНИЕ А

Теоретические сведения

Мастер диаграмм

Одной из наглядных форм представления информации является представление их в виде диаграмм. Сис 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 1.

Мастер предлагает выбрать стиль диаграммы. Есть два стиля диаграмм – связанные с базой данных и несвязанные. В лабораторной работе 14 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 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 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.

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). 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.

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.

При разработке приложения свойства компонента DBChart устанавливаются с помощью Редактора диаграмм, чье окно Editing DBChart. 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. 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 ChartSeries 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 со ответствующей странице Редактора 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 SeriesDataSource 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 выполнении 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.

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. 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 отче том, 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.

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 поместить 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 за головка, 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, 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 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.

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');
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.

ПРИЛОЖЕНИЕ А

Теоретические сведения

Мастер диаграмм

Одной из наглядных форм представления информации является представление их в виде диаграмм. Сис 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 1.

Мастер предлагает выбрать стиль диаграммы. Есть два стиля диаграмм – связанные с базой данных и несвязанные. В лабораторной работе 14 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 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 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.

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). 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.

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.

При разработке приложения свойства компонента DBChart устанавливаются с помощью Редактора диаграмм, чье окно Editing DBChart. 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. 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 ChartSeries 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 со ответствующей странице Редактора 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 SeriesDataSource 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 выполнении 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.

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. 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 отче том, 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.

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 поместить 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 за головка, 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, 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 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.

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');
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.

Список рекомендуемой литературы

Список основной литературы:

1. Кузниченко, М. А. Основы баз данных : учебно-методическое пособие / М. А. Кузниченко. — 2-е изд., стер. — Москва : ФЛИНТА, 2022. — 102 с. — ISBN 978-5-9765-5139-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/266339

2. Махмутова, М. В. Практический подход к проектированию баз данных : учебное пособие / М. В. Махмутова. — 2-е изд., стер. — Москва : ФЛИНТА, 2023. — 159 с. — ISBN 978-5-9765-3694-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/348272

Список дополнительной литературы

1. Щенёва, Ю. Б. Проектирование и разработка клиентсерверных приложений : учебное пособие / Ю. Б. Щенёва. — Рязань : РГРТУ, 2024 — Часть 1 — 2024. — 124 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/494579