ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

МОСКОВСКИЙ ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

В.В. Кондаков

БАЗОВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРИЗИРОВАННЫХ СИСТЕМ УЧЕТА И КОНТРОЛЯ ЯДЕРНЫХ МАТЕРИАЛОВ

Лабораторный практикум

Рекомендовано УМО «Ядерные физика и технологии»
в качестве учебного пособия
для студентов высших учебных заведений

Москва 2007

УДК 621.039.53:004.47(076.5)
ББК 31.46я7
К64

Кондаков В.В. **Базовое программное обеспечение компьютеризированных систем учета и контроля ядерных материалов:** Лабораторный практикум. М.: МИФИ, 2007. – 140 с.

В настоящем практикуме представлены лабораторные работы, посвященные изучению базового программного обеспечения, используемого при создании и эксплуатации компьютеризированные систем учета и контроля ядерных материалов.

Лабораторный практикум предназначен для студентов, специализирующихся в области учета и контроля ядерных материалов, в рамках подготовки инженеров-физиков по специальности 140309 «Безопасность и нераспространение ядерных материалов» направления «Ядерные физика и технологии».

Пособие подготовлено в рамках Инновационной образовательной программы.

Рецензент канд. техн. наук Саманчук В.Н.

ISBN 978-5-7262-0861-9

СОДЕРЖАНИЕ

Лабораторная работа 1
ЗНАКОМСТВО С ОПЕРАЦИОННОЙ СИСТЕМОЙ WINDOWS XP И ОРГАНИЗАЦИЕЙ ЛОКАЛЬНОЙ СЕТИ ЛАБОРАТОРИИ

Цель работы: ознакомить студентов с основными утилитами операционной системы Windows XP, выработать навыки работы в операционной системе, а также ознакомить студентов с физической и программной сторонами организации локальных сетей.

Теоретические основы

Компьютерные сети. В настоящее время большинство персональных компьютеров объединено в сети, начиная от локальных корпоративных сетей и заканчивая глобальной сетью Интернет. Локальные вычислительные сети предприятия могут иметь различную топологию. В учебной лаборатории компьютеры объединены в сеть через коммутатор с помощью витой пары. Сервер сети имеет выход в глобальную сеть. На рис. 1.1 представлена схема соединения компьютеров в ЛВС. Такая топология сети носит название «звезда».

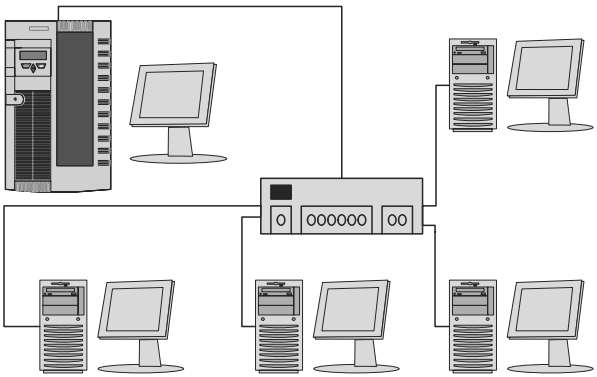

Рис. 1.1. Схема локальной вычислительной сети лаборатории

Локальная вычислительная сеть должна содержать в своем составе следующие устройства.

1. **Сервер**. Компьютер, который будет использоваться в качестве сервера, должен быть размещен в месте, удобном для администратора, кроме того, необходимо наличие кондиционера – в мощных сетях сервер может выделять много энергии, которую нужно отводить. Серверу необходим источник бесперебойного питания (ИБП). ИБП представляет собой аккумулятор, включенный между сетью электроснабжения и компьютером.

2. **Коммутатор**. Аналогичные требования предъявляются к коммутатору. Необходимо наличие закрываемого, вентилируемого помещения, наличие ИБП. Доступ к коммутатору должен быть ограничен, так как через него возможен доступ ко всей сети.

3. **Рабочие станции**. В зависимости от используемого программного обеспечения к компьютерам предъявляются различные требования.

Необходимо предусмотреть возможность сохранения и резервирования информации.

Логическая организация сети. С логической точки зрения компьютерная сеть может быть организована двумя способами. В первом случае все компьютеры сети имеют равные права. Для работы в сети пользователь должен быть зарегистрирован на каждом используемом им компьютере. Такая организация сети называется рабочей группой.

Другой формой логической организации сети является доменная организация. При этом в сети существует выделенный компьютер называемый контроллером домена, который служит для управления объектами (компьютеры, пользователи и др.) во всей сети. Преимуществами доменной организации сети является централизованное управление доступом пользователей ко всем ресурсам сети и то, что администрирование сосредоточено в одном месте.

Операционная система Windows XP. К моменту написания этого практикума единственной операционной системой, сертифицированной на соответствие требованиям Гостехкомиссии РФ по защите информации от несанкционированного доступа (НСД) в ав-

томатизированных системах учета и контроля ядерных материалов, является Windows NT 4.0. Данная операционная система морально и физически устарела и в 2004 г. снята с поддержки разработчика. По инициативе СКЦ Росатома проводится работа по сертификации операционных систем следующего поколения. Windows XP является перспективной операционной системой, сертифицированной Гостехкомиссией РФ для использования при работе с конфиденциальной информацией. В основе данной ОС лежит та же технология, с помощью которой создавались операционные системы Windows NT 4.0, Windows 2000 и Windows 2003 Server. Архитектура этих систем и большинство решений по обеспечению безопасности и организации работы одинаковы. Поэтому, изучая работу современной операционной системы Windows XP, можно получить представление о работе с сертифицированной системой Windows NT 4.0 и ОС Windows 2000.

Все операционные системы, основанные на технологии NT, являются 32-разрядными операционными системами с приоритетной многозадачностью. В качестве фундаментальных компонент в них входят средства обеспечения безопасности и развитый сетевой сервис. Системы обладает способностью функционировать на различных платформах, включая многопроцессорные. Существуют версии Windows XP и Windows 2003 Server, предназначенные для работы на 64-разрядной платформе.

Конфигурация Windows XP. Интерфейс операционной системы Windows XP является дальнейшим развитием интерфейсов Windows NT/2000. Существует возможность настраивать системные установки. Рассмотрим основные процедуры работы с Windows XP.

Процедура входа в систему (LOGON). Если компьютер под управлением Windows XP включен в домен с центральным контроллером под управлением Windows 2000 Server или Windows 2003 Server, то при перезагрузке системы или включении компьютера на экране появляется окно регистрации, в котором указывается имя пользователя, имя компьютера, пароль. Если пользователь

не зарегистрирован, он может быть допущен в систему в качестве гостя с минимальными правами использования ресурсов системы.

При завершении работы необходимо включить процедуру выхода из системы (LOGOFF). При этом производится завершение всех процессов. Питание отключается автоматически. Принудительное отключение питания компьютера может привести к потере информации и повреждению самой системы.

Общий вид экрана (рис. 1.2) при работе системы Windows XP похож на экраны других систем. Основное поле экрана занимает рабочий стол. Вдоль нижнего края размещена панель задач. Панель может постоянно присутствовать на экране, может быть выпадающей, появляющейся при подводе вниз экрана стрелки мыши. На рабочем столе размещаются пиктограммы приложений, наиболее часто используемых пользователем, и рабочие папки, содержащие информацию. Свойства системы настраиваются каждым пользователем и реализуются при каждом его входе в системы. Установив курсор манипулятора – мыши на папке или пиктограмме приложения и щелкнув два раза левой кнопкой, открываем папку или запускаем приложение. При щелчке правой кнопкой мыши открывается меню, позволяющее корректировать свойства приложения или папки.

Для того чтобы стартовать приложение, пиктограммы которого нет на рабочем столе, нажимаем кнопку «Пуск» на панели задач. При этом открывается вложенное меню, в котором можно выбрать необходимое приложение.

Windows XP имеет свой, отличный от предыдущих версий операционных систем, вид меню «Пуск». Однако существует возможность настроить это меню в «классической» форме, т.е. аналогично интерфейсу Windows 2000. Для этого вызываем утилиту Пуск → Настройка → Панель задач и меню «Пуск». В открывшемся окне устанавливаем требуемые параметры.

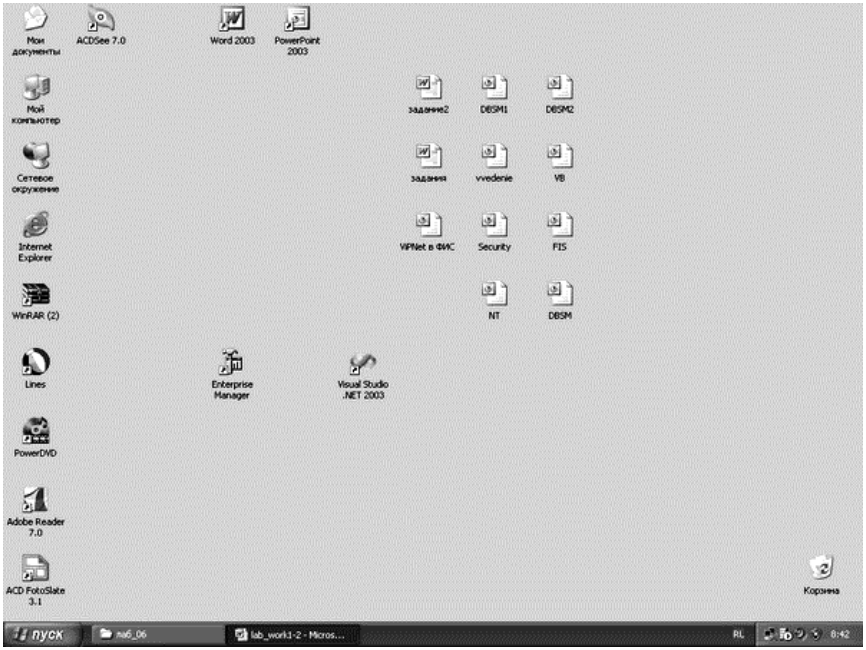

Рис. 1.2. Вид экрана Windows XP

Одно из наиболее часто используемых приложений операционной системы – «Проводник». Эта утилита позволяет работать с файлами и папками пользователя: копировать, уничтожать и т.п. Проводник в операционной системе Windows XP имеет новый интерфейс и большую функциональность, чем эта же утилита в предыдущих операционных системах. Вид запущенной утилиты «Проводник» представлен на рис. 1.3.

Для управления параметрами операционной системы предусмотрен набор утилит, доступ к которым осуществляется через «Панель управления» (рис. 1.4): Пуск → Настройка → Панель управления. «Панель управления» содержит утилиты, меняющие различные установки системы: языковые настройки, установки экрана, управление пользователями, утилиты для определения параметров безопасности и другие.

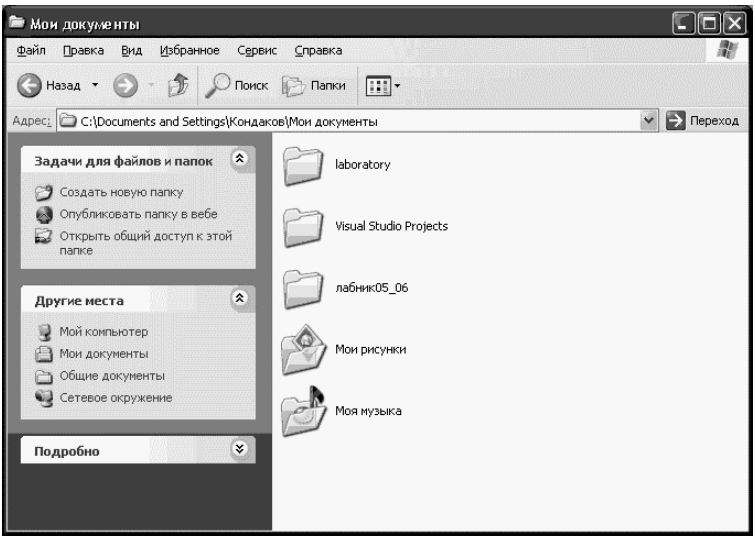

Рис. 1.3. Вид окна приложения «Проводник»

Рис. 1.4. Открытое окно «Панели управления»

Утилитами настройки могут пользоваться все зарегистрированные пользователи системы. Однако существует набор утилит, предназначенных для администрирования. Этими утилитами могут пользоваться только пользователи, входящие в группу Администраторы. Эти утилиты на панели управления сгруппированы в папке «Администрирование». Содержимое этой папки представлено на рис. 1.5. Данные утилиты позволяют устанавливать параметры информационной безопасности, управлять пользователями и группами пользователей, осуществлять аудит событий в системе, устанавливать сетевые подключения, распределять ресурсы компьютера. Здесь же находится утилита управления локальным Web-сервером. Следует отметить, Web-сервер является штатным сервисом операционной системы начиная с Windows 2000. Однако по умолчанию он не устанавливается.

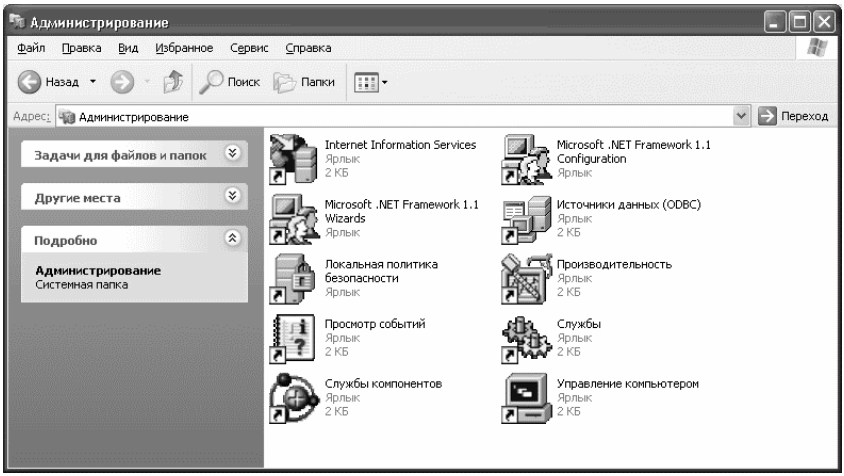

Рис. 1.5. Содержимое папки «Администрирование»

Выполнение работы

1. Осмотр сетевого оборудования и системы физической организации локальной сети.

Преподаватель демонстрирует студентам основное компьютерное и сетевое оборудование, образующее локальную сеть лаборатории. Отобразить на схеме основные элементы локальной сети.

2. Создание группы и регистрация новых пользователей.

Администратор системы демонстрирует студентам создание группы «Magistr» на контроллере домена, регистрирует новых пользователей и объединяет их в новую группу.

3. Вход в систему.

Пользователь включает компьютер, наблюдает процесс загрузки системы, после появления окна Ctrl-Alt-Del – вводит имя и пароль.

4. Изучение рабочего стола Windows XP и получение навыков работы с основными программными утилитами.

• Настроить вид меню «Пуск». Рекомендуется настроить «классический» вид меню для того, чтобы вид меню соответствовал описанию лабораторного практикума. Для этого щелкаем правой клавишей мышки на кнопке «Пуск». В выпадающем меню стартовать опцию «Свойства». Установить классический вид меню пуск.

• Открыть папку «Мой компьютер».

♦ Открыть логический диск D: . Создать папку **Laboratory** на диске D: .

♦ Открыть папку «Мой компьютер» → «Панель управления». Установить классический вид папки. Изучить возможности получения информации о системе с помощью содержащихся в ней утилит:

▪ «Язык и региональные настройки»: открыть закладку «Языки» и нажать кнопку «Подробнее». Установить удобную для себя комбинацию клавиш для переключения языка.

▪ «Сетевые подключения»: Открыть свойства сетевого подключения, определить используемые сетевые протоколы. Попытаться просмотреть свойства протокола.

▪ «Система»: открыть закладку «Имя компьютера», списать конфигурацию компьютера, его имя и домен в который он входит. Открыть закладку «Оборудование», нажать кнопку «Диспетчер устройств», просмотреть список оборудования, попробовать обновить драйвер у видеомонитора. Объяснить результат.

▪ Экран: установить фон экрана, выбрать заставку и задать время ожидания 1 минуту.

▪ «Принтеры и факсы». Осуществить установку сетевого принтера.

▪ Вернуться в папку «Мой компьютер» просмотреть информацию о жестких дисках и записать ее в лист отчетности.

♦ Закрыть папку «Мой компьютер».

• Изучить возможности настройки панели задач. Для этого: нажать кнопку «Пуск», выбрать последовательно опции Настройка → Панель задач. Пометить необходимую опцию.

• Изучить сетевое окружение.

♦ Открыть папку «Сетевое окружение». В боковом меню набрать «Вся сеть» и войти в домен. Перечислить компьютеры, включенные в настоящий момент в домен. Просмотреть свойства компьютеров и определить, какой из них является главным контроллером домена.

♦ Открыть главный контроллер домена. Просмотреть устройства, содержащиеся на нем. Попытаться открыть все диски и устройства, доступные в локальной сети. Зафиксировать и объяснить результат.

• Создать и разместить на рабочей панели ярлыки приложений: Microsoft Office\MS Word 2003 и MS Access 2003, Microsoft Visual Studio.Net 2003\ MS Visual Studio.Net 2003 и Microsoft SQL Server\ Enterprise Manager, Стандартные \ Проводник.

• Запуск приложения.

♦ Запустить текстовый редактор MS Word.

♦ Набрать в редакторе фразу: Файл создан для демонстрационных целей пользователем «_____».

♦ Сохранить файл на диске D: в директории Laboratory под именем «DemoFile “Номер рабочей станции”». Закрыть Word.

• Работа с утилитой «Проводник».

♦ Стартовать утилиту «Проводник». Найти созданный вами файл. Просмотреть его свойства, ничего не меняя.

♦ Подключить в качестве сетевого диска устройство D: на главном контроллере домена. Имя установить по умолчанию.

♦ Открыть подключенное сетевое устройство (диск) и скопировать ваш файл на него.

• Знакомство с системой администрирования.

♦ Открыть «Панель управления» → «Администрирование».

♦ Запустить утилиту «Управление компьютером». Попробовать открыть опцию «Управление дисками». Зафиксировать результат.

♦ Запустить утилиту «Просмотр событий». Просмотреть подробную информацию о последнем системном событии. Зафиксировать его дату и время.

♦ Открыть утилиту «Локальные пользователи и группы». Просмотрите свойства пользователя «Гость». Попробуйте изменить установку «Отключить учетную запись». Объясните результат.

5. Вызвать преподавателя, сдать лист отчетности и выключить компьютер.

Лист отчетности

Студент_____, группа_____

Имя домена_____

Имя компьютера в домене_____

Используемые сетевые протоколы_____

Сетевой принтер_____

В момент выполнения работы в сеть были включены следующие компьютеры:_____

Главным контроллером домена является компьютер _____

Устройства сервера, доступные через локальную сеть _____

Информация о дисках

Диск C имеет размер _____ байт

Диск D имеет размер _____ байт

Последнее зафиксированное в системном журнале событие имело место: дата_____, время_____

В локальной конфигурации зафиксированы пользователи: _____

Дата_____ Подпись преподавателя_____

Лабораторная работа 2
ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ИНФОРМАЦИИ В ЛОКАЛЬНЫХ СЕТЯХ ПОД УПРАВЛЕНИЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS XP

Цель работы: ознакомить студентов с основными понятиями и возможностями операционной системы Windows XP по обеспечению конфиденциальности информации, выработать навыки администрирования в локальных сетях и работы в сети с учетом необходимости определять степень допуска других пользователей к объектам, находящимся в собственности.

Теоретические основы

Компания Microsoft, являющаяся производителем семейства операционных систем, базирующихся на NT технологии, предприняла специальные усилия в вопросах, касающихся безопасности информации. Все операционные системы, начиная с Windows NT 4.0, проходили сертификацию по критериям обеспечения информационной безопасности. В РФ операционная система Windows NT 4.0 сертифицирована по третьему классу на основании требований Гостехкомиссии РФ для автоматизированных систем учета и контроля ядерных материалов. Операционная система Windows XP сертифицирована на работу с конфиденциальной информацией по единым критериям информационной безопасности.

Защищенные операционные системы должны удовлетворять ряду требований. Так, они обязаны содержать следующие функции:

- идентификация и аутентификация пользователей при входе в систему;
- управление доступом к объектам операционной системы и его контроль;
- аудит событий и ряд других.

Рассмотрим, как эти функции реализованы в Windows XP.

Модель безопасности ОС, созданных по NT технологии. В Windows NT/2000/XP заложена объектная модель защиты. Объект представляет собой любой ресурс системы – файл, устройство, память, программа. Каждый объект содержит информацию о том, что разрешено делать тому или иному пользователю с этим объектом и

чего делать нельзя. Эта информация переносится вместе с объектом и наследуется. Основной целью является обеспечение контроля и управления доступом к объекту. Права доступа контролируются не только при прямом обращении пользователя к ресурсу, но и при опосредованном, через приложение, запущенное пользователем и работающее в его интересах.

Модель безопасности NT технологии состоит из следующих компонентов.

- Процесс входа в систему (LOGON). Он включает в себя два возможных процесса – начальный вход в систему и удаленный вход, т.е. получение доступа к серверным процессам и ресурсам.
- Распорядитель локальной безопасности – этот компонент гарантирует, что пользователь имеет разрешение на обращение к системе. Он генерирует маркеры пользователя, управляет политикой локальной безопасности и обеспечивает интерактивный сервис аутентификации пользователя.
- Диспетчер бюджета безопасности (SAM – Security Account Manager) поддерживает базу данных бюджетов пользователя и групп пользователей. SAM обеспечивает аутентификацию пользователя, которая используется распорядителем локальной безопасности.
- Монитор безопасности проверяет, имеет ли пользователь право доступа к объекту и отслеживает любое предпринимаемое действие. Проводит в жизнь проверку правильности доступа, обеспечивает сервис и привилегированному, и пользовательскому режимам, который обеспечивает гарантию доступа к объектам только пользователей и процессов, имеющих соответствующий допуск.

Управление счетами пользователей и группами пользователей. Каждый пользователь, зарегистрированный в системе Windows NT/2000/XP, имеет уникальное имя и пароль. Кроме того, система создает счет пользователя, содержащий права конкретного пользователя и его членство в различных группах.

Управление пользователями и группами пользователей осуществляется с помощью специальной утилиты «Управление компьютером»: Панель управления → Администрирование → Управление компьютером → Локальные пользователи и группы. Утилита предоставляет возможность создавать нового пользователя, новую

группу пользователей и модифицировать соответствующие параметры. Окно утилиты представлено на рис. 2.1.

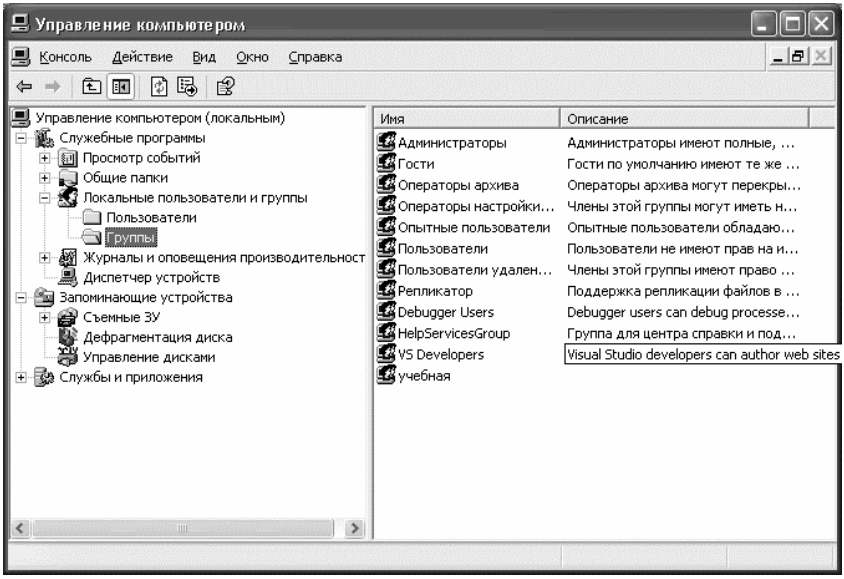

Рис. 2.1. Окно утилиты «Управление компьютером»

Создание нового счета пользователя. В окне «Управление компьютером» в левой части открывается функция управления пользователями: Служебные программы → Локальные пользователи и группы пользователей → Пользователи. Затем в главном меню окна вызываем опцию Действие → Новый пользователь. Открывается окно, представленное на рис. 2.2. Форма заполняется, отмечаются необходимые свойства учетной записи. Для завершения регистрации нового пользователя нажимается клавиша «Создать».

Группы локальных пользователей создаются аналогично. В соответствующем окне существует список пользователей данной группы и кнопки для добавления и удаления пользователей.

Обычная практика управления правами пользователей заключается в том, что права назначаются для различных групп, а затем пользователь включается в необходимые группы.

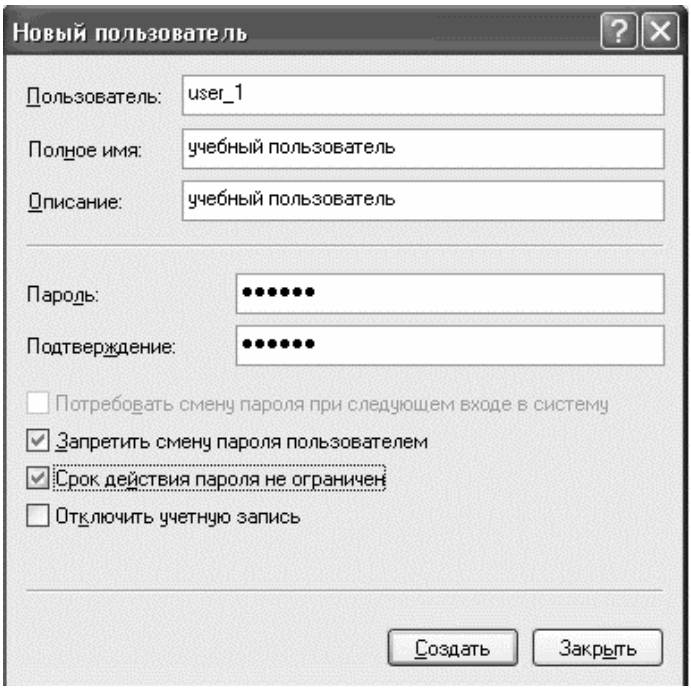

Рис. 2.2. Окно создания нового пользователя

Информация безопасности для объектов. В операционных системах семейства NT реализована дискреционная модель управления доступом к информации. Она заключается в том, что все объекты содержат списки контроля доступа, содержащие списки пользователей и групп пользователей, которым разрешен или запрещен доступ к объекту. Для каждого пользователя или группы из этих списков существует матрица доступа, которая перечисляет права пользователя. При обращении к объекту операционная система организует проверку возможности доступа.

В Windows NT/2000/XP каждый объект содержит дескриптор безопасности. Дескриптор безопасности включает в себя три части:

- Системный идентификатор владельца, который указывает пользователя или группу, являющуюся владельцем объекта. Владелец может изменять права доступа к объекту.

• Список контроля доступа (ACL – access control list). Он идентифицирует, каким пользователям и группам дозволен или запрещен доступ. ACL управляется владельцем ресурса.

• Системный ACL, управляющий списком сообщений контроля. Он управляется системным администратором.

Типы разрешений, которые предусмотрены у объекта, зависят от типа объекта. Некоторые права могут наследоваться. Так, в системе файлов NTFS права, созданные для директории, наследуются файлами.

Каждый ACL состоит из элементов управления доступом (ACE – access control entries). Они специфицируют разрешения на обращение к объекту для конкретного пользователя или группы. Между ACL и ACE существует принципиальное различие. Отсутствие первого делает объект абсолютно незащищенным. Отсутствие второго в присутствии первого, наоборот, отвергает всяческий доступ к объекту. Существует три типа ACE:

• разрешающий (allowed);

• запрещающий (denied);

• системной безопасности.

При проверке доступа сначала проверяется запрещающий список. Если пользователь или группа, к которой он принадлежит, включена в запрещающий список, доступ отвергается, и дальнейшая проверка не производится. Если пользователь не содержится в запрещающем списке, проверяется разрешающий. Если пользователь или какая либо из групп, в которые он входит, имеется в этих списках, то доступ разрешается. Затем определяются права пользователя по работе с объектом. Каждый ACE включает в себя маску доступа, которая определяет возможные действия для отдельного типа объекта. Например, файлы имеют следующие типы доступа:

• нет доступа;

• чтение;

• запись;

• добавление;

• полный контроль;

• выполнение (для исполняемых файлов).

Создание разрешений по файлам и папкам. Назначать права доступа к конкретным файлам и папкам имеют право только члены группы Administrators или Power Users и хозяева файлов. Чтобы выделить в совместное пользование папку, достаточно щелкнуть правой кнопкой мыши у выделенной папки и в выпавшем меню

выбрать опцию «Свойства». Открывается окошко, содержащее ее свойства. Открыв закладку «Доступ», можно разрешить доступ к данной папке по сети.

Назначение прав доступа. Эта функция выделяет пользователей и группы, которым разрешено работать с данным файлом. Для установления прав необходимо выделить папку или файл (или диск) и щелкнуть правой кнопкой мыши. В появившемся меню необходимо выбрать команду «Свойства». В нем надо выбрать «Безопасность» и нажать клавишу «Разрешения».

Ревизия событий безопасности. Под событием понимается попытка получить доступ к объекту с учетом типа доступа и результат этой попытки. Для контроля за событиями в системе безопасности в Windows NT/2000/XP предусмотрены журналы аудита, в которых фиксируются записи о событиях. Существует три типа журналов аудита (рис. 2.3):

- системный журнал – в него записываются события, инициируемые компонентами системы в случае особой ситуации (несрабатывания компонентов системы, включение источника бесперебойного питания).
- журнал безопасности записывает события, связанные с безопасностью (регистрация, использование ресурсов).
- журнал по прикладным программам регистрирует сообщения, поступающие от выполняемых приложений.

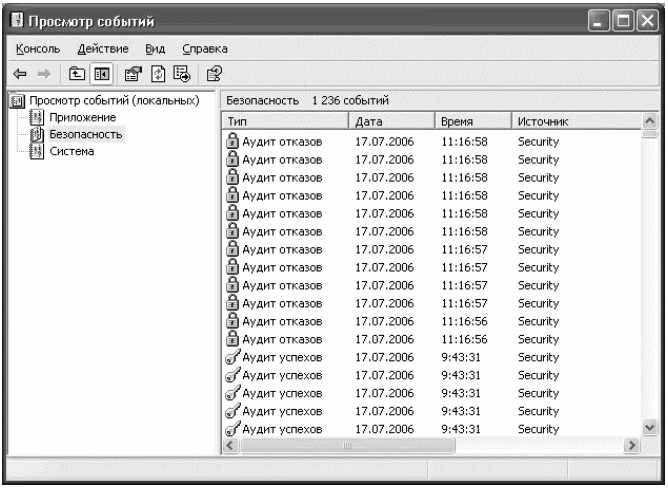

Рис. 2.3. Окно просмотра событий

Для того чтобы записи о событиях имели место, пользователь должен включить опции регистрации событий.

Управление журналами и их просмотр осуществляются специальной утилитой, доступ к которой можно получить через окно «Управление компьютером». Кроме того, доступ к управлению журналами открывается непосредственно из окна «Администрирование» на панели управления (см. рис. 2.1).

Выполнение работы

1. Администрирование локальной рабочей станции.

• Войти в систему рабочей станции локальным образом:

♦ В окне доступа выбрать имя рабочей станции.

♦ В окне имени пользователя ввести «Администратор».

♦ Получить административный пароль у преподавателя.

ВНИМАНИЕ! Работая с административными возможностями, легко нанести ущерб системе, поэтому просьба не отклоняться от предлагаемых далее действий.

• Изучить утилиту управления локальными учетными записями и группами.

♦ Открыть утилиту «Мой компьютер» → «Панель управления» → «Администрирование» → «Управление компьютером» → «Локальные группы и пользователи». Изучить возможность редактировать информацию о локальных группах. Открываем папку «Группы» и выбираем группу «Пользователи». Щелкнув два раза, открыть окно редактирования. Посмотреть ее возможности, определить, кто состоит в этой группе. Просмотреть остальные группы.

♦ Изучить возможность редактировать свойства пользователей. Для этого открыть папку Пользователи и выбрать пользователя «Гость». Открыть окно редактирования. Просмотреть информацию о пользователе. Просмотреть информацию об остальных пользователях.

♦ Создать новую группу пользователей. Для этого открыть папку «Группы». В меню выбрать Действие → Создать группу. Ввести имя группы – «Учебная», описание – «Группа для демонстрации административных средств». Нажать кнопку ОК. Посмотреть на отображение созданной группы.

♦ Аналогичным образом создать нового пользователя. Ввести имя пользователя «Ученик», полное имя «Иванов И.И.», описание «Демонстрационный пользователь». Пароль ввести произвольный, достаточно простой, чтобы не забыть на протяжении ближайших заданий.

♦ Включите нового пользователя в группу «Учебная», для этого открываем свойства пользователя и закладку «Членство в группах», добавить новую группу. Нажать кнопку ОК и просмотреть отображение нового пользователя. Продемонстрировать преподавателю нового пользователя.

♦ Осуществить вход в систему нового пользователя. Для этого закрыть все приложения и стартовать команду Пуск → Завершение сеанса. Входите в систему под именем «Ученик». Попробуйте создать и удалить пользователя. Объясните результат.

♦ Войти в систему с административным именем. Удалить пользователя «Ученик» и группу «Учебная».

• Ознакомиться с установкой локальной политики аудита. В папке «Панель управления» → «Администрирование» запускаем утилиту «Локальные параметры безопасности».

♦ Открыть папку «Локальные политики» → «Политика аудита».

♦ Установить параметры аудита «отказ» для события «Вход в систему».

• Изучить возможности аудита (работы с журналами).

♦ Открыть утилиту просмотра журналов Администрирование → Просмотр событий.

♦ Работа с системным журналом. Просмотреть детальную информацию о каком-либо событии. Для этого щелкнуть два раза на выбранном событии. Поменять вид представления информации – опция меню Вид → Сначала старые (новые).

♦ Просмотреть журналы по безопасности и приложениям.

♦ Сгенерировать записи в журнале безопасности. Для этого перезагрузите систему с тем же именем. Сделайте намеренно одну ошибку в пароле. Затем загрузитесь правильно. Войдите в «Просмотр событий» и просмотрите события в журнале безопасности. Найдите событие, зафиксировавшее отказ в доступе.

• Выйти из системы и осуществить вход со своими сетевыми параметрами пользователя.

2. Изучение прав собственника объекта. Создание разрешений на пользование объектом.

• Найдите файл, созданный вами на предыдущей работе, на диске D:\laboratory.

• Открыть свойства созданного файла. Выделить файл, щелкнуть правой кнопкой мыши и открыть опцию «Свойства».

• Открыть закладку «Безопасность». Нажать кнопку «Дополнительно» и отменить наследование свойств безопасности. Просмотреть владельца файла.

• Вернуться в окно «Безопасность». Установить разрешение для группы «Все» – «Нет доступа». Подтвердить решение. Закрыть окна свойств и попытаться открыть файл.

• Установить следующие разрешения для файла: Добавить группу «Magistr» (кнопка «Добавить», выбрать группу, «Добавить», «ОК»). Установить ей полный доступ. Попытаться открыть данный файл в Word.

• Удалить в разрешениях группу «Все». Группе «Magistr» установить разрешение «Только чтение». Открыть файл в Word. Добавить строку – «Изменение текста». Попытаться сохранить файл под старым именем.

• Добавить в разрешение конкретного пользователя – себя. Проверить действие следующих разрешений: «Magistr» – полный доступ, себе – нет доступа; группа – полный доступ, себе – только чтение; группа – только чтение, себе – полный доступ; группе – нет доступа – себе полный.

3. Действие прав доступа в сети.

Подключить общий диск сервера. Скопировать созданный файл на диск. Просмотреть его разрешения. Установить разрешения группе – только чтение. Определить, какое разрешение препятствует копированию файла.

4. Выключить систему, заполнить лист отчетности, получить подпись преподавателя.

Лист отчетности

Студент_____, группа_____

Имя компьютера в домене_____

Созданы новая группа и новый пользователь (отметка преподавателя)_____

Информация о последнем событии в системном журнале:

Дата_____ Время_____

Пользователь_____ Код_____

Источник_____ Тип_____

Информация в журнале безопасности, связанная с событием отказа.

Дата_____ Время_____

Пользователь_____ Код_____

Источник_____ Тип_____

Категория _____

Результаты действия различных разрешений.

Тип разрешения на группу	Тип разрешения пользователя	Результат
нет доступа	полный доступ	_____
полный доступ	нет доступа	_____
полный доступ	только чтение	_____
только чтение	полный доступ	_____

Дата_____ Подпись преподавателя_____

Лабораторная работа 3
АДМИНИСТРИРОВАНИЕ SQL SERVER. УПРАВЛЕНИЕ ОБЪЕКТАМИ СЕРВЕРА

Цель работы: ознакомить студентов с администрированием MS SQL Server 2000 и управлением объектами сервера, выработать навыки проектирования и создания реляционных баз данных, определения пользователей баз данных и создания объектов баз данных с помощью SQL Enterprise Manager.

Теоретические основы

Общие характеристики MS SQL Server. MS SQL Server 2000 – мощная система управления базами данных (СУБД). Сервер спроектирован специально для распределенных сетевых приложений, реализующих архитектуру клиент/сервер.

Эксплутационные качества MS SQL Server:

- мощность;
- масштабируемость – использует многопроцессорные системы, оптимизируется для достижения максимальной производительности при различном числе пользователей, поддерживает различные форматы баз данных;
- управляемость – поддерживается интерфейсом администратора, использование запросов, сохраняемых процедур;
- надежность – обладает мощной системой безопасности как для предотвращения несанкционированного доступа, так и для обеспечения целостности и сохранности данных.

Основные понятия MS SQL Server:

- базы данных;
- журналы транзакций;
- таблицы;
- представления;
- правила;
- пользователи;
- роли сервера и базы данных;
- хранимые процедуры;

• триггеры.

База данных – это набор данных, организованный в виде таблиц и других объектов. Базы данных располагаются на файлах. База данных может размещаться на нескольких файлах, которые размещаются на различных физических устройствах. Каждой базе данных соответствует журнал транзакций. Транзакция фиксируется в журнале транзакций и только потом записывается в базу данных.

Таблица – набор информации об отдельном объекте. В таблицах SQL Server невозможно удалять столбцы или изменять тип данных.

Представления – это виртуальная таблица. Данные в представлении не сохраняются, они создаются при открытии представления.

Правила – это отдельные объекты, которые связаны с множеством столбцов в базе данных.

Пользователь SQL Server – это пользователь базы данных, но не обязательно уникальный пользователь сети. В случае установления интегрированной системы безопасности сервера, пользователи сети становятся пользователями базы данных.

Роль – множество пользователей SQL Server. Будучи включенным в ту или иную роль, пользователь приобретает определенные права.

Хранимые процедуры – запрос SQL, однажды созданный, проверенный, оптимизированный и сохраненный в базе данных. Сохраненные процедуры могут иметь аргументы ввода и аргументы вывода.

Триггеры можно рассматривать как расширенные сохраняемые процедуры, выполняемые автоматически. Триггеры проверяют допустимость ряда действий – введения данных, обновления таблицы или уничтожения записи.

Утилита управления сервером – SQL Enterprise Manager. Данная утилита содержит полный набор графических инструментов для администрирования сервером и управлением его объектами. На рис. 3.1 приведено открытое окно Enterprise Manager.

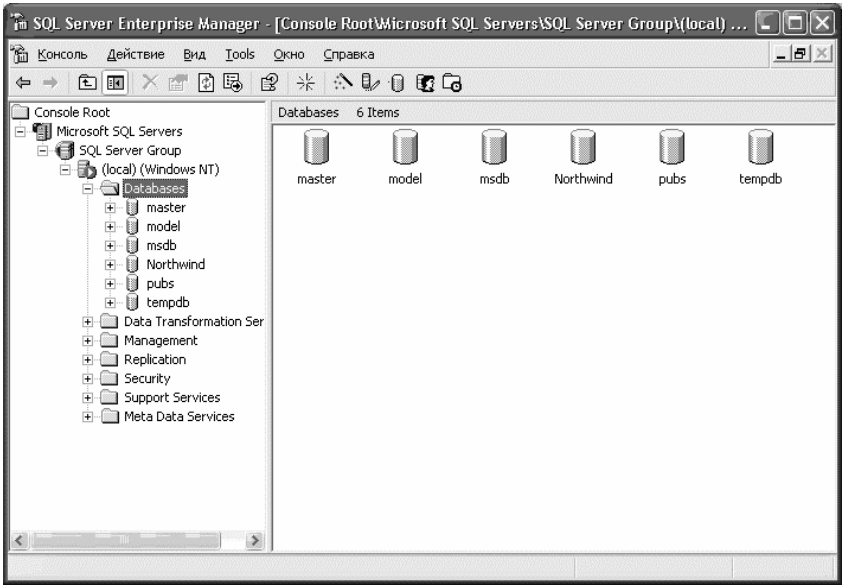

Рис. 3.1. Окно SQL Enterprise Manager

Слева открыто дерево, содержащее папки с объектами сервера. Раскрывая их, получаем доступ к тем или иным объектам для просмотра и управления ими.

Управление базами данных. База данных является базовым элементом SQL Server и контейнером, в котором располагаются объекты и данные. При создании базы данных прежде всего необходимо осуществить планирование размещения базы данных на физических и логических дисках. Как уже говорилось ранее, база данных может быть размещена на одном или нескольких файлах, которые в свою очередь могут быть размещены на различных дисках. Управление этим размещением способно повысить производительность приложений и увеличить степень сохранности данных. Особое внимание необходимо обратить на размещение журнала транзакций. Он должен быть размещен на отдельном носителе для того, чтобы вероятность его порчи одновременно с базой данных была минимальна. Кроме того, в случае напряженной работы с данными расположение журнала транзакций на отдельном физическом устройстве ускоряет производительность.

При создании базы данных администратор может выбирать сопоставление, которое будет использовать база данных. Сопоставление определяет правила работы с именами объектов (например, чувствительность/нечувствительность к регистру). По умолчанию используется сопоставление, установленное для сервера. Выбор сопоставления позволяет использовать национальные алфавиты.

Рассмотрим создание базы данных с использованием Enterprise Manager. Для создания новой базы данныхнужно открыть папку Databases и на панели инструментов нажать кнопку – New Database. Открывается окно, главная страница которого показана на рис. 3.2.

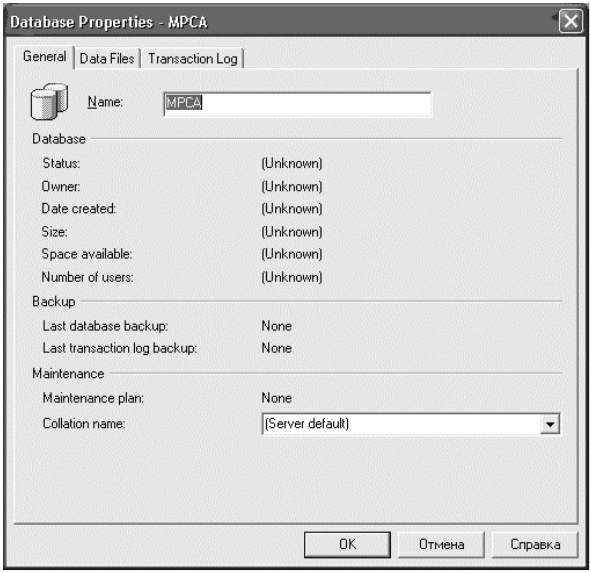

Рис. 3.2. Главная страница окна задания базы данных

Задается имя базы данных и выбирается сопоставление из выпадающего списка. Поясним на примере смысл имени сопоставления. На рисунке в списке выбрано сопоставление Cyrillic_General_CI_AS. Первая часть имени задает национальный набор символов. Второе имя задает особенности национальных кодировок. Для кириллического шрифта имеется только одно значение – General. Далее следуют двухбуквенные сокращения, описывающие свойства сопоставления в отношении сравнения символов. Первая

буква означает свойство символов, а вторая – поведение сервера в отношении этого свойства. Первая буква может принимать значение:

- C – чувствительность к регистру;
- A – чувствительность к акценту;
- W – чувствительность к ширине символа.

Вторая буква может принимать только два значения S – чувствителен, I – не чувствителен.

Открыв закладку Data Files, определяем физическое размещение базы данных на дисках. Окно представлено на рис. 3.3. Задаем логическое имя файлов, размещение на дисках, первоначальный размер файла и файловую группу. При выборе группы файлов открывается список групп. По умолчанию он содержит одну группу PRIMARY. Пользователь может ввести название новой группы файлов. Здесь же располагается кнопка Delete для удаления файла. Внизу окна размещены значения параметров, определяющих автоматическое увеличение размеров файла при его переполнении. Указывается размер приращения файла и максимально возможный его размер.

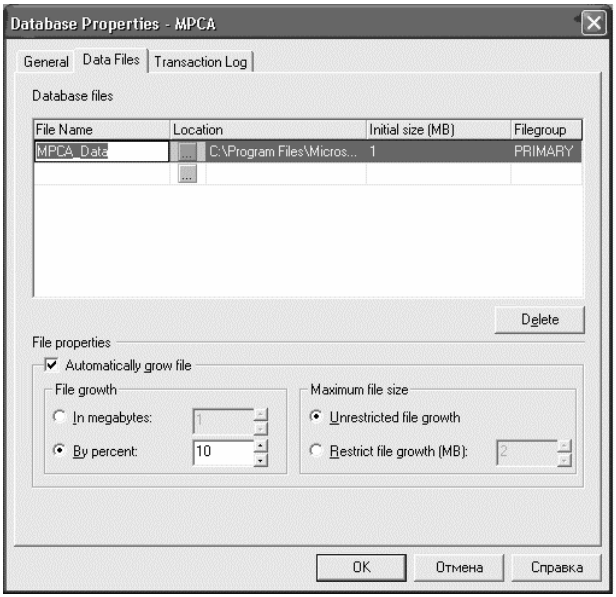

Рис. 3.3 Определение размещения базы данных на файлах

Третья закладка задает параметры журнала транзакций. Параметры журнала определяются так же, как и файлы базы данных.

Затем добавляем пользователей базы данных и определяем их роли. Для этого открываем папку Users и вызываем утилиту New User. Открывается окно (рис. 3.4), в котором из выпадающего списка выбираем пользователя сервера и определяем его имя и роль в качестве пользователя базы данных. При необходимости в этом же окне можно определить разрешения пользователя на работу с объектами базы данных.

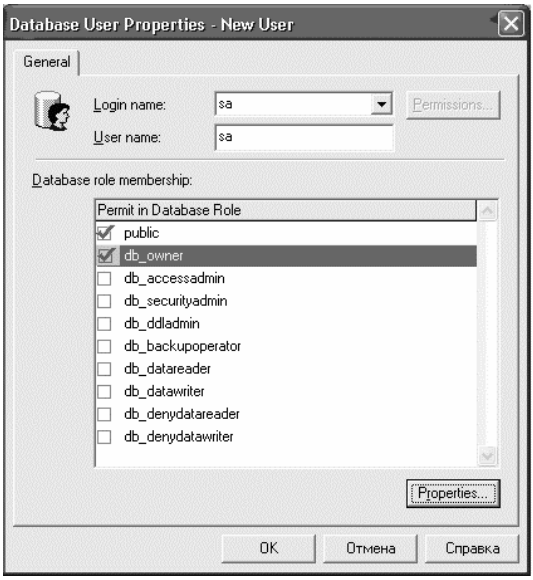

Рис. 3.4. Окно добавления пользователя базы данных

Создание таблиц. В SQL Enterprise Manager существует специальный инструмент для создания таблиц. При открытой папке Tables открывается окно определения таблицы (рис. 3.5).

Ограничения на информацию в полях таблицы:

• Типы данных. Каждому столбцу таблицы в SQL Server задается определенный тип данных.

• NOT NULL. Можно потребовать, чтобы столбец каждой новой строки (записи) всегда содержал какое-либо значение. Таким образом, SQL Server не позволяет опускать необходимую информацию.

• PRIMARY KEY (первичный ключ). Значение первичного ключа в таблице должно быть уникальным. Первичный ключ в SQL Server может создаваться не на одном, а на нескольких полях, тогда уникальной должна быть их комбинация.

• UNIQUE. SQL Server поддерживает возможность требования уникальности поля в дополнение к первичному ключу.

• FOREIN KEY (внешний ключ). Когда SQL Server объявляет поле внешним ключом, то определяется, в какой таблице этот ключ является первичным. Его тип берется из родительской таблицы. Так устанавливается ссылочная целостность между таблицами.

• DEFAULT. SQL Server может определять значение поля по умолчанию. Часто это свойство используется, чтобы предотвратить NULL значения в полях, на которые установлено ограничение NOT NULL.

• CHECK. Для каждого столбца таблицы можно определить выражение CHEK, которое должно быть истинным для любого вводимого значения. Результат этого выражения должен быть булевским – истинно или ложно. Эта возможность реализует пользовательскую целостность данных.

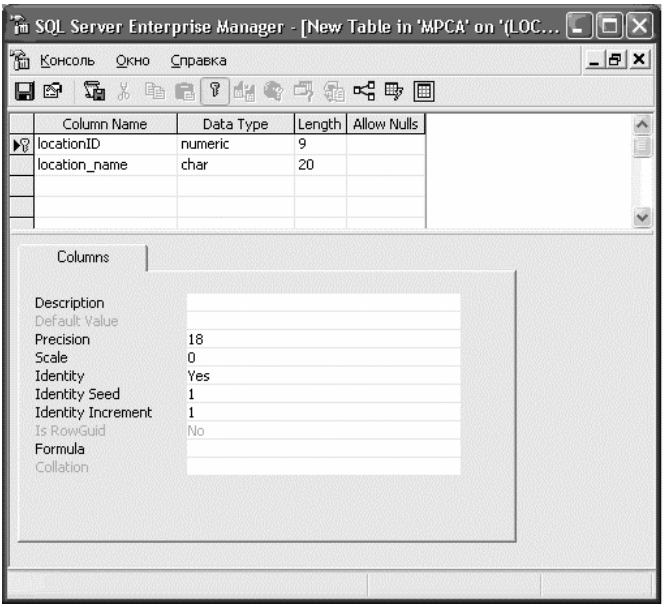

Рис. 3.5. Окно создания таблицы

Для установления ограничений вызывается специальное окно (рис. 3.6), отдельные закладки которого позволяют создать те или иные ограничения.

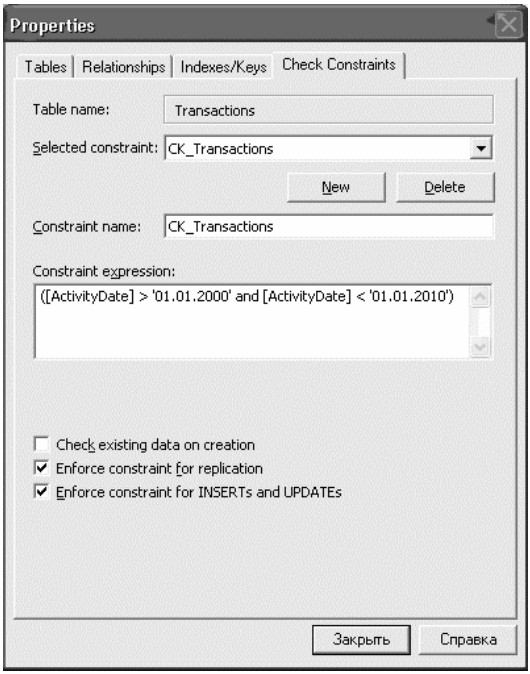

Рис. 3.6. Окно определения ограничений

Создание диаграммы отношения сущностей. Диаграмма отношения сущностей (ERD) позволяет наглядно представить базу данных со всеми существующими таблицами и их связями.

Для создания диаграммы необходимо в базе данных открыть папку Diagrams и в контекстном меню вызвать опцию New diagram. Открывается окно диаграммы и запускается помощник для создания диаграммы. Из списка таблиц выбираем таблицы, которые мы хотим видеть в диаграмме. При нажатии клавиши Finish появляется готовая диаграмма (рис. 3.7). Из рисунка видно, что меню инструментов окна позволяет выполнять все функции, которые использовались при создании таблиц.

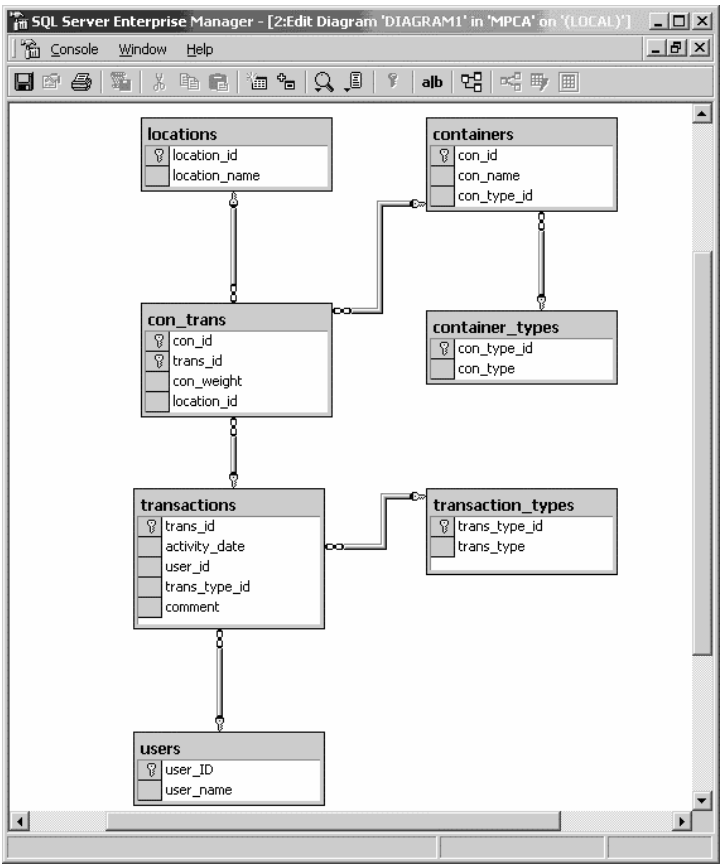

Рис. 3.7. Диаграмма отношения сущностей

Создание представления. Для автоматизации создания представлений SQL Server имеет специальную утилиту. Открываем папку Views. Щелкаем правой кнопкой и в выпадающем меню стартуем утилиту New View. Открывается окно создания представления. Окно разделено на четыре части. В первой части располагаются таблицы, на базе которых создается представление, во второй части располагается список полей представления с указанием ряда их свойств. В третьей части выводится запрос на языке SQL для создания представления. Если стартовать команду Run в выпадаю-

щем меню, то в четвертой части выводится содержимое представления. На рис. 3.8 представлено окно создания запросов.

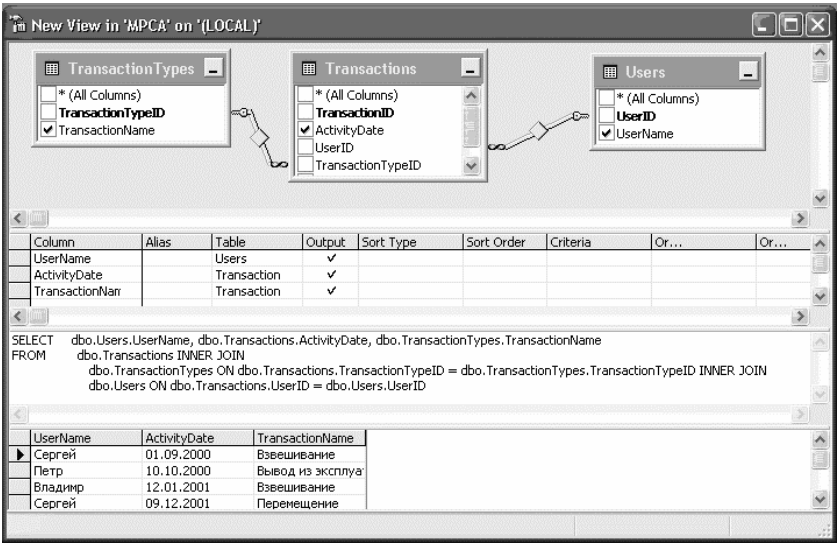

Рис. 3.8. Окно автоматизации создания представления

Выполнение работы

1. Рассмотреть проект базы данных, выданный на лекции (см. приложение). Определить структуру таблиц, ключевые поля, отношения между таблицами, типы полей данных. Разобрать правила и ограничения в базе данных.

2. Получить от преподавателя административный логин и пароль, позволяющий пользователю выступать в роли администратора сервера. Стартовать клиентское приложение SQL Server на рабочей станции и запустить на выполнение утилиту SQL Enterprise manager.

Зарегистрировать сервер. Для регистрации сервера выбрать из выпадающего списка имя сервера MASTER и ввести подтверждение – нажать клавишу «ОК».

3. Осуществить базовые настройки сервера.

Для этого: на дереве объектов выделить сервер, щелкнуть правой клавишей мышки и в выпадающем меню вызвать опцию Properties.

Выбрать закладку Security. Выбрать тип аутентификации пользователя – Windows Only и уровень аудита системы – регистрировать события, связанные с отрицанием допуска (Failure). Завершить работу с окном свойств, нажав клавишу «OK».

На дереве объектов сервера открыть папку Databases. Просмотреть список баз данных, выделить системные базы данных, записать системные базы в лист отчетности. Отключить просмотр системных объектов сервера. Для этого на дереве объектов выделить сервер, щелкнуть правой клавишей мышки и в выпадающем меню вызвать опцию Edit SQL Server Registration properties. На открывшемся окне регистрационных свойств сервера снимите отметку с позиции Show system databases and system objects. Завершить работу с окном свойств, нажав клавишу «OK». Просмотрите, как изменился список баз данных.

4. Добавить свой собственный логин операционной системы в качестве нового пользователя SQL Server. Для этого: на дереве объектов открыть папку Security и вложенную в нее папку Logins. Щелкнув правой кнопкой мыши, открыть в выпадающем меню опцию New login. На открывшемся окне утилиты осуществить следующие действия:

- на открытой закладке General выбрать домен MPCA и в выпадающем списке выбрать свой логин, зарегистрированный в операционной системе;
- на вкладке Server roles отметить серверную роль Database creators. Это позволит Вам под собственным паролем создавать базы данных;
- на вкладке Databases Access отметить базы данных Pubs и Master и в таблице разрешений отметить роль базы данных db_owner. Это необходимо проделать, чтобы у вас была возможность управлять создаваемой базой данных.

Закрыть сервер и отлогиниться от системы.

5. Завершить работу под административным логином. Войти в систему со своим доменным именем. Стартовать утилиту SQL Enterprise manager.

6. Создать базу данных и ее журнал транзакций.

Для этого открыть папку Databases и стартовать опцию меню Действие → New Database. В открывающемся при этом окне выполняются следующие операции:

- В текстовом окне на закладке General задать имя базы данных. Имя образуется из букв db и номера рабочей станции. Например – db3 для компьютера Wks3.
- На закладке Data files выбрать файл, в котором будет размещена база данных. Для этого в выпадающем списке выбрать папку «DB_magistr» на открытом для вас диске сервера. Определить имя файла по умолчанию. Затем определить начальный размер базы данных – 5 Mb и определить шаг увеличения базы данных в 1 Mb .
- На закладке Transaction Logs определить аналогичные параметры для журнала транзакций: размещение в папке DB_log_magistr на том же диске, начальный размер 1 Mb, увеличение – 10%.

Нажав кнопку «OK», создать базу данных.

7. Добавить пользователей базы данных. Для этого в дереве объектов открыть появившуюся папку с объектами вашей, только что созданной базы данных. Выбрать Users. Стартовать утилиту добавления пользователей базы данных: Действие → New Database User. В выпадающем списке выбрать логины пользователей, которых вы хотите добавить в вашу базу данных. Нажать кнопку «OK».

8. Создать новую пользовательскую роль в новой базе данных и включить в нее добавленных пользователей базы данных. Для этого открыть папку Roles и стартовать утилиту Действие → New Database role. В открывшемся окне задать имя группы – Developers и добавить в нее всех подключенных пользователей. Для этого нажать кнопку Add и добавить пользователей из списка.

9. Задание прав пользователям вашей базы данных. Для этого выделить папку вашей базы данных и стартовать утилиту Действие → Свойства. В открывшемся окне выбрать закладку Permissions. Отметить все возможные права для роли Developers.

10. Создание таблицы базы данных, приведенных в приложении.

Для этого открыть папку Tables. Стартовать утилиту создания таблицы Действие → New Table. В открывшемся окне последовательно определить:

• В таблице в верхней части окна последовательно задать имя столбца, его тип, для строковых типов определить длину, разрешить или запретить NULL значение.

• Для столбца или столбцов, которые будут выступать в качестве первичного ключа, установить это свойство. Для этого выделить столбец и щелкнуть правой кнопкой мышки. В выпадающем меню щелкнуть на опции Set primary key. В нижней части окна в таблице свойств столбца установить свойство identity в значение Yes.

• Для столбцов, у которых определено значение по умолчанию, установить это свойство – просто набрать в строке свойства соответствующее значение.

• Сохранить таблицу. Щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Save. Установить соответствующее имя таблицы.

• Для тех таблиц, у которых установлены ограничения по проверке значений (таблицы 6 и 7), открыть окно установления ограничений. Щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Check constraints. Нажать кнопку New и в открывшемся окне записать логическое выражение для проверки ограничения. Например, weight > 30 and weight < 50.

• Закрыть окно создания таблицы.

11. Установить внешние ключевые поля.

Определить внешние ключи таблиц 5, 6 и 7 для этого:

• Выделить нужную таблицу, щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Design table. Открывается окно модификации таблицы.

• Щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Relationship. Открывается окно установления связей.

• Нажать кнопку New и определить внешний ключ таблицы, родительскую таблицу и ее первичный ключ.

• Устанвить проверки существования связанных данных при вводе данных в таблицы.

Повторить действия для всех необходимых таблиц.

12. Создать Диаграмму отношений сущностей. Для этого открыть папку Diagrams и запустить помощник создания диаграмм Действие → New Database Diagram. Выполнить последовательно все требуемые операции. Добавить в список все таблицы и нажать кнопку Готово. Подписать диаграмму с указанием фамилии, груп-

пы и даты. Для этого на панели инструментов нажать кнопку создания текста и в открытом окне набрать требуемый текст. После создания диаграммы распечатать ее.

13. Занести информацию в таблицы. Содержимое таблиц приведено в приложении.

Для этого выполнить следующие действия:

- Выбрать требуемую таблицу. Открыть ее на просмотр, щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Open table → Return all rows.
- Ввести новую информацию.

Обратите внимание на то, что ввод информации в таблицы, на которые существуют ссылки из других таблиц, должен предшествовать вводу данных в эти таблицы.

14. Создать представление, которое выводит информацию из трех таблиц:

UserName из таблицы Users, ActivityDate из таблицы Transactions и TransactionName из таблицы TransactionTypes.

Для выполнения открыть папку View и стартовать опцию Действие → New View. Выполнить следующие действия.

Стартовать опцию выпадающего меню – Add Table. Из выпадающего списка выбрать три нужные таблицы. Они появляются в верхнем окне с указанием связей.

Отметить в таблицах поля, которые нужно просматривать в представлении. Они появляются в таблице, расположенной посередине окна. Одновременно видно, как формируется запрос на языке SQL.

Стартовать опцию – Выполнить. Щелкнуть правой кнопкой мышки и в выпадающем меню выбрать опцию Run. В нижней таблице просмотреть результат.

Сохранить представление с именем Trans_Info.

Переписать SQL запрос, созданный автоматически.

Закрыть окно создания представления.

Получить подпись преподавателя, закрыть сервер, выключить компьютер.

ПРИЛОЖЕНИЕ

Структура таблиц учебной базы данных

Таблица 1

«Locations»

LocationID	LocationName
1	Здание_100
2	Хранилище
3	Перегрузочный док

LocationID – Первичный ключ, свойство «identity»;
LocationName – NOT NULL, значение по умолчанию – «Здание_100».

Таблица 2

«ContainerTypes»

ContainerTypeID	ContainerType
1	Металлическая туба
2	Стеклянная банка
3	Деревянный ящик

ContainerTypeID – Первичный ключ, свойство «identity»;
ContainerType – NOT NULL, значение по умолчанию – «Деревянный ящик».

Таблица 3

«Users»

UserID	UserName
1	Владимир
2	Петр
3	Сергей

UserID – Первичный ключ, свойство «identity»;
UserName – NOT NULL.

Таблица 4

«TransactionTypes»

TransactionTypeID	TransactionName
1	Перемещение
2	Взвешивание
3	Передача в эксплуатацию
4	Вывод из эксплуатации

TransactionTypeID – Первичный ключ, свойство «identity»;
TransactionName – NOT NULL.

Таблица 5

«Containers»

ContainerID	ContainerName	ContainerTypeID
1	ДЛ-2301	2
2	НТ-101	1
3	БН-1999	3

ContainerID – Первичный ключ, свойство «identity»;
ContainerName – NOT NULL;
ContainerTypeID – Внешний ключ, ссылка на табл. 2.

Таблица 6

«Transactions»

TransactionID	ActivityDate	UserID	TransactionTypeID	Comment
1	01.09.98	3	2	
2	10.10.98	1	4	
3	10.11.98	1	1	
4	09.12.98	3	3	
5	12.01.99	2	2	

TransactionID – Первичный ключ, свойство «identity»;
ActivityDate – NOT NULL, тип – дата, ограничение: от 01.01.2000 до 01.01.2010;
UserID – Внешний ключ, ссылка на таблицу 3;
TransactionTypeID – Внешний ключ, ссылка на табл. 4;
Comment – возможны NULL.

Таблица 7

«Container_Transaction»

ContainerID	TransactionID	ContainerWeight	LocationID
2	1	100	2
2	3	100	1
3	2	35	4
1	5	440	2
3	1	250	2

ContainerID + TransactionID – Композитный первичный ключ;
ContainerID – Внешний ключ, ссылка на табл. 5;
TransactionID – Внешний ключ, ссылка на табл. 6;
ContainerWeight – NOT NULL, тип «real», ограничение: от 30 до 400;
LocationID – Внешний ключ, ссылка на табл. 1.

Лист отчетности

Студент_____, группа_____
Имя сервера

Список системных баз данных сервера.

SQL запрос на создание представления TransactionView.

Приложить распечатанную диаграмму отношений сущностей базы данных.

Число _____ Подпись преподавателя_____

Лабораторная работа 4
ИЗУЧЕНИЕ ЯЗЫКА ЗАПРОСОВ TRANSACTION SQL

Цель работы: ознакомить студентов с основными возможностями структурированного языка запросов SQL, который обеспечивает манипулирование данными в существующих базах данных. Изучается использование конструкций запросов при создании объектов базы данных: представления, сохраняемые процедуры, правила. Для работы используется учебная база данных SQL Server – Pubs и созданные на предыдущем занятии учебные базы данных.

Теоретические основы

Редактор запросов SQL. SQL Query Analyzer – утилита SQL Enterprise Manager, позволяющая связать пользователя с конкретной базой данных, сформировать запрос на языке SQL, запустить его на выполнение и просмотреть результаты этого запроса (рис. 4.1). С помощью этой утилиты можно осуществлять все операции, включая создание и сопровождение устройств, баз данных и их объектов.

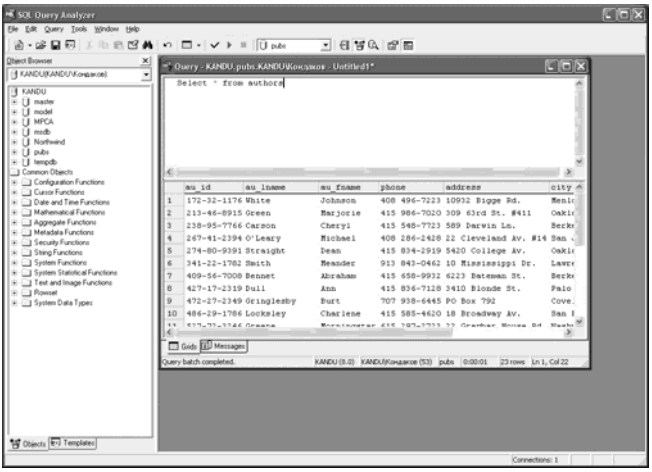

Рис. 4.1. Окно утилиты осуществления запросов

В верхней половине окна располагается область написания запроса. Внизу – область вывода результата. В случае ошибки в этой области появляется код ошибки и его описание. База данных, с которой работает пользователь, выбирается в текстовом окне на панели инструментов. Запрос посылается на выполнение щелчком мышки на зеленом треугольнике на панели инструментов. С помощью соответствующих кнопок пользователь может сохранить запрос в виде скрипта в текстовый файл операционной системы.

Основные команды Transaction-SQL:

• Команды выборки SELECT. Позволяют получать информацию из базы данных, группировать ее, получать некоторые обработанные данные.

• Команды вставки INSERT. Позволяют вводить в таблицы новые записи (строки).

• Команды обновления UPDATE. Позволяют изменять информацию в существующих записях.

• Команды удаления DELETE. Позволяют удалять записи.

Синтаксис **команды выборки SELECT** имеет вид:

```
SELECT [ALL | DISTINCT] select_list
[FROM {table_name | view_name} [(optimizer_hints)]
     [[,{table_name2 | view_name2} [(optimizer_hints)]
     [...,{table_name16 | view_name16} [(optimizer_hints)]]]
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
```

Здесь:

SELECT – ключевое слово, которое сообщает базе данных, что команда является запросом;

select_list – специфицирует список столбцов (полей) для выбора. Может принимать следующие значения:

• вместо списка можно использовать символ (*). При этом будут выводиться все поля таблиц, указанных в предложении FROM;

• список имен столбцов таблицы, которые должны быть представлены в результате выполнения запроса. Столбцы, которые не

представлены в этом списке, не включаются в состав выходных данных;

• можно выводить имя поля вместе с заголовком, который должен быть выведен вместо имени поля. Например:

“Фамилия оператора”=last_name_operator

здесь last_name_operator – название столбца в базе данных, а “Фамилия оператора” – заголовок столбца, который будет выведен в результате выполнения запроса.

FROM – ключевое слово, за которым находится список таблиц или представлений, которые являются источником информации для запроса.

table_name | view_name – список таблиц и представлений, которые должны выводить информацию. В именах может быть специфицирована база данных и пользователи. Например: database.owner.table_name. Специфицировано может быть до шестнадцати различных объектов (таблиц и представлений).

ALL | DISTINCT – специфицирует возможность исключения дублирующих значений. Если используется оператор DISTINCT, то не будут выводиться дублирующие значения. Например, в таблице транзакций вы хотите получить список операторов, осуществляющих транзакцию. Тогда без оператора DISTINCT вы получите список операторов по всем транзакциям, т.е. фамилии могут повторяться. Оператор DISTINCT исключит повторение. ALL – противоположность DISTINCT, используется по умолчанию.

optimizer_hints – специальный индикатор, сообщающий специфический метод, который будет использоваться для выборки в данной таблице. Например, указывается индексное поле. Подробно эту возможность разбирать не будем.

WHERE – это предложение позволяет определить *предикат* – условие, которое принимает значение либо истинно, либо ложно. Команда извлекает из таблицы только те значения, для которых предикат принимает значение «истина».

Возможности создания сложных предикатов. Поскольку предикат в предложении WHERE принимает значение «истина» или «ложь», то для их формирования можно использовать операторы булевой алгебры. SQL может использовать операторы сравнения и основные булевы операторы: AND, OR, NOT.

Кроме булевых операторов для формирования предикатов SQL может использовать специальные операторы:

IN – полностью определяет множество, к которому данное значение может принадлежать.

BETWEEN – этот оператор задает границы, в которые должно попадать значение поля, чтобы предикат был истинным. Оператор чувствителен к порядку задания границ – первое значение должно быть меньше, чем второе.

LIKE – оператор применим только к символьным полям и используется для поиска подстрок. Он выдает значение «истина» в том случае, когда заданная подстрока входит в состав строки. С этой целью оператор использует шаблоны. Шаблоны имеют следующие значения:

- (_) – символ подчеркивания. Заменяет один любой символ;
- (%) – заменяет последовательность символов произвольной, в том числе и нулевой, длины.

IS NULL – оператор, который фиксирует отсутствие значений. Если данные в поле отсутствуют, они не могут участвовать в операторах сравнения – невозможно получить результат. В этих случаях используется оператор IS NULL.

Использование функций агрегирования в запросах SQL. Поля в таблицах реляционных баз данных должны быть независимыми. Тем не менее, часто требуется информация, которую можно получить на основе этих полей, используя некоторые процедуры – суммирования, усреднения и т.п. Это делается с помощью функций агрегирования и суммирования:

- COUNT – определяет количество строк или полей, выбранных посредством запроса и не являющимися NULL значениями;
- SUM – вычисляет арифметическую сумму всех выбранных значений данного поля;
- AVG – вычисляет среднее значение всех выбранных значений данного поля;

MAX – вычисляет наибольшее из всех выбранных значений данного поля;

- MIN – вычисляет наименьшее из всех выбранных значений данного поля.

Предложение GROUP BY позволяет определять подмножество значений одного поля в терминах другого поля и применять функ-

ции агрегирования к полученному подмножеству. Пример: надо получить максимальный по цене заказ, сделанный каждым продавцом:

```
SELECT s_name, MAX(amount)
    FROM Orders
    GROUP BY s_name
```

Функции агрегирования нельзя использовать в предложении WHERE, предикаты относятся к единственной строке, а функции агрегирования оперируют с группами. Чтобы вводить ограничения на группы данных, используют предложение HAVING. В приведенном примере выведем только те максимальные заказы, которые превышают определенные значения:

```
SELECT s_name, MAX(amount)
    FROM Orders
    GROUP BY s_name
    HAVING MAX(amount)>3000.00
```

Упорядочение результатов запросов. В реляционных базах данных таблицы являются неупорядоченными множествами, и полученные данные на их основе необязательно представляются в какой-либо определенной последовательности. В SQL применяется предложение ORDER BY, позволяющее внести определенный порядок в выходные данные. Данные упорядочиваются в соответствии со значениями одного или нескольких выбранных столбцов. Множество столбцов упорядочиваются один внутри другого. Можно задавать возрастающую (ASC) или убывающую (DESC) последовательность сортировки. Пример: выведем список пользователей по алфавиту в возрастающей последовательности.

```
SELECT last_name, first_name FROM Users
  ORDERED BY last_name DESC
```

Использование дополнительных возможностей предложения SELECT. В запросе может использоваться несколько таблиц, учитывая связи между ними. Операции такого рода называются соединением. Обычно эту операцию проделывают по ключевым полям. Например, нам надо получить список транзакций и фамилию поль-

зователя, выполнившего ее. Информация помещена в двух таблицах. В одной содержится внешний ключ – идентификатор пользователя, в другой этот ключ является первичным. Тогда имеем:

```
SELECT Transaction.Trans_name, Transaction.Trans_date,
Users.Users_name
  FROM Transaction, Users
  WHERE Transaction.user_ID=Users.user_ID
```

SQL позволяет вкладывать запросы друг в друга. При этом внутренний запрос генерирует значение предиката, которое затем тестируется на предмет истинности. Пример: надо вывести значения транзакций, осуществленных пользователем “Борис”. Но мы можем не знать его пользовательского ID, по которому связываются таблицы. Тогда можно сформировать следующий запрос:

```
SELECT * FROM Transaction
    WHERE user_ID=
          (SELECT user_ID FROM user_name=“Борис ”)
```

Команда вставки INSERT. Позволяет добавлять новые записи. Синтаксис команды следующий:

```
INSERT [INTO]
      {table_name | view_name} [(column_list)]
      {DEFAULT VALUE | value_list | select_statement}
```

Здесь следующие параметры:

INTO – ключевое слово, используется в стандарте ANSI, может опускаться в версии MS SQL Server;

table_name | view_name – имя таблицы (или представления) в которую будет добавляться строка;

column_list – список столбцов, для которых будет вводиться информация. Если этот список не определен, то вводиться будут все столбцы;

DEFAULT VALUE – когда используется это значение, то в новой строке вводятся значения по умолчанию;

value_list – определяет список вводимых значений. Имеет вид:

VALUES (list_of_constant).

Команды обновления UPDATE. Для изменения данных в существующих строках используется команда следующего синтаксиса:

UPDATE [table_name | view_name]
SET [{table_name | view_name}]
{column_list | variable_list | variable_and_column_list}..
{column_list | variable_list | variable_and_column_list}
WHERE clause

Значения соответствуют тому, что было выше.

Команды удаления DELETE. Синтаксис уничтожения записи имеет вид:

DELETE [FROM] {table_name | view_name}
[WHERE clause]

Создание хранимых процедур. Синтаксис процедуры создания хранимой процедуры следующий:

CREATE PROCEDURE [owner.] procedure_name[;number]
[(parameter1,..[parameter255])]
[{FOR REPLICATION} | {WITH RECOMPILE}
[{[WITH] | [,]} ENCRIPTION]]
AS sql_statement

Здесь: procedure_name – имя создаваемой хранимой процедуры, не должно превышать 20 символов;

number – номер, если создаются групповые хранимые процедуры с одним именем. Дается возможность уничтожить их сразу одной командой по имени. При вызове номер отделяется от имени точкой с запятой;

parameter – хранимые процедуры могут иметь до 255 параметров. Синтаксис задания параметра следующий:

@parameter_name datatype [=default] [OUTPUT]

имя параметра должно удовлетворять правилам идентификатора SQL, специфицируется тип данных параметра. Пользователь может указать значение параметра по умолчанию. Слово OUTPUT показывает, что данный параметр возвращаемый.

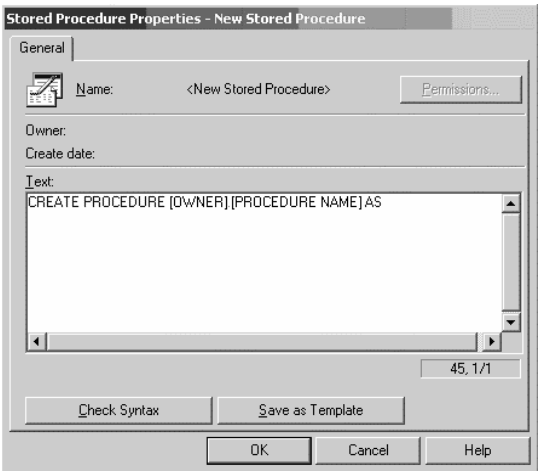

Рис. 4.2. Утилита создания хранимой процедуры

И, наконец, sql_statement – специфицирует действие, которое будет выполнять хранимая процедура.

Выполнение хранимых процедур осуществляется с помощью команды EXECUTE.

SQL Enterprise Manager предусматривает специальную утилиту для помощи при написании хранимых процедур (рис. 4.2).

Данная утилита имеет функцию проверки синтаксиса создаваемой процедуры.

Создание правил. Правила связаны со столбцами таблиц или с созданными пользователем типами данных. Каждый столбец или тип может иметь только одно правило, связанное с ним. Синтаксис команды:

CREATE RULE [*owner*.]*rule_name*
AS *condition_expression*

Здесь rule_name – имя правила, condition_expression – выражение, определяющее правило. Могут использоваться любые арифметические операторы и операторы сравнения, а также выражения **IN, LIKE, BETWEEN.** Выражение включает одну переменную. Она является локальной и должна иметь перед названием символ @. Правила создаются в текущей базе данных. После создания необходимо связать правило со столбцом с помощью системной сохраняемой процедурой **sp_bindrule**. Она вызывается командой:

sp_bindrule ***rule_name, objname***

где rule_name – имя правила, objname – имя объекта, с которым правило связывается.

На рис. 4.3 представлено окно утилиты для создания и связывания правила. В текстовом окне пишется выражение, которое может принимать значение Истинно/Ложно. Нажатием кнопки открывается диалог связывания правила с конкретным столбцом конкретной таблицы.

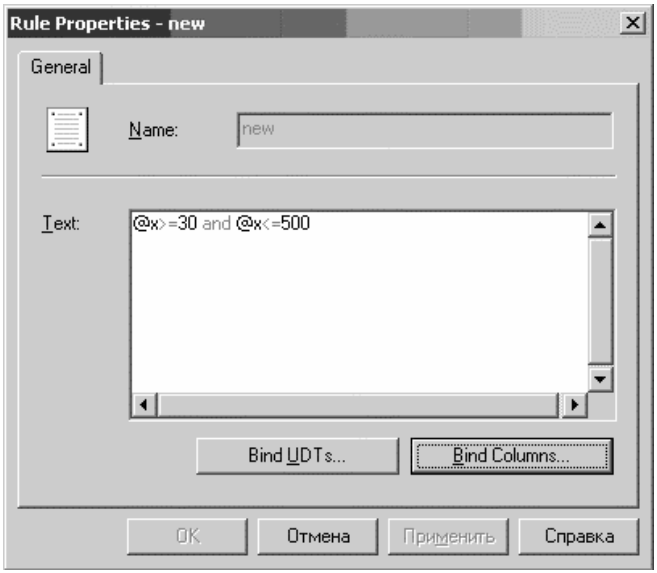

Рис. 4.3. Окно утилиты создания правила

Создание представлений. Синтаксис команды:

CRIATION VIEW [*owner.*]*view_name*
[(*column_name*[,*column_name*]…)]
[WITH ENCRYPTION]
AS *select_statement* [WITH CHECK OPTION]

где: view_name – имя создаваемого представления;

column_name – имя столбцов в представлении, эти имена заменяют имена реальных столбцов, которые берутся из реальных таблиц. Иногда столбцы в представлении могут создаваться в виде некоторой операции над исходными полями. Тогда исходного имени нет в принципе;

ENCRIPTION – наличие этой опции обеспечивает своего рода защиту, не позволяя пользователю определить, откуда представление извлекает столбцы;

WITH CHECK OPTION – включает контроль всех операций над строками для гарантии их сохранения при модификации данных с использованием представления;

AS select_statement – это предложение содержит информацию, каким образом и на основании каких таблиц создается представление.

Со специальной утилитой формирования представления мы познакомились на прошлой работе.

Создание триггера. Триггер – вид хранимой процедуры, выполняемый автоматически при попытке выполнить заданное действие с таблицей. Выполняется при попытке осуществить удаление, вставку или изменение данных. Вид SQL запроса на создание триггера:

Create trigger TrName ON table_name
FOR INSERT, UPDATE, DELETE
AS sql_statement

На рис. 4.4 приведено окно утилиты создания триггера. Она вызывается из редактора создания таблицы.

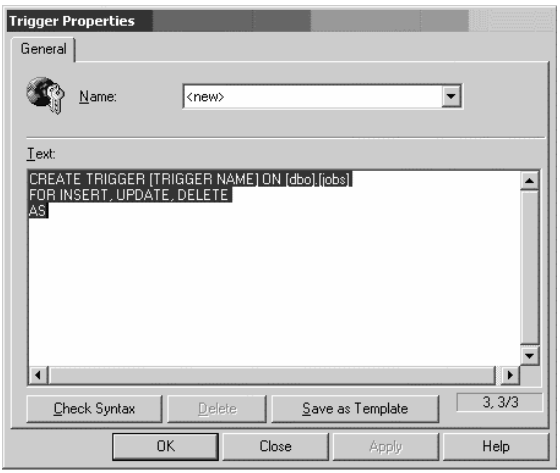

Рис. 4.4. Окно создания триггера

Выполнение работы

1. Первоначальное знакомство с базой данных Pubs.

• Загрузить компьютер и войти в систему со своим пользовательским login.

• Стартовать клиентскую часть SQL Server.

• Открыть в окне Server Manager базу данных Pubs, просмотреть группы и пользователей, которые допущены к работе с базой данных.

• Просмотреть объекты базы данных, таблицы, сохраняемые процедуры, представления, правила.

• Щелкнув два раза на имени, просмотреть структуру таблиц AUTHORS и JOBS.

2. Использование запросов SQL для изучения базы данных Pubs.

• Открыть инструмент SQL Query.

• Просмотреть объекты базы данных, зафиксированные в системной таблице SYSOBJECTS. Для этого выполнить следующий запрос:

```
SELECT name, type FROM sysobjects
```

В результате запроса используются следующие обозначения:

C – ограничение; D – значение по умолчанию; F – внешний ключ; K – первичный ключ; L – журнал; P – сохраняемая процедура; R – правило; RF – сохраняемая процедура для размножения; S – системная таблица; TR – триггер; U – пользовательская таблица; V – представление; X – расширенная сохраняемая процедура.

• Сохранить результат выполнения запроса на диске D: в директории LABORATORY.

• Составить и проверить действие следующих запросов:

♦ Просмотреть всю информацию об авторах (таблица AUTHORS).

♦ Просмотреть все имена авторов (фамилию и имя) и их ID.

♦ Просмотреть все имена авторов и города для авторов, проживающих в Калифорнии (CA).

♦ Просмотреть все имена авторов и города для авторов, проживающих в Калифорнии или Юте (UT). Использовать два варианта – оператор IN и логическое OR.

♦ Просмотреть все имена авторов, проживающих в городах, начинающихся на «C».

♦ Просмотреть все города, в которых проживают авторы, названия не должны повторяться.

♦ Просмотреть названия и цены книг в таблице TITLES, чья цена лежит в промежутке между 5 и 10 долларами. Использовать два варианта – операции сравнения и оператор BETWEEN.

Просмотреть все названия издательств (таблица PUBLISHERS), для которых не определено название штата.

♦ Показать все книги, в названии которых хотя бы один раз встречается буква «p» и цена которых превышает $16.

Синтаксис запросов отразите в листе отчетности. Сохранить на диске и распечатать результат последнего запроса.

3. Изучение использования функций агрегирования.

• Определить полное число записей в таблице AUTHORS.

• Определить полную сумму скидок (discount) в таблице DISCOUNT.

• Определить среднюю, максимальную и минимальную цену (price) книги в таблице TITLES.

• Определить среднюю цену книг в таблице titles, сгруппированную по идентификаторам издательства (внешний ключ), вывести идентификатор издательства и среднюю цену.

• Проделать то же, что и в предыдущем пункте, но для средних цен ≥ 10.

4. Изучение возможностей представления данных.

• Просмотреть названия и цены книг в таблице TITLES, чья цена лежит в промежутке между 5 и 10 долларами. При выводе информации озаглавить столбцы: Название, Цена.

• Теперь проделать то же самое, ограничив вывод названия подстрокой от 1 до 20 символов (функция substring (столбец, первый символ, последний символ)). Сохранить и распечатать результат запроса.

• Вывести список фамилий авторов по алфавиту (из таблицы AUTHORS), озаглавив колонку – фамилия автора.

• Вывести из таблицы TITLES тип книги (type) и среднюю цену для книг, у которых значение royalty = 10, сгруппированное и упорядоченное по типам.

5. Функции изменения данных.

• Перейти в таблицу, созданную на предыдущем занятии.

• Добавить строку в таблицу «тип контейнера».

• Изменить название контейнера в этой строке.

• Уничтожить эту строку.

После всех операций, использовать предложение SELECT для проверки результатов.

6. Создать представление. В процессе выполнения предыдущей лабораторной работы мы создали представление с помощью помощника. Сейчас мы создадим представление с помощью запроса на языке SQL. Представление должно содержать дату транзакции и имя пользователя, проводящего транзакцию.

7. Создать правило, ограничивающее вес контейнера величиной 400. Убрать аналогичное ограничение, установленное на предыдущей лабораторной работе. Проверить функционирование правила.

8. Создать хранимую процедуру, реализующую проведение транзакции.

Имя процедуры – new_trans.

Список формальных параметров:

@us_id – идентификатор пользователя, тип – numeric;

@tr_type_id – идентификатор типа транзакций, тип – numeric;

@con_id – идентификатор контейнера, тип – numeric;

@weight – вес контейнера, тип – real;

@loc_id – идентификатор места расположения, тип – numeric.

Процедура должна быть оформлена как транзакция. В начале и в конце стоят команды BEGIN TRANSACTION и COMMIT TRANSACTION соответственно.

В теле процедуры необходимо декларировать (команда Declare) две переменных @@smd – тип smalldatetime и @@trid – идентификатор транзакции. Далее последовательно выполняются следующие команды:

Записать (select) в переменную @@smd значение сегодняшней даты с помощью функции GETDATE().

Вставить в таблицу transactions информацию о новой транзакции.

Получить значение идентификатора этой транзакции (максимальное значение идентификатора) и записать его в переменную @@trid.

Вставить новую запись в таблицу container_transaction.

Проверить работу хранимой процедуры.

9. Создать триггер, который на попытку удаления строки из таблицы «Пользователи» будет выводить сообщение (сообщение выводится командой Print «текст сообщения») и отменять выполнение команды. Попытаться выполнить удаление.

10. Выйти из SQL Server и выключить компьютер.

Лист отчетности

Студент_____, группа_____

Количество объектов в базе данных Pubs_____
Синтаксис запросов пункта 2:

Результаты использования функций агрегирования:
полное число записей в таблице AUTHORS_____
суммарная скидка в таблице DISCOUNTS_____
средняя цена книги _____
максимальная цена книги_____
минимальная цена книги_____

Сколько издательств имеют среднюю цену книги больше или равную 10 долларам?_____

Фамилия первого и последнего автора по алфавиту_____

Дата_____ Подпись преподавателя_____

Лабораторная работа 5
СОЗДАНИЕ БАЗЫ ДАННЫХ В СУБД MICROSOFT ACCESS

Цель работы: научить студентов созданию реляционных баз данных в программном средстве MS Access. Продемонстрировать основные способы поддержания целостности данных в базах данных Access. Дать основные представления о создании простейшего интерфейса для ввода и просмотра данных, сортировки информации и создании запросов к базе данных.

Теоретические основы

Система управления базами данных Microsoft Access является одной из наиболее популярных среди настольных систем, поддерживающих реляционные модели базы данных. СУБД Microsoft Access входит в пакет MS Office, и в настоящее время эксплуатируется версия MS Access 2003. Данный программный продукт является мощным средством управления данными, позволяющим создавать приложения для работы с базами данных различной сложности. Основные черты Microsoft Access:

- стандартный интерфейс пакета MS Office;
- простой способ создания основных объектов базы данных, позволяющий интуитивно выполнять основные функции даже неопытным пользователям;
- средства автоматизации создания запросов;
- средства автоматизации создания графического интерфейса пользователя с помощью форм;
- наличие языка программирования, позволяющего создавать более сложные управления данными;
- поддержка структурированного языка запросов SQL;
- возможность работы с базами данных различных форматов через использование возможностей ODBC;
- возможность подключения базы данных к World Wide Web;
- легкость экспорта данных в MS SQL Server.

Специалисты рекомендуют использовать MS Access для конструирования приложений клиент/сервер для случаев, удовлетворяющих следующим требованиям:

- количество пользователей ограничено двадцатью;
- объем базы данных не превышает 500 Мб;
- умеренные требования к надежности и защите данных.

В этом случае MS Access легко формирует систему файлового сервера. Рассмотрим последовательность действий при создании и работе с базами данных в Access.

Создание таблиц базы данных. При запуске СУБД появляется заставка, предлагающая открыть уже существующую базу данных, перечисляя все используемые ранее базы данных, либо приступить к созданию новой. Выбираем последнее и нажимаем клавишу подтверждения. Определяем папку, в которой будет расположена база данных, и задаем ее имя. На экране открывается форма, которая в дальнейшем, по мере наполнения базы данных, будет предоставлять нам интерфейс для создания объектов базы и просмотра данных (рис. 5.1).

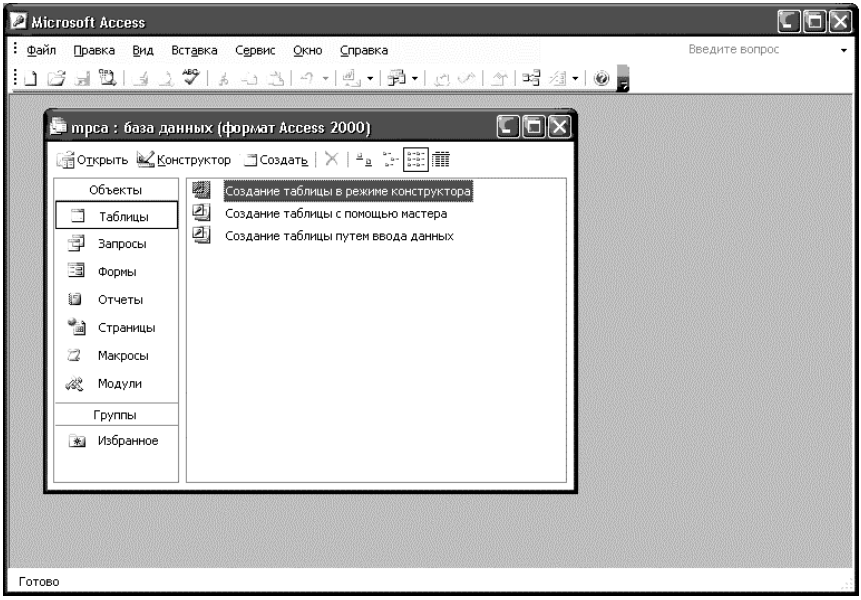

Рис. 5.1. Форма работы с базой данных СУБД MS Access

Основное действие при создании базы данных – создание таблицы. Access предлагает три способа создания таблиц. Можно ис-

пользовать помощник, который предлагает огромный выбор уже готовых шаблонов таблиц. Можно создать таблицу «по факту», сразу заполняя поля данными. Но опытный пользователь всегда предпочтет создание таблицы с использованием конструктора таблиц, поскольку этот подход дает возможность полностью определять типы данных и ограничения на столбцы. На рис. 5.2 представлена форма создания таблицы в режиме конструктора.

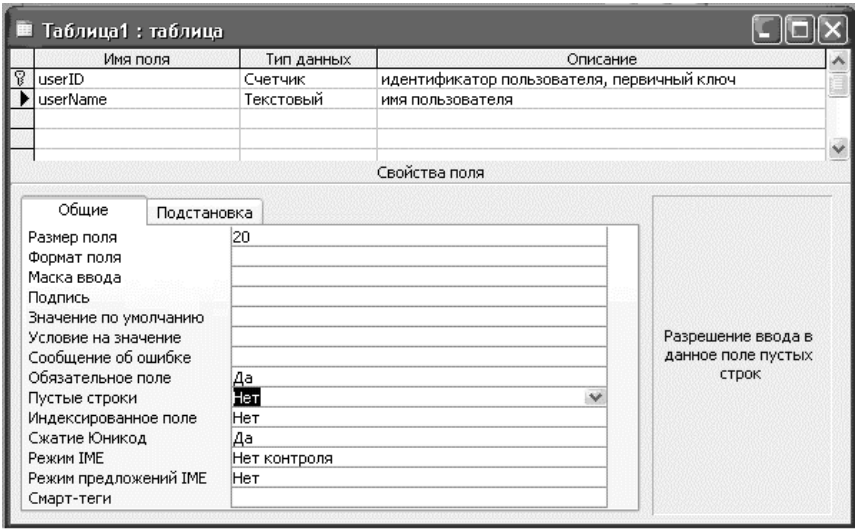

Рис. 5.2. Форма создания таблицы в режиме конструктора

При сохранении таблицы вводим ее имя. Как видно из рисунка, мы определили ключевое поле и задали определенные ограничения на поля данных.

В отличие от MS SQL Server, Access позволяет существенно модифицировать таблицу после ее создания. Для этого достаточно выбрать таблицу и нажать клавишу «Конструктор». Открывается окно конструктора, и пользователь может вводить любые изменения.

Демонстрируется возможность задавать тип данных в поле. Для этого надо щелкнуть в столбце Тип данных нужного поля. Откроется список возможных типов данных. Можно задать длину дан-

ных, формат поля, значение по умолчанию и то, что поле обязательно должно быть заполненным. Обратите внимание на знак ключевого поля и тип данных – счетчик. Для такого типа наращивание номера ключевого поля происходит автоматически, и номера никогда не повторятся.

Для создания ключевого поля необходимо выбрать поле и щелкнуть второй клавишей мыши. Появляется всплывающее меню, в котором можно задать значение Главного ключа. Если выделено несколько полей, то будет создан составной ключ.

Чтобы заполнить таблицу, достаточно щелкнуть на иконке таблицы, и ввести данные непосредственно в поля таблицы. При вводе данных будут осуществляться проверки на выполнение всех ограничений (рис. 5.3)

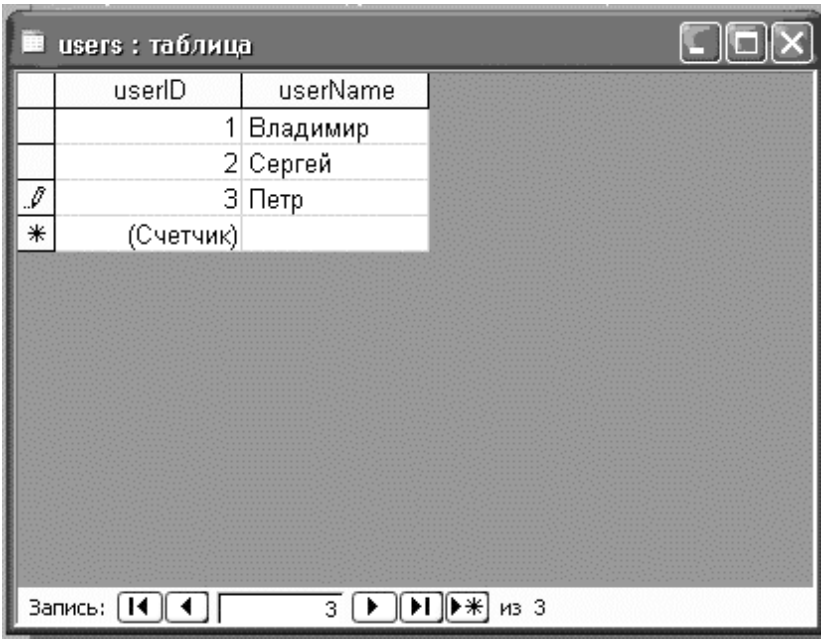

Рис. 5.3. Ввод данных в таблицу

Установление связей между таблицами. Для создания связей между таблицами в главном меню открывается меню «Сервис» и в

нем выбирается опция «Схема связей». Появляется окно, в котором перечислены все таблицы создаваемой базы данных. С помощью клавиши «добавить» помощник добавляет их в выделенное окно. В результате мы получаем окно, в котором расположены прямоугольники, схематически представляющие таблицы. Для связи одной таблицы с другой помещаем курсор мышки на ключевом поле внешнего ключа таблицы, нажимаем кнопку и буксируем специальный знак на ту таблицу, в которой этот ключ является главным. Затем программа запрашивает подтверждение для установления связи. После завершения этой операции в окне появляется схема связей, представленная на рис. 5.4.

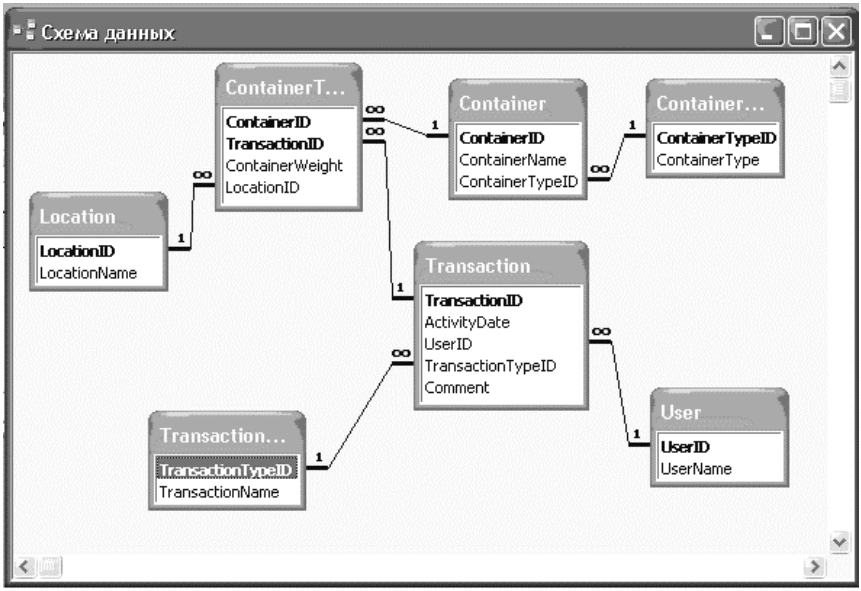

Рис. 5.4. Схема связей между таблицами

Для редактирования связей достаточно навести курсор мышки на линию связи и щелкнуть два раза. Появляется окно определения свойств связи (рис. 5.5).

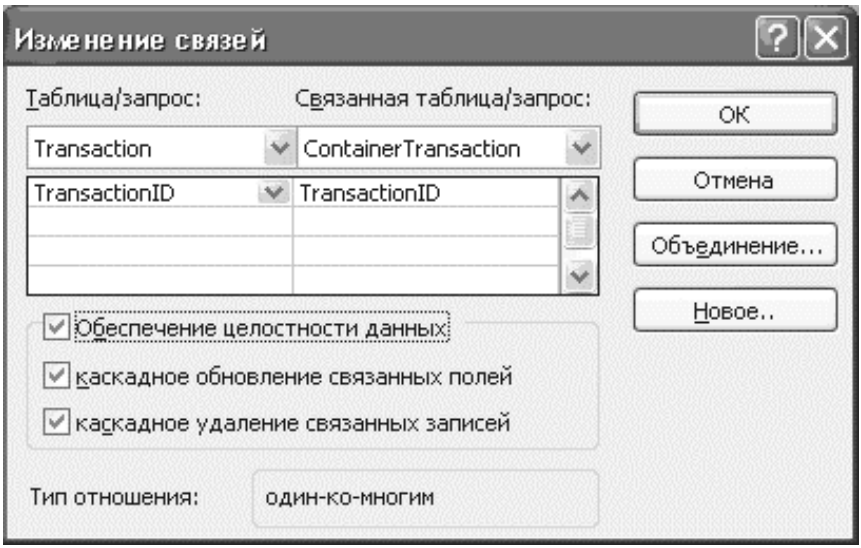

Рис. 5.5. Окно редактирования свойств связи между таблицами

Существует возможность включения обеспечения целостности данных с помощью ключевых полей и задания каскадного удаления и обновления связанных полей при модификации базы данных.

Организация целостности данных препятствует вводу некорректной информации. При попытке ввести данные со ссылкой на несуществующую запись генерируется отказ с указанием причины. При попытке удаления записи, на которую имеется ссылка, появляется запрос на подтверждение каскадного удаления.

Создание запросов. Реляционная модель данных в MS Access несколько отличается от модели SQL Server. В Access отсутствуют понятия представления, хранимой процедуры и триггера. Тем не менее, существует понятие запроса в Access, которое близко к понятию представления, создаваемого с помощью запроса SQL в SQL Server. Для создания запросов в Access существует специальный помощник, автоматизирующий эту процедуру.

В режиме конструктора выбираются таблицы, информация из которых будет составлять запрос. Затем отмечаются поля, которые будут включены в запрос. Имеется возможность определять сорти-

ровку по столбцам, или формировать определенные условия выборки. На рис. 5.6 приведена форма определения запроса.

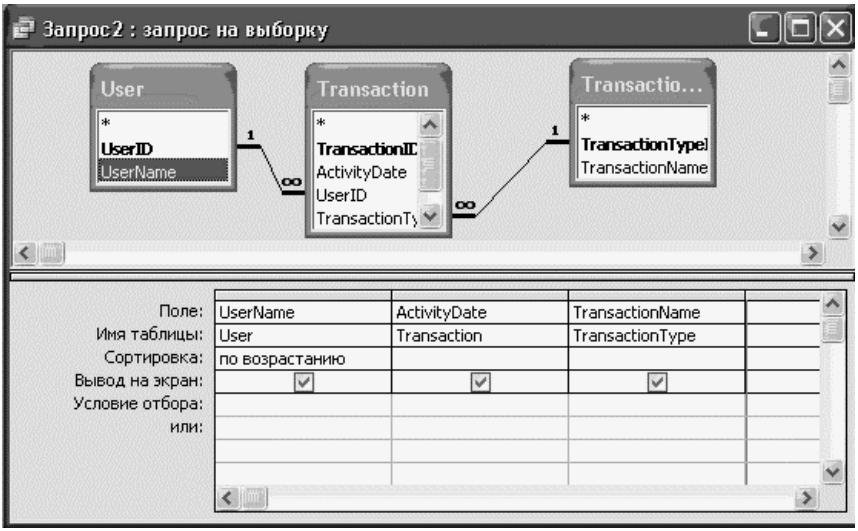

Рис. 5.6. Формирование сложного запроса

Если установить режим просмотра запроса – «режим SQL», то можно увидеть текст запроса, который был автоматически сформулирован:

```
SELECT User.UserName, Transaction.ActivityDate, Transaction-
Type.TransactionName
FROM [User] INNER JOIN (TransactionType INNER JOIN [Transac-
tion] ON TransactionType.TransactionTypeID = Transac-
tion.TransactionTypeID) ON User.UserID = Transaction.UserID
ORDER BY User.UserName;
```

Во всем остальном с запросом можно обращаться так же, как с обычной таблицей.

Создание форм. В отличие от SQL Server, Access имеет в своем составе средства формирования пользовательского интерфейса. Для его создания используется язык программирования VBA (Visual Basic for Application). Для форматирования, удобства представ-

ления данных и создания дополнительных возможностей для поддержки целостности баз данных в Access предусмотрены специальные объекты, которые называются формами. Форма – это элемент интерфейса, на котором расположены элементы управления, связанные с полями базы данных. Для запуска этих элементов достаточно щелкнуть на иконке соответствующей формы. Access позволяет создавать стандартные формы с помощью интерактивного «помощника» и сложные формы с использованием дополнительных объектов и программирования.

На рис. 5.7 представлена форма для отображения содержимого запроса trans_info, созданная в автоматическом режиме.

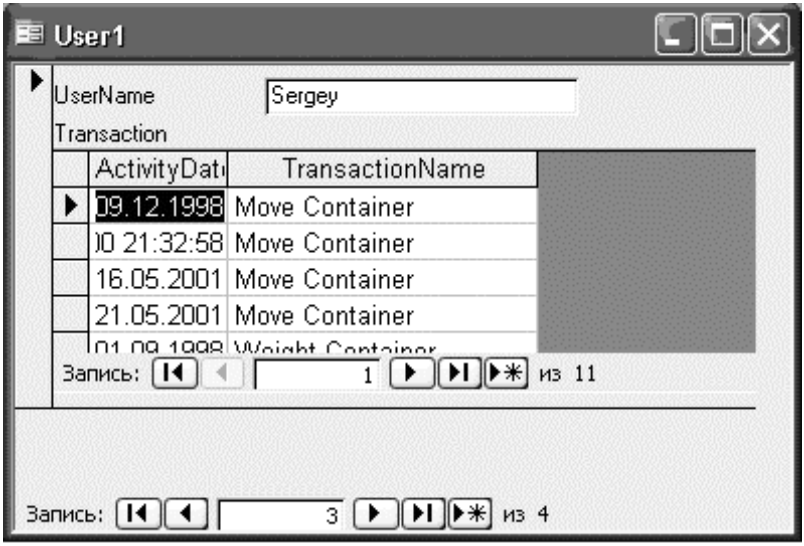

Рис. 5.7. Форма для просмотра представления

Создание отчетов. Access позволяет формировать отчеты, которые можно распечатать нажатием соответствующей кнопки на панели инструментов. Для формирования отчетов используется помощник, аналогичный помощнику создания запросов. На рис. 5.8 представлен фрагмент отчета, сформированного на основании содержимого запроса.

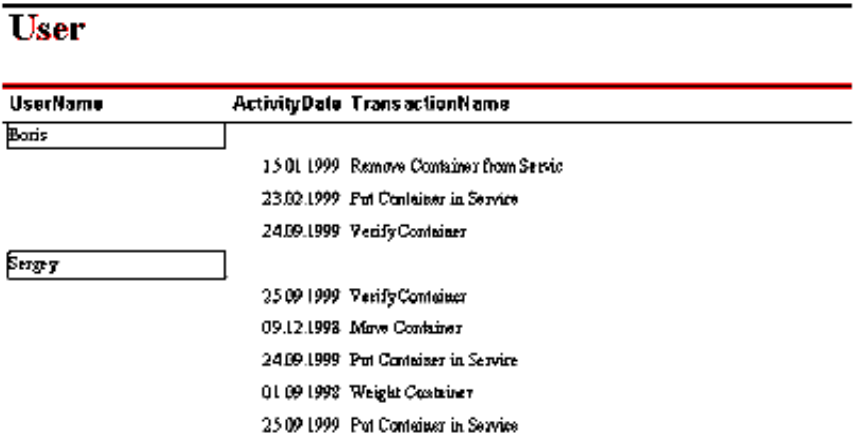

Рис. 5.8. Отчет о проведении транзакций

Экспорт информации в SQL Server. База данных, созданная в MS Access, легко может быть экспортирована в SQL Server. Для этого необходимо стартовать сервер и запустить на выполнение помощник по импорту баз данных TOOLS → Data transformation services → Import Data. Далее необходимо последовательно отвечать на вопросы: определить формат базы данных, ее имя и расположение, имя базы данных в SQL Server и т.п. База данных будет импортирована со всеми данными. Следует учесть, что при экспорте информации не передаются связи между таблицами, их затем надо устанавливать вручную.

Выполнение работы

1. Включить компьютер, загрузить систему и стартовать Access.
2. Задать режим создания базы данных. Имя базы данных – МРСА#, где # – номер компьютера.
3. Создать и заполнить все таблицы учебной базы данных. Таблицы взять из лабораторной работы 3.

Примечание.

Использовать следующие свойства полей:

- Все главные ключи, кроме композиционного, в таблице «Container-Transaction» имеют свойство – счетчик.

• Все поля должны иметь значение.

• Ввести значение по умолчанию для одного-двух полей любой таблицы.

• Распечатать содержимое одной из таблиц.

• Установить связи между таблицами согласно рис. 5.4 с установлением обеспечения целостности данных.

4. Распечатать таблицу связей между таблицами.

5. Создать запрос вывода информации, аналогичный представлению в лабораторной работе 3.

6. Создать форму вывода информации этого запроса.

7. Создать и распечатать отчет для этого запроса.

8. Импортировать базу данных в SQL Server.

9. Заполнить и подписать у преподавателя отчет о проделанной работе.

Лист отчетности

Студент_____, группа_____

Создана база данных (указать имя компьютера, полный путь и имя базы данных):

_____

К работе прилагаются распечатки:

Таблица _____,

Схема связей между таблицами.

Отчет по запросу _____.

Дата _____ Подпись преподавателя_____

Лабораторная работа 6
СОЗДАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ С ПОМОЩЬЮ VISUAL BASIC.NET

Цель работы: ознакомить студентов со средством разработки Visual Studio.Net, включая основные элементы управления, используемые в пользовательских приложениях, получение навыков работы с интерфейсом средства разработки, создание простейших пользовательских приложений.

Теоретические основы

Инструментальные средства разработки. Третьей составной частью базового программного обеспечения компьютеризированных СУиК ЯМ являются инструментальные средства разработки. Они служат для создания графического интерфейса пользователя и могут также реализовывать некоторые бизнес-правила при формировании приложения клиент/сервер с толстым клиентом. Современные средства разработки обычно удовлетворяют двум требованиям: они поддерживают возможность визуального проектирования интерфейса и используют объектно-ориентированный язык программирования.

Для изучения основных приемов создания интерфейсов компьютеризированных СУиК ЯМ в нашем курсе выбрана одна из последних разработок Microsoft – пакет Visual Studio.Net. Он прекрасно совместим с операционными системами семейства Windows NT и обладает развитыми механизмами связи с базами данных. Пакет Visual Studio.Net позволяет программировать на ряде языков высокого уровня: C++, C#, MS Java, Visual Basic.Net. Для реализации примеров создания элементов приложения СУиК ЯМ мы будем использовать Visual Basic.Net. Прежде всего этот язык является дальнейшим развитием популярного языка Visual Basic, разработанного специально для создания приложений баз данных. Последняя версия языка – Visual Basic.Net является полностью объектно-ориентированным языком с поддержкой наследования и полиморфизма. Наконец, его синтаксис достаточно прост, и для понимания логики реализуемых примеров не требуется глубокого знания языка программирования.

Все последующие примеры реализованы с использованием пакета MS Visual Studio.Net 2003.

Интегрированная среда разработки Visual Studio.Net. При запуске Visual Studio.Net пользователь выбирает язык разработки (в нашем случае – Visual Basic.Net), тип приложения (Windows Application, Web Application), название проекта и его расположениие на диске. После подтверждения указанных данных, открывается окно разработки нового проекта (рис. 6.1).

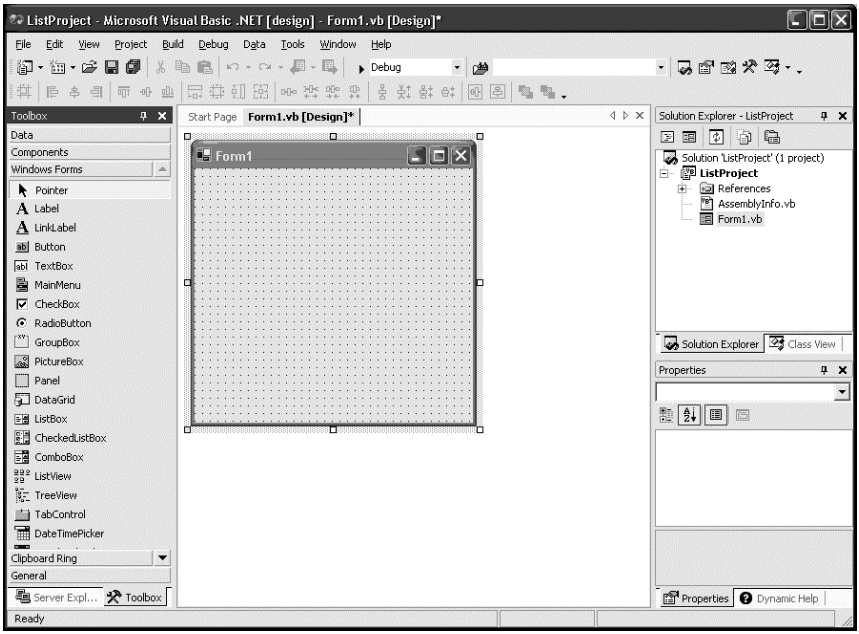

Рис. 6.1. Окно интегрированной среды разработки VB.Net

Основные компоненты интегрированной среды разработки:

1. Строка меню – содержит все функции и команды для создания приложений, редактирования программного кода, форматирования, отладки и т.п.

2. Панель инструментов – содержит набор пиктограмм, нажатие на которых активизирует соответствующие опции меню.

3. Палитра элементов управления (Toolbox) – содержит элементы управления, которые может использовать разработчик при создании пользовательского интерфейса.

4. Форма – «заготовка» для окна среды Windows, в котором будет работать создаваемое разработчиком приложение.

5. Окно свойств (Properties) – отображает свойства элемента управления, выделенного в данный момент.

6. Окно проекта (Solution Explorer) – содержит иерархическую структуру файлов, форм и модулей текущего проекта. Позволяет быстро переключаться между компонентами проекта.

Проекты Visual Basic. При разработке пользовательских приложений в среде Visual Basic создается проект Visual Basic. Проект содержит в себе все файлы, в которых сохраняется информация об элементах управления и формах, используемых в приложении, программный код, связанный с обработкой тех или иных событий и т.п. Для более легкого управления проектом и оперативного перехода от одних файлов проекта к другим используется окно проекта (Solution Explorer). Пользователь, выделяя в окне необходимую ему часть проекта, загружает требуемый файл в рабочее окно и может переходить к работе с ним. Нажатием на кнопки, помещенные вверху, можно переключаться между визуальным представлением формы объекта к содержимому его программной части.

Работа с элементами управления Visual Basic.Net. В среде Visual Studio.Net реализованы графические объекты или элементы управления. Процесс создания графического интерфейса заключается в том, что пользователь перемещает на форму необходимые элементы управления, определяет их свойства и пишет программный код обработки того или иного события, связанного с данным элементом управления.

Стандартные элементы управления расположены на панели элементов управления среды разработки. Рассмотрим те из них, которые используются при создании интерфейса в данной лабораторной работе.

Форма (Form). После открытия проекта на экран выводится форма – прототип окна, которое будет выводиться на экран при запуске программы. Это первый и пока единственный элемент управления. В начальный момент он выделен и готов к редактиро-

ванию. В окне свойств элемента перечислены свойства объекта «Форма». Перечислим основные.

- Name (имя) – идентификатор элемента управления формой. Существует необязательное правило наименования объектов. Принято, что бы имя формы имело впереди три строчные буквы – frm. Например, frmTransaction.
- Caption (заголовок) – содержит строку, которая выводится наверху формы.
- Height и Width определяют размеры формы.
- Top и Left определяют положение формы на экране (координаты левого верхнего угла формы в пикселах).

Два основных события, связанные с формой, это открытие формы и ее закрытие. Событие открытия формы обрабатывает метод Load. Далее приводится текст метода обработки события загрузки формы frmTransaction. При загрузке задается значение заголовка формы.

```
Private Sub frmTransaction_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        frmTransaction.Caption = “Осуществить транзакцию”
End Sub
```

При закрытии формы работает метод Closing. Обычно в этот метод включают действия по очистке памяти, закрытию соединений с базами данных и файлами.

Чтобы определить, какие еще события существуют у элемента управления, надо отметить объект и нажать клавишу F1. При этом открывается окно подсказок, в котором, выбирая функци, можно познакомиться со всеми существующими для этого элемента событиями.

Пользовательское меню (MainMenu). Пользовательское меню является необходимым элементом большинства приложений. VB.Net имеет на палитре объектов специальный элемент управления, позволяющий легко создавать и редактировать пользовательские меню. Объект «перетаскивается» из палитры на форму. После чего он оказывается выделенным и можно задавать его свойства. На рис. 6.2 видно пользовательское меню в процессе формирования.

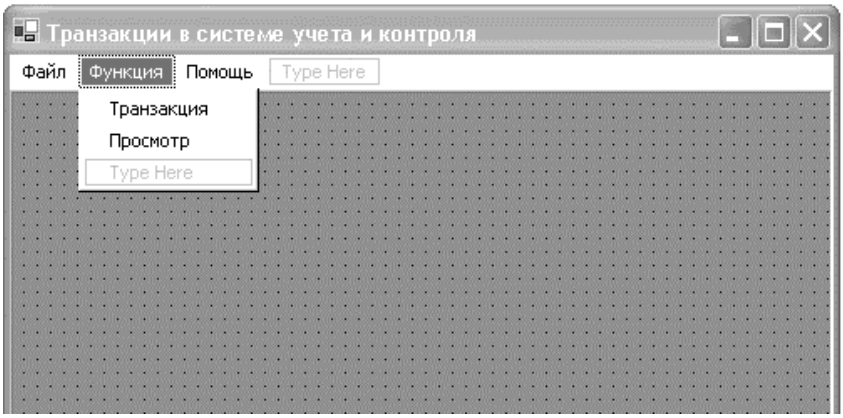

Рис. 6.2. Формирование пользовательского меню

Имена даются каждой «кнопке» меню. Принято соглашение, по которому имя этого объекта начинается с сокращения mnu. Например, mnuHelp – имя кнопки вызова помощи. Основное событие, связанное с меню, – нажатие кнопки меню. Чтобы создать метод обработки этого события, достаточно навести курсор мышки на кнопку и щелкнуть два раза левой кнопкой мышки. Откроется окно с заготовкой подпрограммы метода обработки события. Пользователю достаточно написать требуемый текст. Ниже приведена распечатка метода обработки нажатия кнопки, по которому открывается новая форма.

```
Private Sub mnuTrans_Click(ByVal sender As System.Object,
 ByVal e As System.EventArgs) Handles mnuTrans.Click
      Dim f_Trans As New frmTrans
      f_Trans.MdiParent = Me
      f_Trans.Show()
End Sub
```

Существует еще одно полезное свойство элемента «кнопка меню» – Visible. Это свойство принимает значение истина или ложно (True или False). Когда его значение равно «ложно», кнопка делается невидимой, а функция меню – недоступной для пользователя.

Используя это свойство, можно управлять доступом пользователей к различным функциям системы.

Кнопка управления (CommandBatton). Этот элемент управления по своим свойствам аналогичен описанным выше кнопкам меню, но представляется в виде отдельной кнопки на форме. Начальные буквы имени обычно устанавливаются в значении «cmb».

Основные свойства элемента CommandButton:

♦ Name (имя) – идентификатор объекта «Кнопка управления».

♦ Caption – надпись на кнопке. Следует отметить одну особенность, что если поставить перед буквой, входящей в заголовок, символ «&», то этот символ будет подчеркнут на кнопке и будет являться «коротким ключом» для нажатия кнопки.

Метка (Label). Этот элемент управления обычно используется для вывода текстовых строк, заголовков и сообщений. Элемент управления обладает теми же свойствами, определяющими его расположение на форме, свойством Имя и Заголовок. В имени объекта первые буквы «lbl».

Текстовое окно (TextBox). Позволяет, так же, как и элемент Label, выводить текстовую информацию. Отличие заключается в том, что пользователь может редактировать эту информацию во время работы приложения. Этот объект обычно используется для ввода тестовых строк в программу. Первые буквы в имени объекта txt.

Обратим внимание на свойства Multiline объекта TextBox. Когда это свойство принимает значение True (истина), то данный объект может содержать текст, разбитый на строки.

Список (ListBox). Элемент управления Список позволяет хранить и просматривать наборы текстовых строк. Среди многочисленных свойств и методов этого объекта следует отметить следующие:

♦ свойство Sorted. При установке значения этого свойства True, вводимая информация будет сортироваться по алфавиту.

♦ метод AddItem. Данный метод добавляет строки в список. Синтаксис использования метода следующий:

```
lstPhoneList.AddItem “Иванов 613-88-45”
```

Привязка объектов к форме. При изменении размеров формы в процессе работы приложения встает вопрос о положении элементов, размещенных на форме. Для того чтобы определить их привязку к сторонам формы, используется свойство Anchor (якорь). При его установке появляется условное изображение объекта с четырьмя связями. Необходимо отметить те стороны, расстояние от которых не будет меняться при изменении формы. Если отмечено более двух сторон, то размеры элемента будут изменяться вместе с изменением размеров формы.

С рядом элементов управления мы познакомимся при выполнении следующей работы. А сейчас надо познакомиться еще с рядом возможностей, предоставляемых Visual Studio.Net для формирования графического интерфейса.

Окно сообщения (MassageBox). С помощью этого окна можно организовать простой диалог с пользователем. В окне выводится сообщение, и пользователю предлагается выбрать дальнейшее действие путем нажатия одной из кнопок, выведенных в окне. Число кнопок и их значение могут варьироваться. На рис. 6.3 приведено стандартное окно диалога, которое запрашивает разрешение на закрытие формы.

При нажатии различных кнопок подпрограмма возвращает различные целочисленные значения, анализируя которые, программа будет выполнять различные действия.

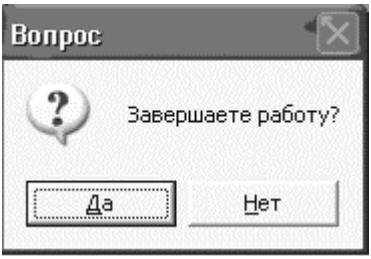

Рис. 6.3. Стандартное окно диалога

Синтаксис вызова данного окна следующий:

```
Rezalt = MsgBox(“Завершаете работу?”,_
    MsgBoxStyle.YesNo + MsgBoxStyle.Question,“Вопрос”)
```

MsgBoxStyle.YesNo и MsgBoxStyle.Question являются системными константами, имеющими целочисленное значение. Они определяют количество и наименование кнопок диалога и рисунок (иконку) на форме. Возможные значения этих констант, а также возможные значения возвращаемого значения, определяются с помощью контекстной подсказки.

Отметим также то, что данное окно является модальной формой. Это означает, что работа приложения приостанавливается до тех пор, пока не будет реализован выбор альтернативы и модальная форма не закроется.

Многодокументный интерфейс (MDI). При работе достаточно сложного приложения различные его функции могут реализовываться на различных формах, которые открываются и закрываются по желанию пользователя. При этом существуют две возможности организации такого интерфейса. В первом случае формы открываются независимо друг от друга и могут располагаться в любом месте экрана. Во втором случае вызываемая форма (дочерняя) открывается внутри главной (родительской) формы. Обычно на главной форме реализовано меню системы.

Для организации MDI интерфейса необходимо у главной формы установить свойство IsMdiContainer в значение True. При вызове дочерних форм указывается, что они являются дочерними. Пример открытия дочерней формы приведен в тексте примера, посвященного обработке нажатия кнопки меню.

Завершение работы приложения. Работа приложения завершается посылкой команды End. По этой команде завершается выполнение приложения, закрываются соединения с файлами и базами данных и очищается память. Пользователь может управлять закрытием приложения, например, с помощью модального окна диалога. Однако следует учесть, что на форме есть две кнопки, нажатие которых принудительно закрывает форму. Чтобы отменить их действие, необходимо свойству Cancel соответствующего объекта придать значение True. После этого закрытие приложения возможно только подачей пользовательской команды. Текст подпрограммы обработки этого действия приведен ниже в инструкции по выполнению работы.

Работа с текстовым файлом. Поскольку обучение программированию на конкретном языке не является целью данной работы,

будет приведен пример работы с файлом, необходимым для выполнения данной работы. В процессе выполнения работы будет необходимо открыть текстовый файл, считать из него информацию и выгрузить ее в текстовое окно. Эта операция достигается следующим набором операторов:

```
Dim FS As FileStream
Dim SR As StreamReader
    FS = New FileStream("help_file.txt", FileMode.Open)
    SR = New StreamReader(FS)
    txtHelpBox.Text = SR.ReadToEnd
    SR.Close()
    FS.Close()
```

Здесь открываются два специальных объекта работы с файлами: FileStream – поток, связанный с конкретным файлом, и объект StreamReader, осуществляющий чтение из потока. При открытии потока указывается адрес файла. В нашем примере файл размещен в том же каталоге, откуда стартует программа. Для чтения текста используется команда ReadToEnd (читать полностью). Эта команда переносит содержимое файла в строковую переменную. В завершении программы объекты работы с файлами закрываются.

Выполнение работы

1. Войти в систему Windows XP и запустить программное средство Visual Studio.Net.

2. Изучить правила работы с элементами управления в процессе создания простого пользовательского приложения. Для этого: создаем простое приложение, состоящее из текстового окна, окна списка и трех кнопок. Нажатие одной кнопки переносит содержание текстового окна в список, очищая одновременно текстовое окно. Нажатие второй кнопки очищает список. Нажатие третьей кнопки завершает работу приложения.

Вид готовой формы представлен на рис. 6.4.

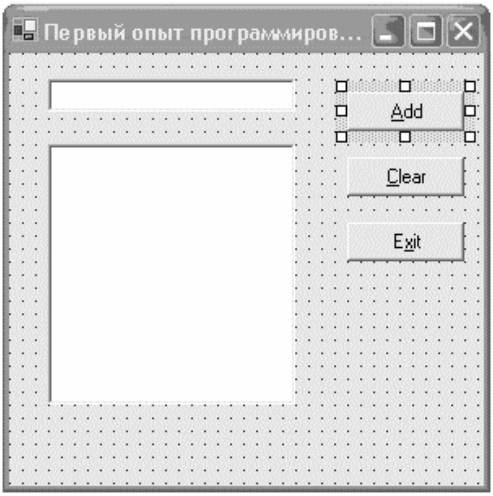

Рис. 6.4. Готовая форма первого проекта

Для создания этого приложения нужно выполнить последовательно следующие операции.

• Стартовать опцию меню File → New → Project. Выбираем язык программирования – Visual Basic, тип приложения Windows Application. Определить название проекта ListProject, определить место хранения файлов проекта: C:\Magistr\"Личная папка" и нажать кнопку «OK».

• Установить следующие свойства открывшейся формы:

Name – frmListProject;

Text – "Первый опыт программирования на VB".

• На панели Toolbox выбрать объект TextBox и поместить его на форме. Придать ему соответствующие размеры и положение, в свойствах указать:

Name – txtString;

Text – оставить пустым.

• Вывести на форму объект ListBox, придать ему желаемые размеры и положение и указать свойства:

Name lstList.

• Вывести на форму три объекта Command Button, придать им желаемые размеры и положения на форме. Задать имена и заголовки:

Имя – cmdAdd, Текст – &Add;

Имя – cmdClear, Текст – &Clear;

Имя – cmdExit, Текст –E&xit.

• Так как форма была переименована, необходимо установить в качестве стартовой формы ее новое имя. Для этого в окне Solution Explorer щелкните правой кнопкой на имени проекта и в открывшемся меню вызовите опцию Properties. В открывшемся окне выберите новый стартовый объект (Startup Object). Сохраните проект, нажав опцию главного меню – File → Save all.

• Генерировать код приложения:

♦ Программировать опцию «Выход». Щелкнуть два раза на кнопке «Exit», записать в текст процедуры обработки события единственную команду – End.

♦ Программировать опцию «Добавить». Щелкнув два раза на кнопке «Add», ввести код добавления содержимого текстового окна в список:

```
Private Sub cmdAdd_Click(ByVal sender As System.Object,_ ByVal e
As System.EventArgs) Handles cmdAdd.Click
    lstList.Items.Add(txtString.Text)
    txtString.Text = “”
End Sub
```

Жирным шрифтом выделен текст, который должен вставить учащийся. Первая строка кода добавляет текст из текстового окна в список, вторая – очищает текстовое окно.

• Программировать кнопку «Clear» – очистки списка, вводя следующий код:

```
Private Sub cmdClear_Click(ByVal sender As System.Object,_ ByVal e
As System.EventArgs) Handles cmdClear.Click
    lstList.Items.Clear()
End Sub
```

• Сохранить проект, запустить его, проверить работоспособность. Проверить функциональность горячих клавиш.

• Модифицировать список таким образом, чтобы строки выводились в алфавитном порядке. Для этого установить в свойстве Sorted объекта ListBox значение «True». Проверить сортировку в случаях ввода латинских и русских символов.

• Сохранить приложение, продемонстрировать преподавателю его работу.

3. Начинаем создавать интерфейс пользователя для работы с созданной на предыдущих занятиях базой данных. Изучим в этом разделе следующие возможности:

• создание многооконного интерфейса;

• создание вложенного меню с помощью редактора меню;

• программирование кнопок меню;

• использование стандартного окна Сообщение;

• работа с текстовыми файлами в Visual Basic.

4. Создание MDI формы и пользовательского меню.

• Открыть новый проект с именем DB_Interface и расположить его в своей личной папке.

• Задать следующие свойства первой формы:

Name – frmMPCA

Caption – “Транзакции в системе учета и контроля”.

IsMdiContainer – True.

Установить примерно следующие размеры формы:

Высота – 400,

Ширина – 500.

Установить стартовое расположение формы в центре экрана.

• Приступить к созданию главного меню. Выбрать в Toolbox объект MainMenu и поместить его на форме. Создать систему вложенных меню, представленных в табл. 6.1.

Сдвинутые в таблице наименования опций меню являются вложенными.

• Сохранить проект и проверить работу приложения.

Таблица 6.1

Свойства опций меню

Опция меню	Caption	Name
Файл	&Файл	mnuFile
Выход	&Выход	mnuQuit
Функция	Ф&ункция	mnuFunction
Транзакция	&Транзакция	mnuTrans
Просмотр	&Просмотр	MnuLook
Помощь	По&мощь	MnuHelp
О программе	&О программе	MnuAbout
Описание	Описа&ние	MnuDescript

5. Программирование кнопок меню.

• Программировать нажатие кнопки «Выход».

• Открыть опцию меню Файл → Выход, щелкнуть два раза мышкой и вставить в подпрограмму вызов окна диалога, который запрашивает разрешение на закрытие приложения. Вид окна представлен на рис. 6.3. Вставить в подпрограмму обработки следующий программный код:

```
If (MsgBox("Завершаете работу?", MsgBoxStyle.YesNo + _
   MsgBoxStyle.Question, "Вопрос") = MsgBoxResult.Yes) Then
        End
End If
```

Сохранить проект, проверить результат.

Сделаем так, чтобы при закрытии приложения этот вопрос появлялся в любом случае, даже при нажатии кнопок безусловного закрытия формы. Для этого вставляем код обработки события закрытия формы

```
Private Sub frmMPCA_Closing(ByVal sender As System.Object,_ By-
Val e As System.ComponentModel.CancelEventArgs)_
 Handles MyBase.Closing
      e.Cancel = True
      Call mnuQuit_Click(Me, e)
End Sub
```

Сохранить проект, проверить результат.

• Пишем код обработки опции меню **Помощь → О программе**. Выводим окно сообщения (MsgBox) со следующими параметрами:

заголовок – «О программе»;

тип окна – информационное окно с одной кнопкой «ОК»;

текст в окне: «Программа регистрации транзакций».

Не забудьте, что функция MsgBox возвращает результат целого типа. Поэтому надо описать переменную целого типа. Сохранить проект, проверить результат.

• Создать окно помощи. Для этого создаем новую форму. Определить ее свойства:

Name – frmHelp;

Caption – Информация о функциональности.

Разместите на форме TextBox и задайте ему свойства:

Name – txtHelpBox;

Text – пробел;

Multiline – true;

Scrollbar – vertical.

Чтобы размеры текстовой области совпадали с размерами формы, установить с помощью свойства Anchor привязку по всем четырем сторонам. Сохранить проект.

• Загрузить окно помощи при нажатии опции меню – Описание. Для этого открыть подпрограмму обработки нажатия клавиши и написать код вызова формы:

```
Private Sub mnuDescript_Click(ByVal sender As System.Object,_ By-
Val e As System.EventArgs) Handles mnuDescript.Click
    Dim f_Help As New frmHelp()
    f_Help.MdiParent = Me
    f_Help.Show()
End Sub
```

Сохранить проект, запустить его и проверить загрузку новой формы. Чтобы новая форма выводилась на экран с максимальным размером, установить свойство WindowState в положение maximized. Проверьте результат.

• Вывести в окно файл помощи. Для этого разместить в папке проекта файл help_file.txt. Файл размещен на сервере по адресу

D:\magistr\lab6\. В процедуре, реализующей обработку загрузки формы frmHelp, разместить код работы с тестовым файлом, приведенным в теоретическом разделе.

Чтобы перечисленные в коде объекты были восприняты транслятором, необходимо подключить библиотеку классов, отвечающих за ввод/вывод с помощью команды Imports System.IO. Указанная строка размещается первой в тексте программного кода формы.

Сохранить проект, проверить результат.

• Создать две открывающиеся формы по нажатию опций меню – Транзакция и Просмотр. Задать формам следующие параметры:

Name – frmTrans;

Caption – Определить транзакцию;

Name – frmView;

Caption – Просмотр транзакций.

Вывести на каждую форму по одной кнопки, задать параметры соответственно:

Name – cmdQuit;

Caption – Выход (для обеих).

Запрограммировать закрытие текущей формы по нажатию кнопки. Для формы frmTrans текст выглядит так:

```
Private Sub cmdQuit_Trans_Click()
   Me.Close()
End Sub
```

Для формы frmView – аналогично.

• Написать код загрузки новых форм по нажатию соответствующих клавиш меню. Код аналогичен тому, что использовался при загрузке формы frmHelp.

• Сохранить проект, проверить результат. Продемонстрировать преподавателю работающее приложение. Подписать отчет о выполнении работы.

Дата _____ Подпись преподавателя _____

Лабораторная работа 7
РАБОТА С БАЗАМИ ДАННЫХ В VISUAL BASIC.NET

Цель работы: ознакомить студентов с основными возможностями работы в системе Visual Studio.Net с базами данных. Изучается использование коллекции классов ADO.Net для работы с реляционными базами данных, созданными в SQL Server 2000. Изучаются работы с использованием отсоединенных объектов DataSet и работа с использованием запросов SQL и хранимых процедур, исполняемых непосредственно в базе данных.

Теоретические основы

Механизмы доступа к данным ADO.Net. Microsoft Visual Studio.Net имеет в своем составе новую библиотеку классов для соединения приложения с источником данных. Это коллекция классов ADO.Net. Главное, что отличает этот новый набор от предыдущих объектов доступа данным, заключается в том, что ADO.Net позволяет работать в отрыве от источника данных. Это очень важно при создании Web-приложения. В этом случае приложение считывает данные, формирует объект, содержащий эти данные, и закрывает связь с источником данных. Пользователь работает с этими данными, формирует необходимые изменения, устанавливает контакт с источником данных и передает ему изменения. Такой режим работы позволяет экономить ресурсы сервера. В то же время ADO.Net позволяет работать и в связи с данными.

Второе отличие ADO.Net заключается в том, что при работе в отрыве от источника данных обмен данными осуществляется в формате XML.

Рассмотрим далее основные объекты, входящие в ADO.Net, которые будут использованы в данной работе. В интегрированной среде разработки Visual Studio.Net эти объекты находятся на палитре объектов (Toolbox) в разделе Data.

Поставщики данных ADO.Net. Поставщик данных – это набор классов, предназначенных для взаимодействия с хранилищем данных определенного типа. ADO.Net имеет в своем составе два поставщика: SQL Client.Net Data Provider и OLE DB.Net Data Provider. Первый поставщик используется при работе MS SQL Server, начи-

ная с 7-й версии. Второй поставщик используется при работе с любым источником данных при условии, что в системе установлен поставщик OLE DB требуемого типа. Оба поставщика включают в себя одинаковый набор классов. Далее мы будем описывать объекты поставщика SQL Server.

Connection. Объект SQLConnection реализует соединение с базой данных. Для организации связи формы с базой данных можно перетащить соответствующий объект с палитры на форму.

Главное свойство объекта Connection – ConnectionString (строка подключения). В строке указывается источник данных и необходимые атрибуты для подключения. Формировать строку подключения достаточно сложно, поэтому имеет смысл использовать специальный помощник, который устанавливает соединение при создании следующего объекта – DataAdapter.

DataAdapter. Объект SQLDataAdapter является своеобразным мостом между базой данных и отсоединенным объектом DataSet. С помощью DataAdapter данные перемещаются из базы данных в набор DataSet. Как уже отмечалось выше, Visual Studio.Net имеет специальный помощник для создания соединения и адаптера. Рассмотрим его работу. Перетаскиваем из палитры объектов SQLDataAdapter на форму. Начинает работать помощник.

Прежде всего предлагается установить соединение с данными. Либо выбирается уже установленное ранее, либо нажимаем кнопку New Cjnnection. Открывается окно установления соединения (рис. 7.1).

Так как мы используем объект связи с SQL Server, выбирать поставщика данных не требуется, он определен заранее. Сразу открывается окно Подключение. Выбираем из выпадающих списков SQL сервер и базу данных. Список баз данных появляется после указания существующего сервера. Следует установить тип проверки входа. Мы будем использовать учетные сведения операционной системы. Нажимая кнопку «Проверить подключение», проверяем установление соединения с данными. После подтверждения создается строка подключения и создается объект Connection.

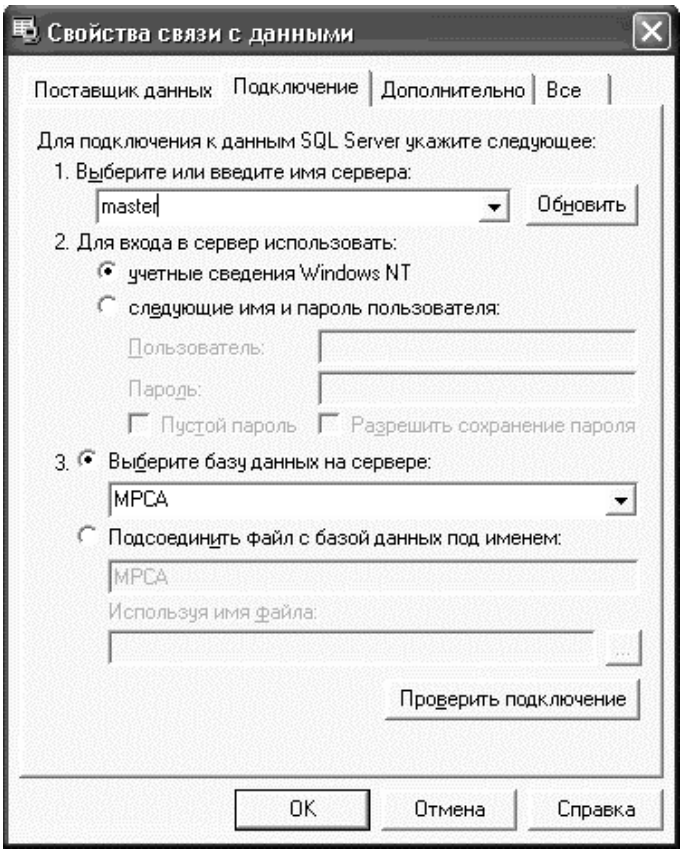

Рис. 7.1. Окно установления соединения с хранилищем данных

Продолжаем создание адаптера. Выбираем тип запроса для создания адаптера данных. Существует три возможности: SQL запрос, новая хранимая процедура и существующая хранимая процедура. Мы будем использовать запрос. Для этого вызываем создатель запросов (кнопка Query Builder). Это средство очень похоже на аналогичную процедуру SQL Server. На рис. 7.2 представлен результат создания запроса.

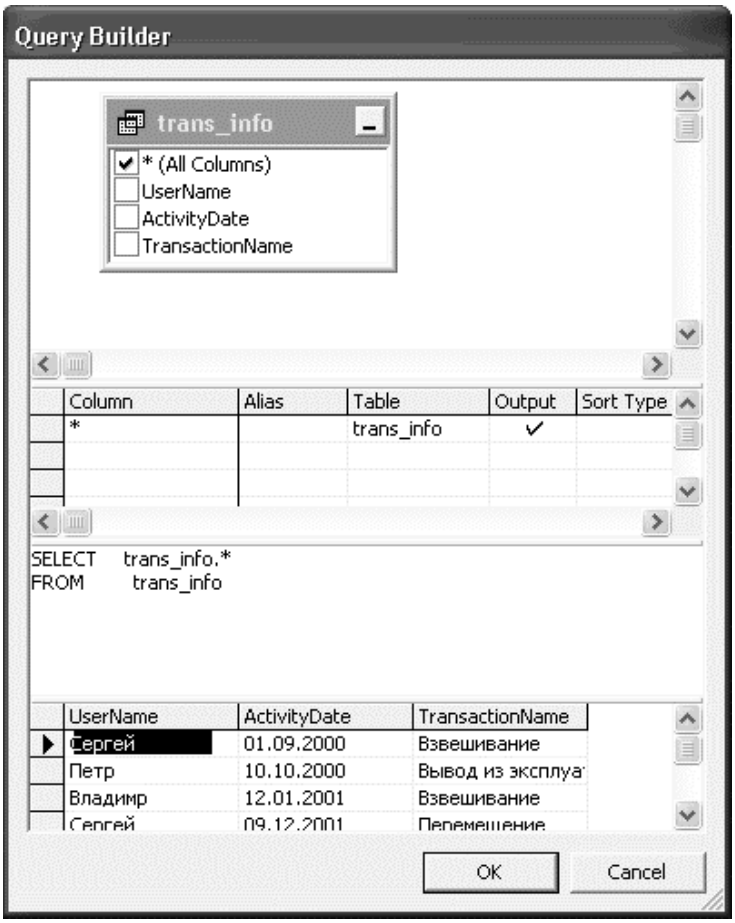

Рис. 7.2. Окно создания запроса для определения DataAdapter

На этом этапе сформировано соединение с хранилищем данных и выбран набор данных, который будет копироваться в отсоединенный набор DataSet.

DataSet. Объект DataSet – это набор данных. Его отличие от наборов данных, используемых в более ранних коллекциях объектов связи с данными, заключается в том, что он способен включать в себя данные с помощью нескольких адаптеров. DataSet по сути –

копия части базы данных. Рассмотрим процедуру создания набора DataSet.

Стартуем средство создания набора данных с помощью главного меню среды разработки **Data → Generate DataSet**. Открывается окно, представленное на рис. 7.3.

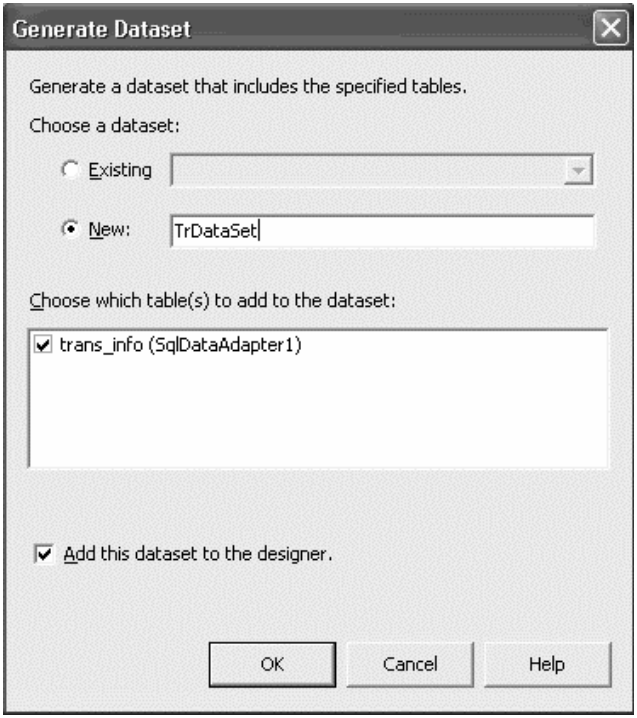

Рис. 7.3. Окно создания объекта DataSet

На этом этапе создано соединение с хранилищем данных, адаптер данных и набор DataSet. Однако следует учесть, что при этом созданный набор DataSet является пустым. Чтобы заполнить его данными, необходимо использовать метод Fill объекта DataAdapter. Ниже приводится фрагмент кода заполнения набора данных.

```
DS_trans1.Clear()
SqlDataAdapter1.Fill(DS_trans1)
```

Первая команда очищает набор DS_trans1 с использованием его метода Clear(). Во второй команде метод Fill() объекта SQLDataAdapter1 заполняет набор данных новыми данными из хранилища данных.

Данный фрагмент текста обычно вставляется либо в текст метода загрузки формы, либо в метод обработки специальной кнопки загрузки информации.

Ни один из перечисленных объектов не имеет визуальных компонент для представления данных при работе приложения. Для этого используются **связанные элеметы управления**. Связанные элементы управления – это объекты управления связанные с данными, например, с набором DataSet.

DataGrid. Элемент управления DataGrid – это специальный объект, служащий для представления данных. Он имеет форму таблицы, столбцы которой соответствуют столбцам ображаемой таблицы базы данных.

Размещаем его на форме и задаем свойство этого объекта DataSource по имени таблицы из набора DataSet, которую мы хотим отобразить на форме. После этого при запуске приложения в объекте DataGrid будет отображаться содержание таблицы, скопированной из базы данных в набор DataSet.

На этом этапе разработки наименования столбцов соответствуют их наименованиям в исходной таблице. Это может быть неудобно, так как при разработке базы данных обычно используются сокращенные условные наименования, набранные латинскими буквами. Таблицы пользовательского приложения должны быть на национальном языке и полностью понятными. Кроме того, возможно желание отформатировать ширину столбцов, шрифт и т.п. Для настройки вида таблицы в окне свойств объекта необходимо установить коллекцию свойств TableStyles. Для этого щелкнуть мышью на кнопке свойства. Открывается окно редактора свойств таблицы, представленное на рис. 7.4.

Нажав кнопку ADD, добавляем описание таблицы. Устанавливаем значение свойства MappingName по имени выводимой таблицы.

Затем переходим к форматированию столбцов таблицы. Для этого устанавливаем коллекцию свойств столбцов – GridCol-

umnStyles. Открывается окно редактора свойств столбцов таблицы, представленное на рис. 7.5.

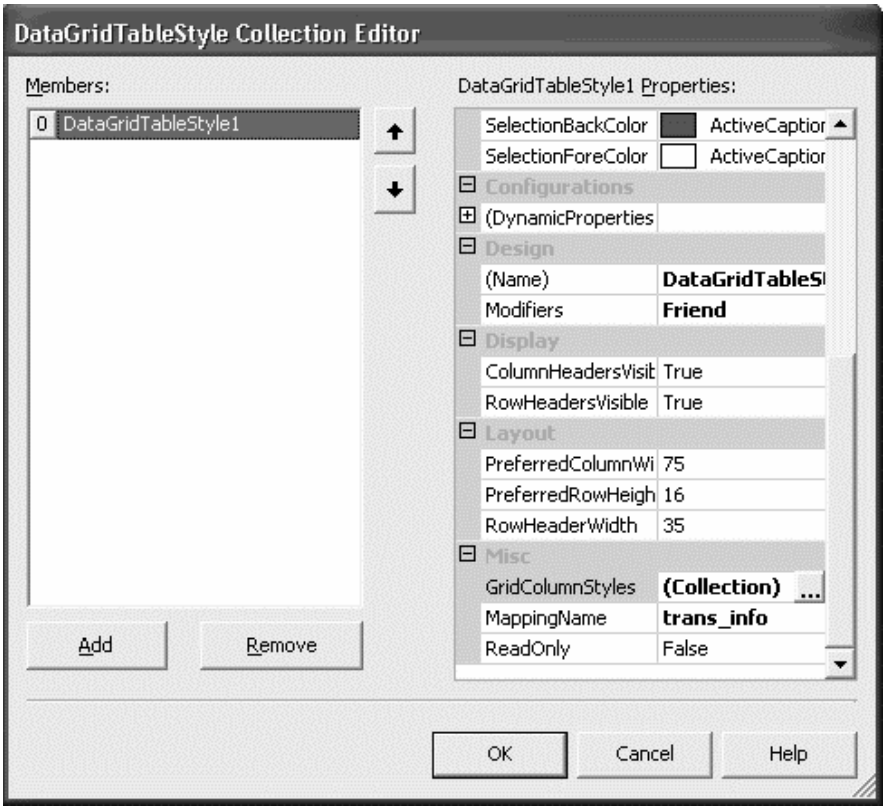

Рис. 7.4. Окно редактора форматирования объекта DataGrid

Нажимая кнопку Add, добавляем три объекта, соответствующие трем столбцам таблицы. Для каждого из них устанавливаем необходимые свойства. Прежде всего в выпадающем списке свойства MappingName выбираем наименование отображаемого столбца. Свойство HeaderText определяет отображаемый заголовок. Свойство Width – ширину колонки. Помимо этих свойств существуют и другие, которые в данной работе не используются.

После установления свойств таблицы форма готова к работе.

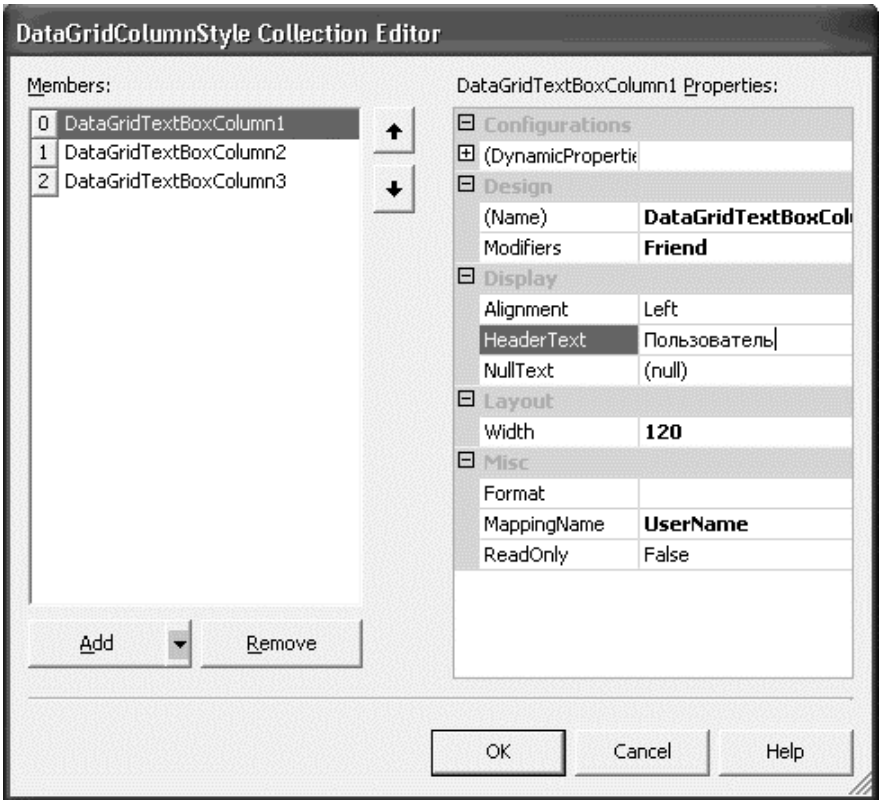

Рис. 7.5. Окно редактора форматирования столбцов DataGrid

ComboBox. Элемент управления ComboBox очень похож на объект ListBox. Отличие между ними заключается в том, что список ComboBox является выпадающим. Кроме того, у ComboBox имеется текстовое окно, в котором отображается значение выбранного элемента списка. Элемент управления ComboBox можно связать с данными. В отличие от элемента управления DataGrid, ComboBox может представлять пользователю значения только одного поля таблицы. Чтобы связать элемент управления с данными, необходимо установить два свойства ComboBox. Свойство DataSource указывает на таблицу соответствующего набора данных DataSet. После установки данного свойства у другого свойства –

DisplayMember появляется выпадающий список, содержащий имена столбцов выбранной таблицы. В этом списке отмечаем требуемый столбец.

У ComboBox существует еще одно свойство, содержащее значения столбца таблицы. Это свойство – ValueMember. Значение этого свойства не отображается в интерфейсе пользователя, но его можно считать программно. При работе с данными в этом поле удобно, например, размещать значение первичного ключа. Тогда, выбрав значение столбца, легко определить первичный ключ выбранной записи. Значение свойства ValueMember выбранного элемента считывается из свойства SelectedValue.

На этом этапе мы научились просматривать информацию из базы данных с помощью различных элементов управления Visual Basic.Net. Вторая задача приложения базы данных заключается в передаче изменений в базу данных. У набора данных DataSet существуют специальные механизмы, позволяющие передавать изменения данных в исходную базу данных. Кроме того, в ADO.Net существует специальный объект **Command**, который способен посылать запросы базе данных, включая запросы вставки и редактирования данных. С помощью этого объекта возможно также передавать параметры хранимой процедуре и посылать ее на выполнение.

Одним из приемов, который повышает защищенность данных в компьютеризированных СУиК ЯМ, является реализация всех функций системы посредством хранимых процедур. В данной работе мы рассмотрим именно такой подход.

Для использования объекта SQLCommand перетаскиваем его с палитры объектов на форму и устанавливаем ряд свойств. Свойству Connection присваиваем значение открытого соединения, например, SqlConnection1. У свойства CommandType выбираем в выпадающем списке значение StoredProcedure. И, наконец, в окне свойства CommandText набираем имя сществующей хранимой процедуры в квадратных скобках – [new_trans]. После задания этих свойств Visual Studio предлагает регенерировать параметры хранимой процедуры. Необходимо подтвердить это действие. После завершения операции, если все было установлено корректно, открыв коллекцию параметров процедуры, можно увидеть окно редактирования параметров (рис. 7.6).

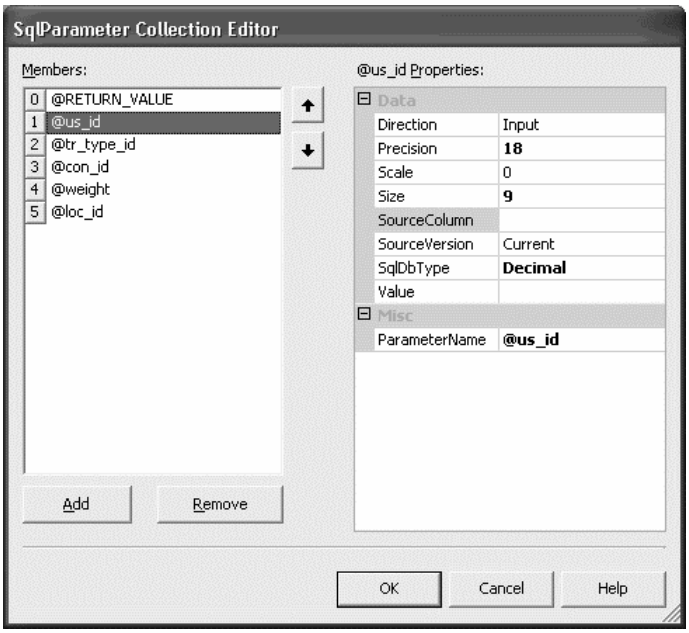

Рис. 7.6. Окно редактирования параметров хранимой процедуры

В тексте программного кода необходимо открыть соединение с базой данных, установить значения параметров, послать хранимую процедуру на выполнение и закрыть соединение. Ниже приводится текст кода, который будет создаваться в текущей работе.

```
SqlConnection1.Open()
SqlCommand1.Parameters("@us_id").Value =_
        ComboBox1.SelectedValue
SqlCommand1.Parameters("@tr_type_id").Value =_
        ComboBox4.SelectedValue
SqlCommand1.Parameters("@con_id").Value =_
        ComboBox2.SelectedValue
SqlCommand1.Parameters("@weight").Value = Val(TextBox1.Text)
SqlCommand1.Parameters("@loc_id").Value =_
        ComboBox2.SelectedValue
SqlCommand1.ExecuteNonQuery()
SqlConnection1.Close()
```

Команда ExecuteNonQuery() посылает процедуру на выполнение.

Описанные выше приемы работы с базами данных из пользовательского приложения не исчерпывают всех возможностей ADO.Net. Интересующиеся программированием баз данных могут найти полное описание классов ADO.Net в литературе.

Выполнение работы

1. Загрузить компьютер. Стартовать Visual Studio.Net и открыть проект, над которым работали на предыдущей лабораторной работе.

2. Создать функцию просмотра информации о транзакциях, собранную в представлении trans_info. Для реализации этой функции выполнить следующие действия:

- Открыть для работы форму frmView. Создать объект ADO.Net – DataAdapter и связать его с представлением trans_info в базе данных, созданной вами ранее в SQL Server. Для этого открываем на палитре объектов ToolBox вкладку Data и перетаскиваем на форму объект SQLDataAdapter. Запускается помощник для создания объекта. В процессе выполнить следующие действия.
  - ♦ Установить новое соединение с базой данных. Для этого на появившейся форме установить следующие свойства:
    - а) провайдер – Microsoft OLE DB Provider for SQL Server;
    - б) соединение – указать имя SQL Server – Master и в выпадающем списке выбрать свою базу данных.
  - ♦ Выбрать тип запроса – Use SQL Statements и сформировать запрос с помощью редактора запросов. В редакторе запросов выбрать нужное представление, отметить все поля (три), которые будут выводиться в набор данных.
  - ♦ Создаем новый набор DataSet. Для этого стартуем меню Data → Generate Dataset. Ввести имя набора DS_trans.
- Разместить на форме элемент управления DBGrid. Связать его с объектом DataSet и настроить его свойства, для этого установить свойство DataSource по имени объекта DataSet – DS_trans1.trans_inf.
- В свойстве TableStyles щелкнуть на кнопке открытия формы установления свойств. Нажав кнопку ADD – добавить описание таблицы. Установит свойство MappingName – trans_info.
- Открыть свойства GridColumnStyles и добавить описания свойств трех столбцов.

• Для каждого из них установить три свойства:
HeaderText – “Название столбца по-русски”
Width – 120 (ширина столбца)
MappingName – “Название столбца в базе данных”.
• Закрыть окно. Сохранить проект.
• Создать кнопку загрузки данных в набор DataSet. Для этого разместить на форме новую кнопку. Определить параметры:
Имя – cmdLoad,
Надпись – Загрузить.
• Написать код обработки нажатия клавиши. Он приведен ниже:

```
Private Sub cmdLoad_Click(ByVal sender As System.Object,_ ByVal e
As System.EventArgs) Handles cmdLoad.Click
    DS_trans1.Clear()
    SqlDataAdapter1.Fill(DS_trans1)
End Sub
```

• Сохранить проект. Протестировать приложение. Продемонстрировать работу преподавателю. Результат представлен на рис. 7.7.

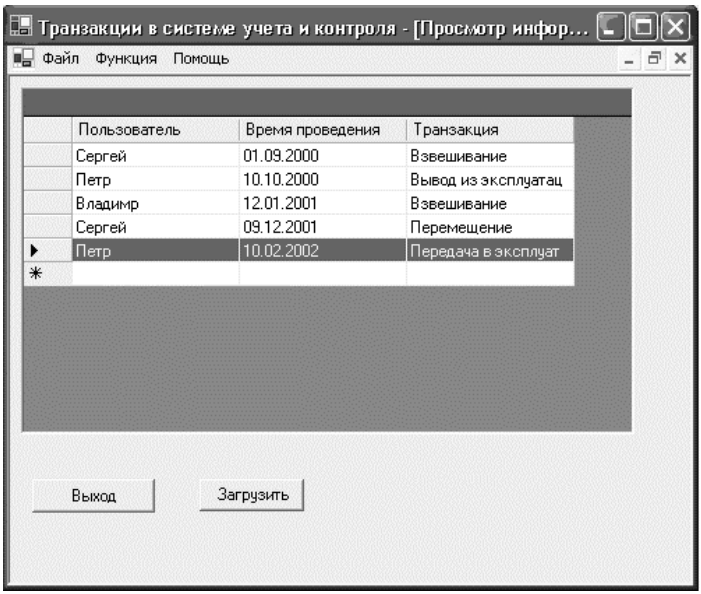

Рис. 7.7. Окно просмотра транзакций

• Создаем функцию формирования информации о транзакции и ее ввода в базу данных. Для этого:

• Открываем для работы форму frmTrans. Размещаем на форме 6 объектов label, 4 объекта ComboBox, 1 объект –TextBox и еще одну кнопку. Установить свойства объектов, которые видны на рис. 7.8. Сохранить проект.

• Создаем соединение объектов с базой данных. Для этого размещаем на форме объект SQLConnection. В окне свойств выбираем свойство ConntctionString из существующих. (Строка соединения уже есть, так как форма нашего проекта уже была связана с нужной базой данных). Затем генерируем 4 объекта DataAdapter, связывая их с таблицами Users, Containers, Locations и Transaction_Type. Процедура генерации адаптеров данных аналогична тому, что делалось на предыдущей форме. Наконец создаем набор DataSet, содержащий все таблицы имеющий имя DB_data.

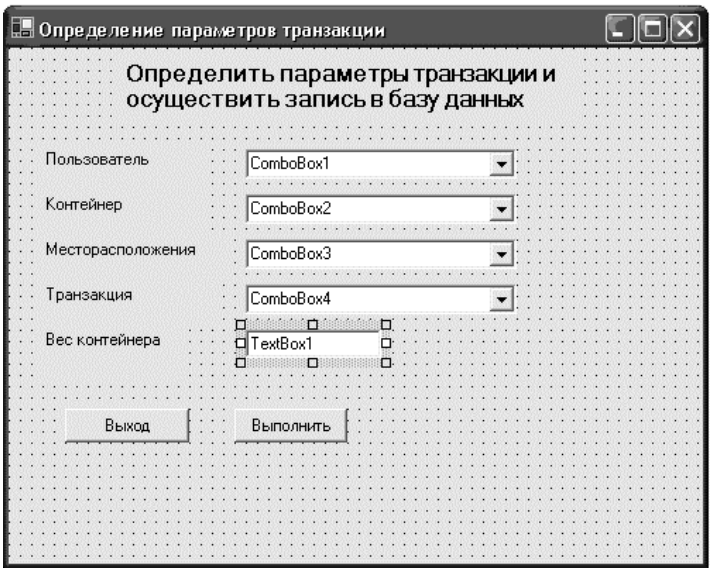

Рис. 7.8. Вид окна записи транзакции

• Связываем объекты ComboBox с таблицами из набора данных.
Объект ComboBox1:
DataSource – DB_data1.users (определяем таблицу),
DisplayMember – user_name (определяем поле, выводимое в списке),
ValueMember – user_ID.
Объект ComboBox2:
DataSource –DB_data1.containers,
DisplayMember – con_name,
ValueMember - con_id.
Объект ComboBox3:
DataSource – DB_data.locations,
DisplayMember – location_name,
ValueMember - location_id.
Объект ComboBox3:
DataSource – DB_data1.transaction_types,
DisplayMember – trans_type,
ValueMember – trans_type_id.
Объект TextBox1.Text="100.0".

Сохранить проект.

• Пишем код заполнения наборов данных. Помещаем код в подпрограмму, обрабатывающую событие загрузки формы frmTrans. Текст подпрограммы приведен ниже.

```
Private Sub frmTrans_Load(ByVal sender As System.Object,_ ByVal e As System.EventArgs) Handles MyBase.Load
    DB_data1.Clear()
    SqlDataAdapter1.Fill(DB_data1, "Users")
    SqlDataAdapter2.Fill(DB_data1, "containers")
    SqlDataAdapter3.Fill(DB_data1, "locations")
    SqlDataAdapter4.Fill(DB_data1, "Transactiontypes")
End Sub
```

Сохраняем проект, запускаем программу на выполнение. Демонстрируем преподавателю, что соединение с данными есть, и объекты позволяют устанавливать параметры транзакции.

• Программируем запуск хранимой процедуры, обеспечивающей запись информации о транзакции. Для этого:

1. Генерируем объект SQLCommand. Для этого из ToolBox перемещаем объект SQLCommand и устанавливаем его свойства:

Connection – SqlConnection1
CommandType – StoredProcedure
CommandText – [new_trans]

2. Пишем текст программного кода. Он преобразует строковую переменную веса в числовую, присваивает параметрам хранимой процедуры значения, выбранные в комбинированных боксах и посылает процедуру на выполнение. Текст приводится ниже.

```
Private Sub cmdDone_Click(ByVal sender As System.Object,_ ByVal e
As System.EventArgs) Handles cmdDone.Click
    Dim weight As Single
    weight = Val(TextBox1.Text)
    SqlConnection1.Open()
    SqlCommand1.Parameters("@us_id").Value =_
        ComboBox1.SelectedValue
    SqlCommand1.Parameters("@tr_type_id").Value =_
        ComboBox4.SelectedValue
    SqlCommand1.Parameters("@con_id").Value =_
        ComboBox2.SelectedValue
    SqlCommand1.Parameters("@weight").Value = weight
    SqlCommand1.Parameters("@loc_id").Value =_
        ComboBox2.SelectedValue
    SqlCommand1.ExecuteNonQuery()
    SqlConnection1.Close()
End Sub
```

- Сохранить проект. Стартовать приложение. Продемонстрировать работу приложения преподавателю.

5. Завершить работу. Получить подпись преподавателя, выключить компьютер.

Дата _____ Подпись преподавателя _____

Лабораторная работа 8
СОЗДАНИЕ И ПУБЛИКАЦИЯ WEB-ДОКУМЕНТОВ

Цель работы: научить студентов основным навыкам создания www-страниц с использованием специального языка гипертекстовых документов HTML, работы с броузером MS Internet Explorer. Кроме того, изучается публикация документов с использованием MS Internet Information Services.

Теоретические основы

В настоящее время мы являемся свидетелями настоящей революции в информационных технологиях. Наличие новых технологий, и прежде всего технологии www-страниц, позволяет передавать информацию пользователю в любой точке земного шара.

Эти новые технологии могут быть использованы при создании информационно-поисковых и учетных систем. Для этого помимо технологий создания и сопровождения баз данных и технологий программирования интерфейсов требуются знания технологии создания www-документов.

Программа-навигатор – MS Internet Explorer. Для просмотра информации в глобальной сети Internet на компьютере-клиенте должно быть установлено единственное программное средство – программа навигатор (броузер). Наибольшей популярностью в настоящее время пользуются навигаторы двух фирм: Netscape и MS Internet Explorer. Причем продукция Microsoft используется более чем на 70% компьютеров, включенных в сеть. Рассмотрим навигатор MS Internet Explorer. На рис. 8.1 представлено открытое окно навигатора.

В окне видны кнопки, которые позволяют осуществлять навигацию в Internet. Одни из основных кнопок – «Вперед» и «Назад». Нажимая эти кнопки, можно переходить по соединениям между открытыми документами. Кнопка «Обновить» выводит новое содержимое динамической страницы. «Домой» – возвращает пользователя на домашнюю страницу. Следует отметить, что броузер позволяет просматривать не только www-документы, но и файлы.

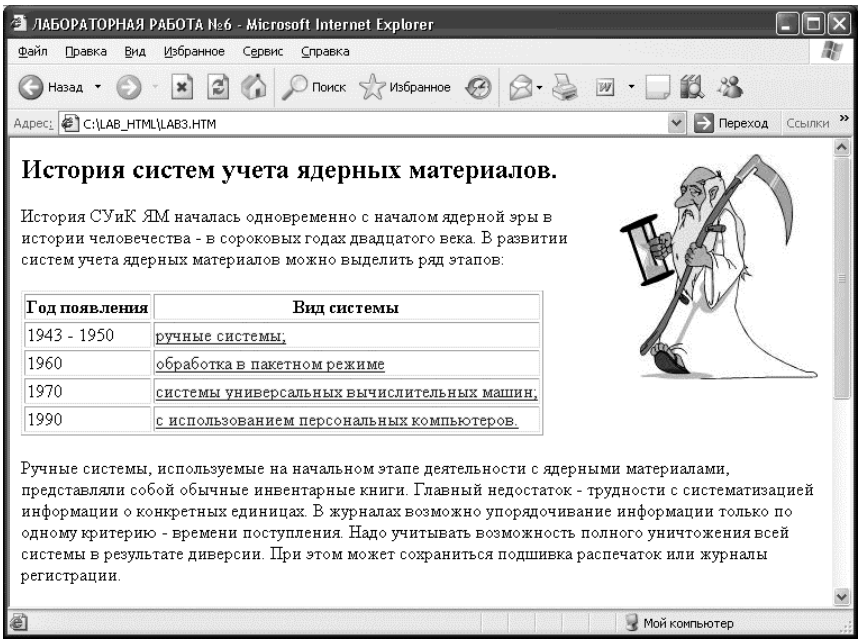

Рис. 8.1. Окно навигатора MS Internet Explorer

Язык создания гипертекстовых документов – HTML. В 1989 г. в ЦЕРНЕ был разработан метод передачи по компьютерной сети гипертекстовых документов и данных различных типов – изображений, аудио- и видеоинформации.

Термин «гипертекст» означает, что в тексте документа содержатся выделенные участки (слова, группы слов или изображения), воздействуя на которые пользователь переходит к другому участку документа или к другому документу. Для реализации этих возможностей был создан язык разметки документов – Hiper Text Marked Language (HTML).

Документ разбивается на структурные компоненты: заголовок, параграф, список и т.д., определяемые специальными конструкциями, называемые тегами. Большинство тегов парные – открывающий и закрывающий. Ниже приводится текст, содержащий базовую структуру документа на языке HTML.

<HTML><HEAD>
<TITLE>ЛАБОРАТОРНАЯ РАБОТА №6</TITLE></HEAD>
<BODY>
<H1>ИСПОЛЬЗОВАНИЕ ЯЗЫКА HTML ДЛЯ СОЗДАНИЯ WEB СТРАНИЦЫ</H1>
</BODY>
</HTML>

Каждый документ содержит парные теги <HTML> и </HTML>. Теги <HEAD> и </HEAD> ограничивают область заголовка. Предложение, заключенное между тегами <TITLE> и </TITLE>, выводится в строке заголовка броузера. Далее следует основное тело документа, заключенное между тегами <BODY> и </BODY>.

В теле документа используется ряд конструкций. Прежде всего теги, определяющие заголовки <H1> и </H1>. Номер рядом с буквой H обозначает уровень заголовка, соответствующим образом форматированный. Теги <P> и </P> заключают между собой выделенный параграф.

Списки. Важным элементом документов являются списки. Представим несколько возможных вариантов. Первый тип списков – нумерованные списки – представляются следующим набором тегов.

<OL>
<LI> Первый элемент,
<LI> Второй элемент,
<LI> Третий элемент.
</OL>

Если в открывающий список тег добавить опцию <OL TYPE="I">, то нумерация будет осуществлена римскими цифрами.

Ненумерованные списки определяются следующими тегами:

<UL>
<LI> Первый элемент,
<LI> Второй элемент,
<LI> Третий элемент.
</UL>

Для выделения строк могут использоваться элементы различной формы (диск, квадрат, кружок). Соответственно в открывающий список тег надо вставлять TYPE= : “disc”, “square”, “circle”.

Элементы форматирования и оформления страницы. Существуют специальные теги, позволяющие оформить страницу и сделать ее нарядной. Рассмотрим несколько возможностей.

Включение рисунка в документ достигается тегом <IMG SCR=“полное имя файла с рисунком”>. Под полным именем понимается путь к файлу, включающий драйвер и директории. Если поместить только имя файла, то броузер будет искать его в директории, откуда загружена страница. Возможно подключение рисунков в форматах GIF и JPG. Рисунок выгружается в том месте, где помещен тег.

Чтобы поместить рисунок в качестве фона страницы, в тег, обозначающий начало тела страницы, помещается следующее выражение. <BODY BACKGROUND=“полное имя файла с рисунком”>.

Для изменения используемого фонта в заголовках или параграфах используется тег: <FONT COLOR=blue SIZE=5> и <FONT>. Текст, заключенный между этими тегами, будет иметь особый цвет и размер. Кроме того, текст можно менять следующими тегами: <STRONG> … </STRONG> – отображает текст жирными буквами; <EM> … </EM> – курсив; <U> … </U> – подчеркивание.

Специальный атрибут ALIGN, размещаемый в теге, позволяет выравнивать расположение заголовков, текста или рисунка. Он может приравниваться значениям: ALIGN= CENTER, LEFT, RIGHT.

Построение таблицы. Пример построения таблицы размерами 2 ячейки на 2 приведен в следующем фрагменте:

```
<TABLE BORDER>
<TR>
<TH>Заголовок 1-го столбца</TH>
<TH>Заголовок 2-го столбца</TH>
</TR>
<TR>
<TD>Содержимое 1-й строки 1-го столбца</TD>
<TD>Содержимое 1-й строки 2-го столбца </TD>
</TR>
```

<TR>
<TD>Содержимое 2-й строки 1-го столбца</TD>
<TD>Содержимое 2-й строки 2-го столбца </TD>
</TR>
</TABLE>

Создание гипертекстовых ссылок. Гипертекстовые ссылки в документе могут быть двух типов: первая – вызывает следующий документ, другие ссылаются на раздел текущего документа. Для организации ссылки на другую страницу используется тег: <A HREF="полный путь к файлу"> выделенный текст </A>. В тексте документа появляется выделенный участок (это может быть и рисунок). При наведении на эту область курсора мышки, он меняет свою форму. При нажатии клавиши загружается требуемый документ.

Для ссылки на часть документа необходимо поставить в этом месте «якорь». Это делается с помощью тегов: <A NAME="имя"> текст</A>. В месте ссылки используется конструкция <A HREF=#имя> выделенный текст </A>. Такие ссылки удобно располагать в больших документах.

Публикация www-документа в Internet. Перечисленные выше структуры гипертекстового документа не исчерпывают многообразия возможностей, даваемых современными версиями HTML. С некоторыми из них мы познакомимся на следующей лабораторной работе.

Сейчас рассмотрим процесс публикации документа, который сводится к его регистрации на сервере, являющемся узлом сети. Чтобы документ был доступен, на сервере должно быть установлено специальное программное обеспечение – Web server. В составе операционных систем семейства Windows NT Microsoft выпускает Internet Information Services. При установке сервера в свойствах всех папок появляется новая закладка Web sharing, определяющая возможность доступа к ней через Internet.

После того как на сервере будет создана директория с набором необходимых html-файлов и файлов, содержащих другие ресурсы документа (видео, аудио, рисунки и т.д.), необходимо разрешить доступ к ней по сети. При этом определяется имя, по которому будут обращаться к директории и первый файл, который будет загружаться при этом. Если разрешенная для доступа директория

помечается, как «домашняя», то при обращении к этому серверу, она будет загружаться автоматически.

Выполнение работы

1. Включить компьютер и загрузить систему.

2. Используя стандартный текстовый редактор «Блокнот», создать первую страницу web-документа. Для этого:

- Открыть файл **text1.txt**, содержимое которого приведено ниже.

Начало текста.

```
<HTML><HEAD>
<TITLE> </TITLE></HEAD>
<BODY>
<H1></H1>
<P></P>
<OL>
<LI>
</OL>
</BODY>
</HTML>
```

ЛАБОРАТОРНАЯ РАБОТА
ВВЕДЕНИЕ В КОМПЮТЕРИЗИРОВАННЫЕ СУиК ЯМ.
Настоящий курс представляет собой обзор систем и программного обеспечения, необходимых при создании и эксплуатации автоматизированных систем учета и контроля ядерных материалов.
Согласно этим целям строится содержание курса. На вводной лекции мы рассмотрим следующие вопросы:
1) требования, накладываемые отраслевым стандартом РФ на СУиК ЯМ;
2) компоненты компьютеризированных СУиК ЯМ;
3) историю развития СУиК ЯМ.
По всем пунктам предусмотрено проведение практических работ в лаборатории кафедры, где построена модель системы учета и контроля. В завершении курса сдается экзамен.
После изучения курса вы должны представлять себе все стороны работы по созданию СУиК, требования к ним и обладать навыком практического общения с современным ПО, используемsv в этих системах.

Конец текста.

• Используя текст, сформировать первую страницу web-документа. Заголовок должен быть виден в верхнем поле окна броузера. «Введение …» – является заголовком первого уровня. Все абзацы должны быть выделены как отдельные параграфы. Необходим нумерованный список. Сохранить полученный документ под названием «default.htm».

• Стартовать MS Internet Explorer и проверить вид документа.

3. Создать вторую страницу документа. Для этого:

• Открыть файл **text2.txt.** Содержимое файла приведено ниже.

Начало текста.

```
<HTML><HEAD>
<TITLE> </TITLE></HEAD>
<BODY>
<H2></H2>
<P></P>
<UL>
<LI>
</UL>
</BODY>
</HTML>
```

ЛАБОРАТОРНАЯ РАБОТА

Требования к системам учета и контроля ядерных материалов.

Назначение систем учета материалов и требования, предъявляемые к ним, собраны в отраслевом стандарте ОСТ 95 10537-97 Минатома Российской Федерации.

Под СУиК стандарт понимает систему, в которой неразрывно связан проводимый учет ЯМ, фиксируемый на бумажных носителях, и учет ЯМ компьютеризированный.

Назначение СУиК стандарт определяет следующим образом:

• СУиК осуществляет информационное обеспечение учета и контроля ядерных материалов (ЯМ), имеющихся на предприятии, поступающих, убывающих с него и уничтожаемых на основе правовой ответственности предприятия.

• Учитывает любое перемещение материалов между зонами баланса материалов (ЗБМ), любое изменение вида и формы материала.

• Предоставляет своевременную и достоверную информацию для осуществления деятельности по учету и контролю.

Для реализации этих положений в каждой ЗБМ и на предприятии в целом СУиК должна обеспечивать:

▪ информационную запись всех измерений в ключевых точках по количеству и иным характеристикам ЯМ;

▪ информационное сопровождение любых изменений в виде, форме и месту положения ядерных материалов, включая внешние операции (отправка/получение);

▪ определение количества материала в каждой ЗБМ и предприятии в целом;

▪ ведение учетной и отчетной документации;

Особые требования налагаются на безопасность информации. Причем принимаются во внимания оба аспекта – сохранность информации при аварии и сбоях системы и предохранение от несанкционированного доступа. При этом накладываются требования:

◊ создания и регулярного обновления резервных копий баз данных;

◊ хранения истории каждой учетной единицы, прошедшей через предприятия, для возможного восстановления последовательности действий;

◊ ведения раздельного учета ЯМ для мирных и оборонных целей.

Конец текста.

• Используя текст, сформировать вторую страницу web-документа. Заголовок «Лаборатор …..» должен быть виден в верхнем поле окна броузера. «Требования к …» – является заголовком второго уровня. Все абзацы должны быть выделены как отдельные параграфы. Создать три ненумерованных списка. Используя атрибут TYPE="", выделить ненумерованный список различным образом: жирной точкой, квадратиком и кругом. Сохранить полученный документ под названием «suik1.htm». Проверить результат.

• Создать для второго документа фон с использованием атрибута BACKGROUND в нужном теге. Изображение загружать из файла globe.gif.

4. На базе файла text3.txt. Создать третью страницу документа.

Начало документа.

<HTML><HEAD>
<TITLE> </TITLE></HEAD>
<BODY>
<H2> </H2>

<P></P>
<UL>
<LI>
</UL>
</BODY>
</HTML>
ЛАБОРАТОРНАЯ РАБОТА
Компоненты компьютеризированных СУиК
Отраслевой стандарт определяет компьютеризированную СУиК как программно-аппаратную составляющую соответствующей системы на предприятии. Программно-аппаратное оснащение состоит из программного обеспечения и средств вычислительной техники.
Программное обеспечение стандарт разделяет на три компонента:
• операционная система на примере Windows XP;
• система управления базами данных на примере MS SQL Server 2000;
• инструментальные средства программирования на примере Visual Basic.Net.
Последние используются в основном для написания пользовательских интерфейсов и реализации некоторых функций администрирования и защиты.
Аппаратное оснащение разделяется на две группы – компьютерное и сетевое оборудование. После того, как мы разобрали основные понятия и требования стандарта к компьютеризированным СУиК, перейдем к истории развития такого рода систем, рассмотрим основные технические решения и архитектуры, используемые для создания систем в различные периоды, и разберем состояние СУиК ЯМ в настоящее время.
Конец текста.

• Используя текст, сформировать третью страницу web-документа. Заголовок «Лаборатор …..» должен быть виден в верхнем поле окна броузера. «Компоненты …» – является заголовком второго уровня. Все абзацы должны быть выделены как отдельные параграфы. Создать ненумерованный список.

• Используя атрибут ALIGN=, разместить заголовок по центру и с помощью тега <FONT COLOR=…> </FONT> выделить заголовок цветом.

• В списке выделить названия программных продуктов – жирным текстом, курсивом, подчеркиванием. Сохранить полученный документ под названием «suik2.htm». Проверить результат в броузере.

5. На базе файла **text4.txt** создать четвертую страницу документа.

Начало документа.

```
<HTML><HEAD>
<TITLE> </TITLE></HEAD>
<BODY>
<H2> </H2><P></P>
<TABLE BORDER>
<TR>
  <TH> </TH>
  <TH> </TH>
</TR>
<TR>
 <TD></TD>
 <TD></TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

ЛАБОРАТОРНАЯ РАБОТА

История систем учета ядерных материалов.

История СУиК ЯМ началась одновременно с началом ядерной эры в истории человечества – в сороковых годах двадцатого века. В развитии систем учета ядерных материалов можно выделить ряд этапов:

Год появления	Вид системы
1943 – 1950	Ручные системы
1960	Обработка в пакетном режиме
1970	Системы универсальных вычислительных машин
1980	С использование персональных компьютеров

Ручные системы, используемые на начальном этапе деятельности с ядерными материалами, представляли собой обычные инвентарные книги. Главный недостаток – трудности с систематизацией инфор-

мации о конкретных единицах. В журналах возможно упорядочивание информации только по одному критерию – времени поступления. Надо учитывать возможность полного уничтожения всей системы в результате диверсии. При этом может сохраниться подшивка распечаток или журналы регистрации.

Системы, основанные на обработке компьютеризированных пакетов, создавались на первом этапе использования ЭВМ, когда не были разработаны системы интерактивного ввода информации. Главные характеристики таких пакетов – примитивные базы данных и крайне медленное обновление информации. Все это приводило к тому, что системы были медленны и ненадежны.

Существенный прорыв в использовании информационных технологий для развития систем учета и контроля произошел, когда были созданы системы универсальных электронно-вычислительных машин, связанных с терминалами на пользовательских местах. К этому времени появились первые СУБД. Сочетание достаточно мощных ЭВМ с преимуществами интерактивного ввода информации привели к созданию эффективных систем, которые до сих пор используются на некоторых предприятиях.

Появление персональных компьютеров изменило все человеческие технологии. Их использование в системах учета может быть организовано различным образом. Практически все современные системы создаются для использования на персональных компьютерах

Конец текста.

- Используя текст, сформировать четвертую страницу web-документа. Заголовок «Лаборатор …..» должен быть виден в верхнем поле окна броузера. «История …» – является заголовком второго уровня. Все абзацы должны быть выделены как отдельные параграфы. Создать таблицу.
- Используя тег <IMG SRC="oldtime.jpg" ALIGN=RIGHT>, разместить рядом с заголовком рисунок, сдвинутый вправо.
- Организовать ссылку внутри файла на определенные абзацы текста. Для этого в четырех последних абзацах текста поставить «якорь» с помощью тегов: <A NAME="имя">…</A>. В таблице в столбце «Вид системы» поставить ссылку <A HREF=#имя>…</A>.
- Сохранить полученный документ под названием «suik4.htm». Проверить результат в броузере.

6. Организовать связи между страницами. Для этого в конце файлов 2, 3 и 4 страниц поместить ссылку на первую страницу с помощью тегов: <A HREF="default.htm"> Здесь – возврат на первую страницу. </A>. Затем в первой странице организовать ссылки на 2, 3, 4 страницу. Это организуется с помощью тегов <A HREF="suik_.htm"> … </A>, размещенных в предложениях нумерованного списка. Сохранить файлы, продемонстрировать работу преподавателю.

7. Организовать публикацию документа. Для этого:

- Разместить документ на сервере. Получить разрешение у преподавателя для администрирования сервера.
- Разрешить доступ к своей директории через Internet. Задать псевдоним. Определить файл, вызываемый по умолчанию.
- Проверить факт возможности просмотра вашего файла через Internet.
- Продемонстрировать работу преподавателю, получить подпись.

Дата _____ Подпись преподавателя _____

Лабораторная работа 9
ТЕХНОЛОГИЯ ASP.NET СОЗДАНИЯ WEB-ПРИЛОЖЕНИЙ БАЗ ДАННЫХ

Цель работы: ознакомить студентов с процессом создания Web-приложений с использованием технологии ASP.Net. Дать понятие о сетевых элементах управления и программировании методов обработки событий. Продемонстрировать использование ADO.Net для доступа к данным SQL Server 2000 из Web-приложения.

Теоретические основы

ASP – технология создания Web-приложений. Главный недостаток гипертекстовых документов, созданных с использованием языка HTML, заключается в их статичности. До пользователя доносится только та информация, которая изначально присутствует в документе. Никакая обработка информации по желанию пользователя невозможна. Все развитие Web-технологий было направлено на ликвидацию этого недостатка.

Для создания полноценных приложений, функционирующих через Интернет, Microsoft создала технологию ASP – Active Server Pages. Ее суть заключается в соединении www-страницы, написанной на языке HTML, с программными элементами, написанными на языках сценариев VBScript или JavaScript. В текст HTML вводятся специальные теги, которые интерпретируются броузерами, как элементы управления: кнопки, текстовые бланки, выпадающие списки и некоторые другие. С этими элементами управления связаны определенные события, обработка которых осуществляется при обращении к подпрограммам, написанным на языках сценария и размещенных в тексте между соответствующими тегами.

Программист может определять, где будет осуществляться обработка этих событий – на компьютере-клиенте или на сервере. При обработке программного кода на сервере программа может формировать www-страницу в соответствии с информацией, полученной от пользователя. Была разработана специальная библиотека классов для связи с базами данных из Web-приложения – ADO.

Главный недостаток технологии ASP заключается в том, что она по существу является шагом назад в отношении объектно-ориентированного программирования. Программист должен владеть двумя технологиями – технологией создания Web-документов и искусством программирования на языке высокого уровня. Достаточно узка палитра элементов управления. Кроме того, VBScript довольно существенно отличается от Visual Basic. Существуют и другие проблемы, ограничивающие функциональность Web-приложений.

ASP.Net – технология. ASP.Net является дальнейшим развитием технологии создания приложений, функционирующих через сети Интернет/Интранет. Она решает ряд обозначенных выше проблем.

Прежде всего и язык программирования VB.Net, и среда разработки приложения Visual Studio.Net используются при создании как Web, так и Windows приложений. При этом полностью используются объектно-ориентированные свойства языка программирования.

Во-вторых, полностью разделено программирование элементов управления и создание Web-страницы. При этом новая технология поддерживает и старые подходы, то есть будут понятны и конструкции с использованием в тексте HTML элементов кода на VBScript.

Помимо элементов управления HTML, ASP.Net использует новую группу элементов – сетевые элементы управления. Их значительно больше, и обработка событий, связанных с ними, осуществляется на сервере. Проектирование интерфейса и программирование событий становится полностью аналогичным тому, как это осуществляется при создании Windows приложений. Сетевые элементы управления размещаются на форме, определяются их свойства, программируются события, связанные с этими элементами.

Существует специфика Web-приложений, связанная с передачей данных между страницами приложения, но в данной работе эти проблемы рассматриваться не будут.

Код, управляющий Web-приложением, располагается на сервере и активируется каждый раз, когда клиент передает страницу на сервер. Web-приложение считывает данные в форме, обрабатывает их и отвечает на клиентский запрос, передавая другую страницу.

Сетевые элементы управления ASP.Net. На рис. 9.1 видно, что визуально сетевые элементы на проекте формы отличаются тем, что они отмечены специальным значком. Во всем остальном визуально они идентичны. Кроме того, сетевые элементы имеют дополнительные свойства, которыми можно легко управлять – например, вид шрифта для надписи у кнопки. Рассмотрим два простейших элемента управления, которые будут использованы в работе.

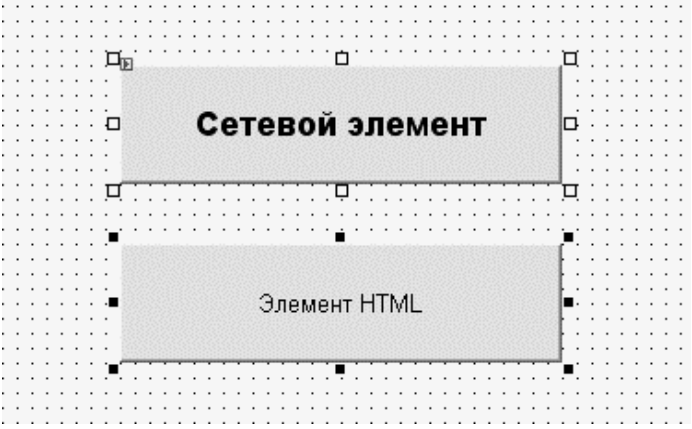

Рис. 9.1. Элементы управления ASP.Net: сетевой и HTML

Web-форма (Webform). Элемент, аналогичный форме Windows приложения. Является контейнером для всех остальных объектов. Основные свойства позволяют управлять внешним видом документа. **Title** – заголовок в окне броузера. **bgColor** – цвет фона. По сути все эти свойства соответствуют атрибутам документа HTML.

Метка (Label). Этот элемент полностью аналогичен по функциям и виду соответствующему элементу управления Windows. Несколько отличаются свойства элемента. Имя элемента обозначается свойством **ID**. Свойство **Text** содержит текст, который отображается на объекте. Существует возможность выбора параметров шрифта текста. Для этого раскрывается коллекция свойств **Font**. Свойство **Bold** принимает значение истинно/ложно и, соответственно, делает шрифт жирным или обычным. Свойство **Name** позволяет выбирать в выпадающем списке тип шрифта. Свойство **Size** – его размер.

Кнопка (Button). Данный элемент также аналогичен элементу управления Windows. Перечислим основные свойства. Имя (идентификатор) объекта – **ID**. Надпись на объекте – **Text**. И параметры используемого шрифта – коллекция свойств **Font**.

Основное событие – нажатие кнопки. Обрабатывается методом **Button_Click**. Написание подпрограммы обработки события осуществляется так же, как и при создании интерфейса Windows приложения: два раза щелкаем мышкой на объекте – открывается заготовка подпрограммы. Достаточно написать требуемый код.

Элементы управления HTML. При создании приложения ASP.Net не рекомендуется использовать элементы управления HTML. Тем не менее, в данной работе используется единственный элемент управления **Кнопка**. Это делается для того, чтобы подчеркнуть различия между технологиями ASP и ASP.Net. Кроме того, функции вызова другой страницы и закрытия окна броузера легко реализуются на клиентском компьютере без обращения к серверу.

Основные свойства кнопки HTML. Идентификатор объекта – **ID**. Надпись на объекте – **Value**. Событие нажатия кнопки обрабатывается методом **Button_onClick**. Отличия в синтаксисе вызвано тем, что в случае сетевого элемента управления текст метода писался на языке VB.Net, а в случае элемента HTML на VBScript.

Чтобы написать подпрограмму обработки события нажатия кнопки, необходимо перейти от окна дизайна страницы к содержимому страницы HTML. Откроем требуемую закладку в левом нижнем углу окна. И в теле текста HTML разместим следующий скрипт:

```
<script language="vbscript">
 sub cmdQuit_onClick
    Window.close
end sub
</script>
```

В данном случае при нажатии кнопки с ID = cmdQuit выполняется команда закрытия окна броузера. Обратите внимание на различия при создании методов обработки одного и того же действия при использовании кнопок двух различных типов.

Средства доступа к базам данных. Приложения, созданные с использованием ASP.Net, работают с той же коллекцией объектов доступа к данным, что и Windows приложения – ADO.Net. Мы будем использовать те же самые объекты: SQLConnection, SQLDataAdapter и DataSet. Именно в Web-приложениях существенным является возможность работать с данными, отсоединенными от их источника.

Так же, как и в случае Windows приложения, пользователю предоставляется ряд элементов управления, связанных с источниками данных. В данной работе мы будем использовать два типа таких объектов.

DropDownList. Этот элемент управления по виду и функциональным возможностям аналогичен элементу управления ComboBox, который мы использовали при создании Windows приложения. При связывании с данными, выпадающий список объекта заполняется значениями выбранного поля таблицы. Пользователь может выбирать строку в списке, которая затем высвечивается в текстовом поле.

Для связи элемента управления с данными необходимо установить значения следующих его свойств.

Свойство DataSource указывает на требуемый набор DataSet. Свойство DataMember указывает на имя нужной таблицы, содержащейся в выбранном наборе данных. Наконец, свойство DataTextField указывает на название поля таблицы, значения которого будут находиться в выпадающем списке. Для того чтобы легко определить дополнительные параметры выбранной записи, элемент управления имеет свойство DataValueField. Это свойство можно заполнить, например, значениями ключевого поля. Тогда, выбрав требуемую запись, легко можно определить значение первичного ключа этой записи.

DataGrid. Данный элемент управления тождественен соответствующему элементу управления для Windows приложений. Visual Studio.Net предоставляет пользователю специальное средство для автоматического создания Web-формы, на которой будет отображена таблица данных. Для этого достаточно в главном меню выбрать опцию Project → Add Web form. Затем выбрать тип создаваемой формы – Data Form Wizard. Начинает работать помощник для создания формы. Пользователь последовательно определяет пара-

метры источника данных. Автоматически создаются объекты связи с данными и готовая таблица. Пользователь может дополнительно форматировать вид созданной таблицы. Для этого достаточно выбрать в таблице свойство Columns. Открывается окно, представленное на рис. 9.2. Можно установить заголовки столбцов (Header text). При открытии других закладок данного окна можно менять шрифт, типы линий и цвета.

При создании формы на экране размещается элемент управления кнопка, на которой написано Load. При нажатии этой кнопки содержимое таблицы стирается, приложение связывается с источником данных, набор DataSet заполняется заново и таблица отображает новое состояние выбранной таблицы.

Существуют и другие типы элементов управления, которые можно связывать с данными.

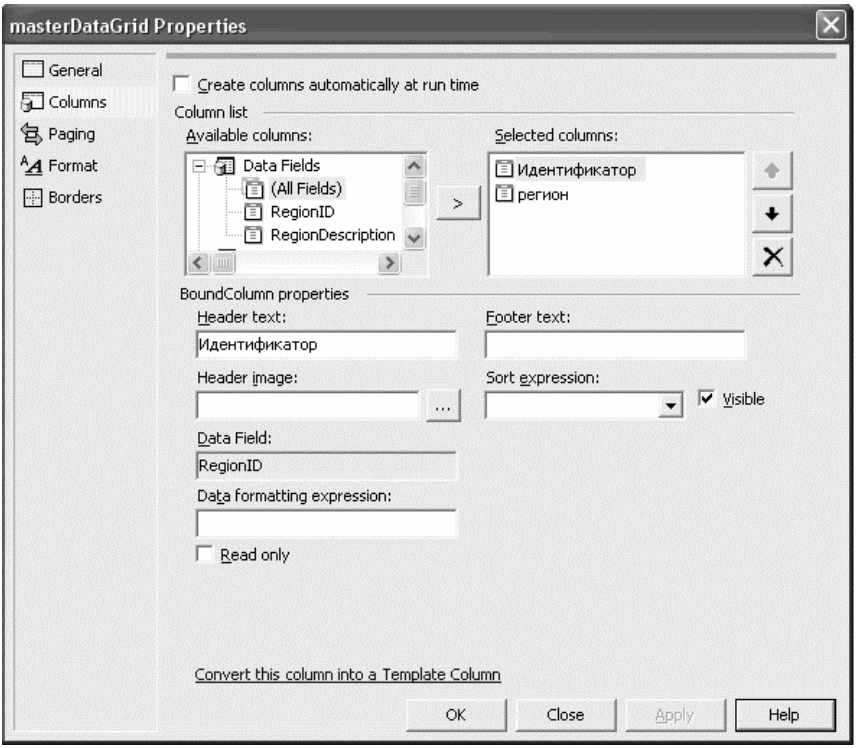

Рис. 9.2. Окно установки свойств элемента управления DataGrid

Для того чтобы связать элемент управления Web с объектом DataSet, необходимо при загрузке формы в обработчике события Load вызвать метод DataBind. Метод DataBind может относиться к конкретному элементу управления или к текущей странице. Когда метод DataBind применяется к текущей странице (Me.DataBind), все элементы управления, связанные с данными, заполняются из соответствующих объектов DataSet.

Внесение изменений в источник данных. Изменения, произведенные в отсоединенных от данных объектах, необходимо передать в базу данных. Это можно осуществить разными способами. В данной работе мы для этой цели используем объект InsertCommand, являющийся свойством объекта DataAdapter. Данный объект реализует функцию добавления записи в таблицу. InsertCommand, в свою очередь, имеет коллекцию свойств Parameters. Parameters представляет собой массив значений, содержащий поля добавляемой записи. Когда параметры установлены, команда посылается на выполнение путем старта метода InsertCommand.ExecuteNonQuery(). Поскольку мы работаем в отрыве от источника данных, перед стартом данного метода необходимо открыть соединение (SqlConnection.Open()), а после выполнения команды закрыть его (SqlConnection.Close()).

Выполнение работы

В процессе выполнения работы создаем Web-приложение базы данных, связанное с базой данных, созданной на предыдущих занятиях.

1. Включаем компьютер. Запускаем Visual Studio.Net. Открываем новый проект (File → New →Project), в качестве языка разработки выбираем Visual Basic и вид проекта – ASP.NET Web application. Задаем имя проекта – Web_trans.

2. Создаем первую страницу приложения. Web-форме задаем свойство **Title** – «Система регистрации транзакций». Затем открываем на палитре объектов (Toolbox) закладку Web forms и размещаем на экране 4 объекта Label.

Устанавливаем свойства этих объектов. Для объекта Label1 устанавливаем значения следующих свойств:

- **Text** – «Система информационного обеспечения проведения транзакций»;

• **Font** – Bold устанавливаем в положение «True», Size – X-Large.

Для объекта Label2:

• **Text** – «Система позволяет осуществить следующие действия:»;

• **Font** – Bold устанавливаем в положение «True», Size – Larger.

Для объекта Label3:

• **Text** – «- Просмотреть информацию о транзакциях»;

• **Font** – Bold устанавливаем в положение «True».

Для объекта Label4:

• **Text** – «- Записать информацию о новой транзакции»;

• **Font** – Bold устанавливаем в положение «True».

Открываем на палитре объектов закладку HTML и переносим на страницу три элемента управления типа «командная кнопка» (Button). Устанавливаем для всех элементов свойство **language** – vbscript. Свойство **value** у них устанавливаем соответственно: «ПРОСМОТР», «ТРАНЗАКЦИЯ», «ЗАКРЫТЬ». Имя кнопок (свойство **ID**) соответственно: cmdTrans, cmdView. cmdQuit.

Сохраняем проект (File → Save All).

3. Проверяем результат. Для этого открываем окно Solution Explorer, выбираем имя формы WebForm1.aspx. Щелкаем правой кнопкой мыши и в выпадающем меню запускаем опцию View in browser. На экране должна появиться следующая страница (рис. 9.3).

4. Программируем событие нажатия кнопки «ЗАКРЫТИЕ». Поскольку кнопки являются объектами HTML, для ее программирования открываем окно кода HTML (переключатель внизу окна). И вставляем в тело страницы после тега <body> следующий текст:

```
<script LANGUAGE="vbscript">
sub cmdQuit_onClick()
if(MsgBox("Вы хотите прекратить работу? ",_
    vbYesNo+vbQuestion, "Вопрос")=vbYes) then
    Window.Close
    end if
</script>
```

Сохраняем проект и проверяем выполнение команды.

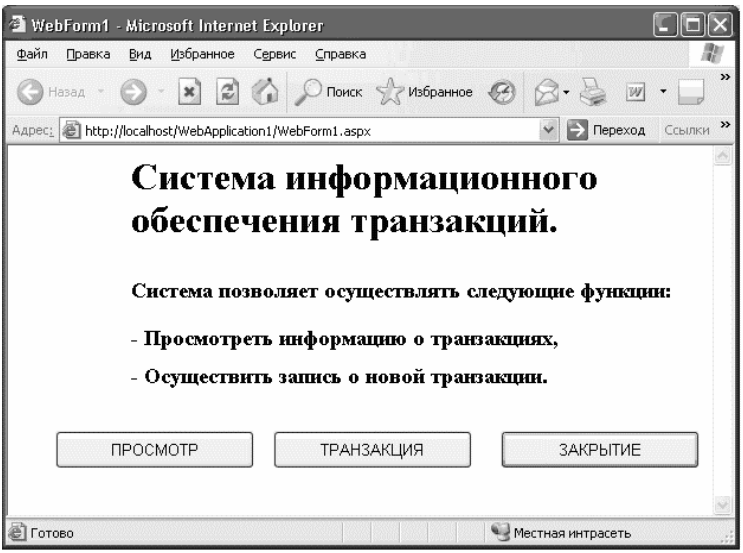

Рис. 9.3. Первая страница приложения

5. Создаем вторую страницу приложения, служащую для просмотра информации о транзакции. Стартуем опцию меню Project → Add Web Form, выбираем тип формы – Data Form Wizard. Начинает работать помощник формирования формы просмотра данных. Выполняя инструкции помощника, последовательно выполняем ряд действий:

- Дать имя новому набору Dataset – «Transact».
- Открыть соединение, созданное в процессе выполнения предыдущих работ (можно открыть новое).
- Для наполнения набора данных выбрать представление trans_info.
- Закончить работу помощника.

На экране появится сетка данных (DataGrid) и кнопка загрузки информации в связанный объект. Можно просмотреть программный код созданной страницы, для этого нужно щелкнуть два раза на кнопке загрузки. При этом открывается код обработки события на сервере. Можно также просмотреть код HTML.

Введем некоторые изменения и дополнения на созданной странице. Добавляем заголовок формы: **title** – «Просмотр транзакций».

Затем изменим названия столбцов таблицы. Для этого открываем окно установки свойств таблицы и выбираем свойство Columns. Последовательно устанавливаем новые заголовки: «Имя пользователя», «Дата проведения», «Тип транзакции».

Затем выделяем кнопку загрузки и заменяем надпись на кнопке на значение «Обновить». Размещаем рядом с кнопкой загрузки еще одну кнопку – элемент управления HTML. Задаем ее свойства: **Value** – «Возврат», **ID** – cmdBack. На странице HTML пишем скрипт обработки события нажатия кнопки. По нажатию этой кнопки мы будем возвращаться на первую страницу:

```
<script language="vbscript">
sub cmdBack_onClick
    Window.Location.Href="webform1.aspx"
end sub
</script>
```

Сохраняем проект. Вид созданной таблицы представлен на рис. 9.4.

Рис. 9.4. Форма просмотра информации о транзакции

Возвращаемся на первую страницу webform1.aspx и в текст HTML добавляем скрипт загрузки второй формы. Следующий текст добавляется рядом со скриптом закрытия приложения:

```
sub cmdView_onClick()
    Window.Location.Href="DataWebForm1.aspx"
end sub
```

Сохраняем проект, запускаем его на выполнение.

Если при запуске Web-приложения возникнут проблемы с доступом к данным, необходимо открыть SQL Enterprise Manager и выполнить следующие действия:

- открыть новый серверный логин – ASPNET (стандартный пользователь ASP приложения);
- добавить этот логин в качестве пользователя вашей базы данных;
- установить для нового пользователя разрешения на просмотр представления и на запуск хранимой процедуры.

6. Создаем третью страницу Web-приложения, на которой будем определять параметры транзакции и добавлять информациию в базу данных.

Создаем новую Web-форму (Project → Add Web-form). Устанавливаем заголовок формы – «Запись информации о транзакции». Размещаем на форме 4 объекта Label, 3 объекта DropDownList, 2 командные кнопки из палитры Web form и 1 кнопку из палитры HTML.

Для объектов Label установить следующие значения свойства **Text**: «Определите параметры транзакции и введите ее в базу данных», «ПОЛЬЗОВАТЕЛЬ», «КОНТЕЙНЕР» и «ТРАНЗАКЦИЯ». У главного заголовка также установите параметры шрифта: **Bold** – True, **Size** – larger.

Для кнопок установите следующие свойства. Двум кнопкам из объектов Web form присваиваем имена cmdLoad, cmdWrite и устанавливаем надписи «ЗАГРУЗИТЬ» и «ЗАПИСАТЬ» соответственно. Для кнопки из объектов HTML устанавливаем имя cmdLook и заголовок «ПРОСМОТРЕТЬ». В итоге получаем форму, представленную на рис. 9.5. Сохранить проект.

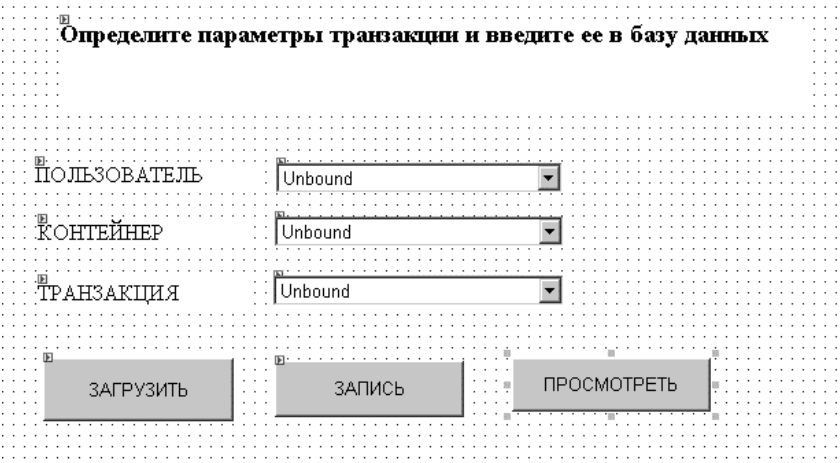

Рис. 9.5. Форма определения и записи транзакции

7. Программируем кнопку «ПРОСМОТР». Для этого открываем HTML текст и вставляем в тело скрипт, загружающий вторую форму. Он аналогичен скрипту в пункте 5. Сохранить проект, проверить работу приложения.

8. Связываем объекты DropDownList с данными. Для этого необходимо организовать соединение с базой данных (объект Connection), создать три объекта DataAdapter для связи с конкретной таблицей, создать набор DataSet и связать данные в этом наборе с объектами управления.

Открываем палитру объектов Data и размещаем на форме объект SqlDataAdapter. Запускается помощник для создания объекта. Выбираете уже существующее соединение с базой данных. Затем выбираете опцию – Use SQL statement. Далее открываете редактор запросов и формируете запрос для вывода всей информации из таблицы Users.

Аналогичные процедуры осуществляются с таблицами Containers и TransactionTypes.

Затем создаем объект DataSet. Запускаем опцию меню Data → Generate DataSet. Выбираем новый набор данных, указываем его имя Dat_trans, сохраняем отметку всех трех таблиц, чтобы они вошли в этот набор, и подтверждаем его создание.

Затем связываем объекты DropDownList с данными. Для этого устанавливаем параметры **DataSource** – Dat_trans1. **DataMember** – имя таблицы, соответственно users, containers и transaction_types. **DataTextField** – название выводимого поля, соответственно user_name, con_name и trans_type. Кроме того, свяжем еще одно свойство объекта с полем таблицы. Это делается для того, чтобы легко получить доступ к первичному ключу, соответствующему имени. Устанавливаем свойство **DataValueField** – соответственно userid, conteinerid и transactiontypeid.

Для заполнения объекта DataSet данными необходимо добавить в обработку события загрузки формы команды заполнения объекта данными.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e_
    As System.EventArgs) Handles MyBase.Load
      SqlDataAdapter1.Fill(Dat_trans1.users)
      SqlDataAdapter2.Fill(Dat_trans1.containers)
      SqlDataAdapter3.Fill(Dat_trans1.transactiontypes)
End Sub
```

Программируем кнопку «ЗАГРУЗИТЬ». Щелкаем на ней два раза и в открытой подпрограмме пишем простую команду:

```
Private Sub cmdLoad_Click(ByVal sender As System.Object, ByVal_
    e As System.EventArgs) Handles cmdLoad.Click
        Me.DataBind()
End Sub
```

Сохраняем проект и проверяем его работоспособность.

9. Программируем событие нажатие кнопки записи информации о транзакции. По нажатию этой кнопки необходимо записать информацию в таблицу Transactions. Для этого создадим новый объект SQLDataAdapter.

Затем добавляем в набор данных Dat_trans новую таблицу. Для этого запускаем команду Data → Generate DataSet, выбираем существующий набор, добавляя в него таблицу Transactions.

Затем записываем подпрограмму обработки события нажатия кнопки «Записать».

```
Private Sub cmdWrite_Click(ByVal sender As System.Object, ByVal_
    e As System.EventArgs) Handles cmdWrite.Click
      SqlDataAdapter4.InsertCommand.Parameters(0).Value = Now()
      SqlDataAdapter4.InsertCommand.Parameters(1).Value =_
    DropDownList1.SelectedItem.Value.ToString
      SqlDataAdapter4.InsertCommand.Parameters(2).Value =_
    DropDownList3.SelectedItem.Value.ToString
      SqlConnection1.Open()
      SqlDataAdapter4.InsertCommand.ExecuteNonQuery()
      SqlConnection1.Close()
      SqlDataAdapter4.Fill(Dat_trans1.transactions)
End Sub
```

Сохраняем проект. Демонстрируем работу приложения преподавателю. Получаем подпись. Завершаем работу.

Дата _____ Подпись преподавателя _____

Лабораторная работа 10
ИСПОЛЬЗОВАНИЕ WEB-СЕРВИСОВ В ПРИЛОЖЕНИЯХ БАЗ ДАННЫХ

Цель работы: ознакомить студентов с процессом создания и использования Web-сервисов с применением технологии ASP.Net. Продемонстрировать использование ADO.Net для доступа к данным SQL Server 2000 из Web-сервиса и последующую передачу данных приложению с использованием XML формата.

Теоретические основы

Web-службы (сервисы) появились как решение, позволяющее стандартным способом получать необходимые данные, без какого-либо специально для этого созданного программного или аппаратного обеспечения. Краеугольным камнем технологии Web-служб является их способность передавать данные от поставщика к потребителю, используя всего лишь повсеместно распространенный HTTP-протокол; при этом в качестве формата данных используется XML, что существенно облегчает преобразование первичных данных в формат, пригодный для просмотра пользователем. Такое простое преобразование не требует сложных программ для разбора данных. В документе «Defining the Basic Elements of .Net» Microsoft определяет Web-службу так:

«Web-службы, основанные на XML, служат для обмена данными между приложениями и, что более важно, позволяют вызывать другие приложения независимо от того, как эти приложения устроены, на какой платформе они работают и какие устройства используются для доступа к ним».

Появление Web-служб влечет серьезные изменения в парадигме разработки программного обеспечения. Web-службы подталкивают нас к разбиению больших приложений на небольшие независимые части, которые могли бы существовать в качестве Web-служб. Такая модель, возможно, заменит существующую парадигму, в соответствии с которой разбиение происходит на динамические библиотеки (DLL, Dynamic Link Library) и COM (Component Object Model) объекты.

На самом деле, Web-службы и библиотеки DLL весьма похожи. И те и другие аккумулируют некий набор взаимосвязанных функций; например, бизнес-логику или логику доступа к базе данных. Тем не менее, между ними есть и существенная разница. Во-первых, Web-службы доступны через протокол HTTP, что позволяет любому Web-клиенту вызвать их. В случае DLL все обычно происходит по-другому, и клиент находится в том же интранете, что и DLL. Таким образом, Web-службы открывают новую эру распределенных вычислений. Во-вторых, Web-службы возвращают данные клиенту в формате XML. DLL обычно возвращают типы данных, специфические для используемого языка программирования.

Эти отличия между Web-службами и их предшественниками (DLL) определяются следующими тенденциями в программной индустрии, проявившимися до появления Web-служб:

- принятие HTTP как стандартного протокола, с помощью которого осуществляется доступ в Интернет;
- принятие XML де-факто как стандарта для передачи данных.

Эти две тенденции обеспечили базис, на котором были построены Web-службы.

Web-служба – это класс, расположенный на Web-сервере и обслуживающий подобно приложению ASP запросы клиентов. Отличается он от последнего тем, что не генерирует Web-страниц для отправки клиенту. Он ведет себя аналогично функции.

Возникает вопрос: смогут ли другие операционные системы и Web-серверы взаимодействовать друг с другом посредством Web-служб? Таинственными ингредиентами, благодаря которым все это должно заработать, являются XML и SOAP (Simple Object Access Protocol). Это открытые стандарты, которые очень легко реализовать. Всегда можно создать ASP страницу, возвращающую одно или более значений в формате XML или SOAP, и отправить ее клиенту вместо HTML страницы. Оба формата позволяют с помощью протокола HTTP отправлять информацию любого типа, так что если вторая сторона может обрабатывать теги HTML, она сможет использовать Web-службы. В настоящее время все ведущие производители программного обеспечения обещают включить в свои продукты поддержку XML, что делает Web-службы исключительно многообещающей технологией, поскольку она основана на стандартах.

Создание Web-служб. При создании Web-службы определяются методы класса. Перед определением каждого метода размещается следующий тег: <WebMethod()>. Этот тег сообщает компилятору, что данный метод должен быть доступен для запросов HTTP. В составе класса могут быть и локальные члены. Подобно членам элементов управления, тег Web Method может содержать атрибуты.

Кроме того, этот класс выводит информацию о себе на странице, автоматически генерируемой при попытке обращения к членам Web-службы. Такого рода информация включает имя службы и имена ее членов. Полученная страница даже позволяет протестировать Web-службу из Internet Explorer. Конечно, формат выходных данных довольно необычен, но это единственный способ получить информацию посредством протокола HTTP. Кроме того, через брандмауэры проходит только текст. Для форматирования результирующих значений и передачи их клиенту Web-службы используют XML. В результате Web-служба и ее пользователь (вызывающее ее приложение) делают вид, что обмениваются запросами и HTML страницами в текстовом формате. Как видите, для использования Web-служб ни на клиенте, ни на сервере не понадобится устанавливать какие-либо дополнительные компоненты.

Выполнение работы

1. На первом этапе выполнения работы создадим простую Web-службу, позволяющую рассчитать количество радиоактивного материала с учетом распада. Затем создадим Web-приложение, которое вызывает эту службу.

Для создания Web-службы запускаем Visual Studio.Net и открываем новый проект с именем http://localhost/DecayService с типом проекта ASP.NET Web service.

Открываем окно кода и добавляем в существующий шаблон службы следующий код.

```
<WebMethod()> Public Function Decay(ByVal m0 As Double, _
   ByVal lyambda As Double, ByVal t As Double) As String
     Const exp = 2.71828
     Decay = m0 * exp ^ (-lyambda * t)
End Function
```

Функция Decay возвращает массу изотопа после определенного времени распада.

Сохранить проект. В Solution Explorer выделить файл Service1.asmx, щелкнуть правой кнопкой и запустить опцию выпадающего меню View in Browser. Открывается окно проверки работы службы. Щелкаем на гипертекстовой ссылке Decay – открывается интерфейс ввода формальных параметров (рис.10.1).

Вводим некоторые данные, нажимаем клавишу Invoke и получаем результат в виде XML записи

<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">2,62558771933794E-07</string>

Web-служба создана.

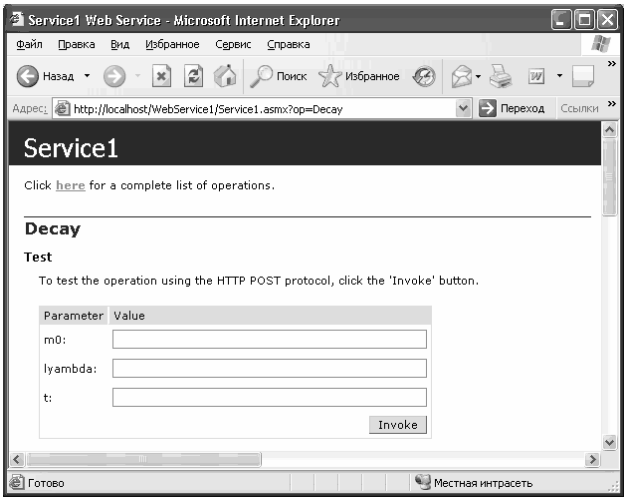

Рис. 10.1. Окно тестирования созданного сервиса

2. Создаем Web-интерфейс, который будет вызывать Web-службу. Открываем новый проект – ASP.NET Web application с именем DecayApplication.

На форме размещаем ряд Web-объектов управления таким образом, чтобы получилась форма, представленная на рис. 10.2. Размещаем на форме пять сетевых элементов управления типа Метка. Заполняем значения их свойства Text содержимым, указанным на рисунке. Затем размещаем четыре сетевых элемента управления типа TextBox. Наконец, размещаем командную кнопку и вводим на ней надпись «Рассчитать».

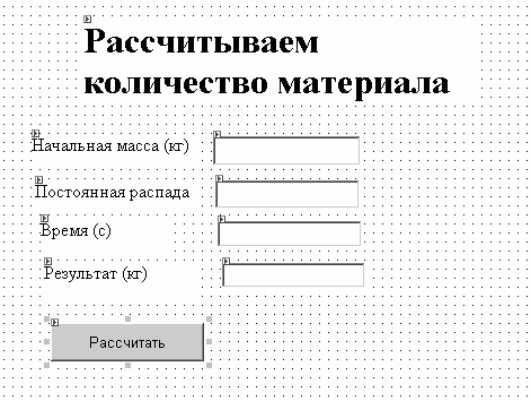

Рис. 10.2. Web-интерфейс для вызова сервиса

3. Создаем ссылку на Web-службу. Для этого вызываем опцию главного меню Project → Add Web reference. Открывается форма подключения службы к приложению. Необходимо указать адрес ресурса, содержащего требуемую службу. Для этого выбираем локальный сервер. В окне высвечивается список Web-служб, зарегистрированных на сервере. Выбрав нужную, получаем окно, представленное на рис. 10.3. Нажимаем клавишу Add Reference.

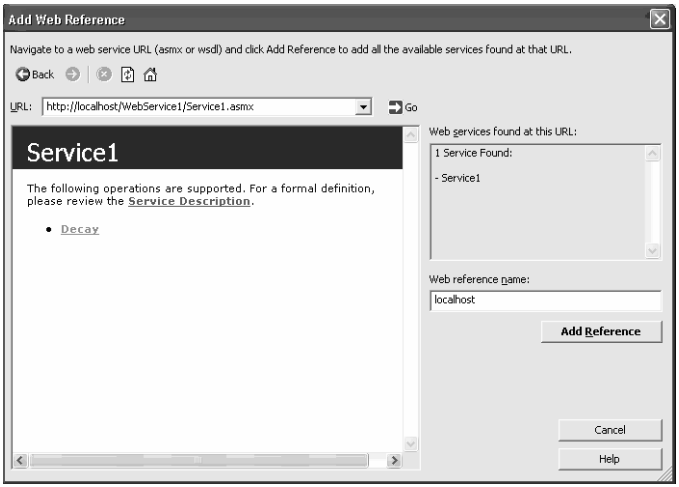

Рис. 10.3. Подключение сервиса к приложению

4. Далее пишем код обращения к функции Web-службы. Щелкаем два раза на кнопке и в открывшемся окне пишем следующий код.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
    e As System.EventArgs) Handles Button1.Click
      Dim ws As New DecayApplication.localhost.Service1()
      Dim a, b, c As Double
      a = Convert.ToDouble(TextBox1.Text)
      b = Convert.ToDouble(TextBox2.Text)
      c = Convert.ToDouble(TextBox3.Text)
      TextBox4.Text = ws.Decay(a, b, c)
   End Sub
```

В окне Solution Explorer отмечаем Service1.aspx, щелкаем правой кнопкой и нажимаем кнопку View in Browser. Проверяем работу приложения, демонстрируем результат преподавателю.

5. На втором этапе работы создаем Web-службу для работы с данными. Она будет выполнять работу связи с базой данных и передачи этих данных в формате XML приложению.

Открываем новый проект http://localhost/LookDBService типа ASP.NET Web Service. Создаем SQLDataAdapter. Для этого запускаем помощника, выбираем созданный ранее объект связи с данными и формируем выходной набор данных на основе представлении trans_info.

Создаем объект DataSet. Для этого запускаем функцию генерации набора данных Data → Generate DataSet. Отмечаем окно «Добавить набор данных на форму» и создаем новый набор данных DSLook.

Теперь создаем функцию, которая получает в качестве формального параметра строковую переменную, содержащую имя пользователя. Затем динамически изменяем содержимое SQLDataAdapter, изменяя строку запроса, и возвращаем набор данных DataSet. Для этого в теле текста сервиса пишем следующий код:

```
<WebMethod>
Public Function GetData(ByVal strName As String) As DSLook
      Dim tr As New DSLook
      Dim str As String
      str = “Select * from trans_info where username=”
      str = str & strName & “”
```

```
    SqlDataAdapter1.SelectCommand.CommandText = str
    SqlDataAdapter1.Fill(tr)
    Return tr
End Function
```

Сохраняем приложение и тестируем работу сервиса. На рис.10.4 приведен фрагмент возвращаемого текста в формате XML.

Рис. 10.4. Фрагмент XML текста переданного сервисом

6. Создаем Windows приложение, которое использует данную службу, как источник данных. Открываем новый проект с именем WinDBService. Добавляем ссылку на web service с именем http://localhost/LookDBService/service1.asmx.

Формируем интерфейс представления данных. Для этого размещаем на форме объект TextBox, DBGrid и командную кнопку. Устанавливаем надпись на кнопке – «ЗАГРУЗИТЬ». Форма представлена на рис. 10.5.

Создаем набор данных, который получает информацию от данных, передаваемых сервисом. Для этого вытаскиваем из вкладки Data окна ToolBox на форму объект DataSet. На рис. 10.6 представлено окно формирования набора DataSet.

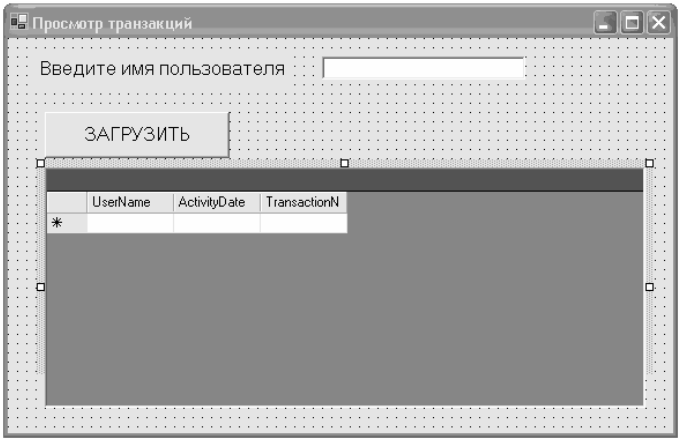

Рис. 10.5. Форма вызова Web сервиса из Windows приложения

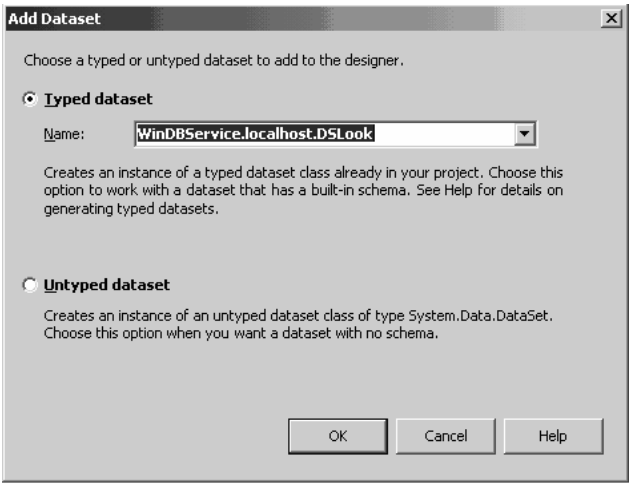

Рис. 10.6. Окно формирования набора данных

Затем связываем сетку данных с данными, для этого устанавливаем свойство DataSource – DsLook1.trans_inf.

На заключительном этапе программируем событие нажатия кнопки «ЗАГРУЗКА». Щелкаем два раза на кнопке и вставляем следующий отрывок текста:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
    e As System.EventArgs) Handles Button1.Click
      Dim ws As New WinDBService.localhost.Service1()
  Dim str as string
      Str=TextBox1.Text
      DsLook1.Merge(ws.GetData(str))
End Sub
```

Сохранить все, проверить работу приложения, получить подпись преподавателя.

Дата _____ Подпись преподавателя _____

Лабораторная работа 11
E/Z MAS – КОМПЬЮТЕРИЗИРОВАННАЯ СУиК НА ОСНОВЕ WEB-ТЕХНОЛОГИИ

Цель работы: познакомить студентов с дизайном базы данных, интерфейсом и функциональностью компьютеризированной СУиК E/Z MAS, созданной на основе Web-технологии. Дать понятие о конфигурировании системы, ознакомить с моделью предприятия, заложенной в системе. Продемонстрировать легкость расширения системы за счет включения новых функциональных возможностей.

Теоретические основы

Точное описание процессов производства, транспортировки и хранения ядерных материалов является важным фактором в деле защиты и предотвращения хищений ядерных материалов. Предприятия по хранению ядерных материалов, перерабатывающие заводы и производители ядерного топлива должны собирать и хранить данные о транзакциях, осуществляемых с ядерными материалами.

Подобная информация должна включать в себя следующие данные:

- тип материала;
- изотопный состав;
- количество;
- место расположения;
- паспортные данные (набор характеристик, уникальным образом описывающих материал);
- паспортная история.

Компьютеризированная СУиК материалов позволяет ядерному предприятию автоматизировать слежение за ядерными материалами и регистрировать несанкционированное переключение и хищение материалов. Программа УиКМ следит за перемещениями материалов по всему ядерному предприятию и генерирует итоговые отчеты.

Общее описание E/Z MAS. СУиК E/Z MAS – это система, основанная на технологии World Wide Web. Она позволяет следить за наличием ядерных материалов и вести учетные записи для управления и контроля материала.

Основополагающими критериями при разработке E/Z MAS были простота, легкость в обращении и экономичность. Они отразились в каждом аспекте программы E/Z MAS от первоначальных установок системы до функционирования программы и манипуляции базой данных.

E/Z MAS разработана в Лос-Аламосской национальной лаборатории с использованием только коммерческих программных продуктов. Система работает в режиме клиент/сервер. Данные хранятся в СУБД, находящейся на сервере с операционной системой Microsoft Windows NT. Контроль доступа и другие функции защиты, необходимые для системы учета материалов, обеспечиваются сервером. Интерфейс пользователя, обеспечивающий функции ввода, извлечения и модификации информации, работает на рабочей станции и получает доступ к серверу через броузер. Программное обеспечение E/Z MAS создано с использованием VBScript и HTML, что обеспечивает среду для быстрой разработки экрана интерфейса пользователя. Рабочие станции соединяются с сервером посредством Ethernet или телефонной линии.

Система содержит универсальную функциональность, необходимую всем потенциальным предприятиям-пользователям, которая может быть легко расширена добавлением специфических функций.

Дизайн базы данных. В таблицах представлены основные сущности базы данных. Материал представлен «партией» материала. Он должен быть заключен в контейнер. Без контейнера материал существовать не может. Пустой контейнер может быть в наличии. Данная версия СУиК построена на основании СУБД Access. В связи с этим необходимо при установке системы поставить в файле header.asp установки глобальных переменных соответствующих базам данных Access.

Функциональность E/Z MAS. Перечислим основные функции, которые реализует система.

1. *Внешние получения.* Позволяет пользователю регистрировать получение контейнеров с материалами. Необходимо выбрать ЗБМ назначения (куда контейнеры будут направлены) и внешнее предприятие (откуда контейнеры поступили), а также определить состав материала и число контейнеров.

2. *Внешние отправления.* Позволяет пользователю регистрировать отправление контейнеров с материалами. Необходимо выбрать исходную ЗБМ (откуда отправляются контейнеры) и предприятие назначения (куда направляются контейнеры). Для отправления можно выбрать один или несколько контейнеров.

3. *Перемещения контейнеров внутри ЗБМ.* Позволяет пользователю регистрировать перемещения контейнеров из исходного физического места в физическое место назначения внутри ЗБМ. Необходимо выбрать ЗБМ. Можно перемещать один контейнер или одновременно несколько контейнеров.

4. *Перемещения контейнеров между ЗБМ.* Позволяет пользователю зарегистрировать в одной операции перемещение контейнеров из исходной ЗБМ в ЗБМ назначения. Необходимо выбрать исходную ЗБМ (где контейнеры ранее хранились) и ЗБМ назначения (куда контейнеры направляются). Можно выбрать и направить в ЗБМ назначения один контейнер или одновременно несколько контейнеров.

5. *Перевод в Транзит.* Позволяет пользователю переводить контейнеры в состояние транзита. Это первый шаг в регистрации перемещения контейнера между ЗБМ. Необходимо выбрать исходную ЗБМ и ЗБМ назначения. В состояние транзита можно перевести один контейнер или одновременно несколько контейнеров.

6. *Получение из Транзита.* Позволяет пользователю получить ранее отправленный с помощью опции Перевод в Транзит материал. Это второй шаг в регистрации перемещения контейнера между ЗБМ. Необходимо выбрать ЗБМ назначения (где контейнер будет получен). Получить можно один или несколько контейнеров.

7. *Инвентарные изменения → Объединение.* Позволяет пользователю объединять материал из контейнеров в новый материал в одном или нескольких контейнерах. Необходимо выбрать исходную ЗБМ (где контейнеры хранятся). Для объединения можно выбрать один или несколько контейнеров. Для размещения нового материала также можно выбрать один или несколько контейнеров.

8. *Инвентарные изменения→Разделение.* Позволяет пользователю регистрировать разделение одной партии материала на две (в контейнерах). Необходимо выбрать исходную ЗБМ (где хранится контейнер) и сам контейнер, из которого берется партия для разде-

ления, а также ввести вес каждого материала в каждом из контейнеров.

9. *Инвентарные изменения → Балк-разделение.* Позволяет пользователю регистрировать операции изотопного разделения материала в контейнере. Необходимо выбрать исходную ЗБМ (где хранится контейнер) и контейнер, содержащий существующую партию. Следует ввести вес всех элементов и изотопов для каждого материала в каждом из контейнеров.

10. *Контейнеризация → Создать контейнер.* Позволяет пользователю создавать новый контейнер (пустой контейнер без привязанного к нему материала). Необходимо выбрать ЗБМ и физическое место (в которых был создан новый контейнер).

11. *Контейнеризация → Вложить в контейнер.* Позволяет пользователю упаковывать (вкладывать) контейнеры в другой контейнер (внешний контейнер). Необходимо выбрать ЗБМ (где хранятся контейнеры) и внешний контейнер. Для вложения во внешний контейнер можно выбрать один или несколько контейнеров.

12. *Контейнеризация → Извлечь из контейнера.* Позволяет пользователю распаковывать (извлекать) контейнеры, находящиеся в других контейнерах. Необходимо выбрать ЗБМ, где хранится внешний контейнер. Можно выбрать один или несколько контейнеров, чтобы извлечь их из внешнего контейнера.

13. *Отчеты.* Позволяют пользователю видеть текущие и исторические данные. Пользователи могут видеть содержимое контейнера, содержимое ЗБМ или физического места, а также транзакции, генерируемые функциями E/Z MAS.

14. *Физическая инвентаризация.* Позволяет пользователю создавать, подтверждать правильность и составлять отчеты о текущей или прошлой физической инвентаризации. Необходимо выбрать ЗБМ или физическое место, для которых генерируется произвольный список контейнеров, и ввести процент инвентаризации (от 1 до 100% единиц в ЗБМ или физическом месте). Дата инвентаризации конкретного контейнера в конкретной ЗБМ или физическом месте может быть подтверждена. Можно увидеть конкретные отчеты по инвентаризации.

15. *Устройства индикации целостности.* Позволяет пользователю следить за созданием, установкой, проверкой и удалением

УИЦ (пломб). Можно увидеть отчет о действиях, связанных с УИЦ (статус контейнеров и УИЦ).

16. *Конфигурация данных.* Позволяет администратору СУиК конфигурировать (создавать, переименовывать и удалять) и контролировать процесс использования E/Z MAS. Можно конфигурировать Предприятия, ЗБМ и Физические места. С помощью функции конфигурации данных также создаются, модифицируются и удаляются Права входа в E/Z MAS и Права пользователей.

Интерфейсы E/Z MAS. Вся вышеописанная функциональность реализована с помощью ASP технологии. Каждый файл системы включает два включаемых файла – header.asp и footer.asp. Первый содержит необходимые установки параметров, устанавливает фон и базовые цвета системы и выводит титул и заголовки файла. Последний выводит пользовательское меню, которое позволяет с любой страницы переходить на любую группу функций системы.

Кроме того, практически каждая страница включает файл connect.asp, содержащий процедуру открытия соединения с базой данных и открытия набора записей. Если соединение было установлено ранее, оно просто восстанавливается на данной странице.

Вся функциональность работы базы данных реализуется через sql запросы. Для открытия набора используется метод набора типа Recordset – Open. Запросы выполняются с помощью метода объекта Connection – Execute.

Выполнение работы

1. Изучения дизайна базы данных E/Z MAS. Для этого запускается приложение MS Access. В нем открывается копия базы данных EzmasDB. Нужно выполнить следующее.

2. Просмотреть схему объектных отношений базы данных. Для этого открыть меню Сервис → Схема Отношений.

3. Выписать в отчет основные параметры модели предприятия, заведенную в базу данных. Для этого последовательно открывать и просматривать таблицы:

- MBA – список зон баланса материалов;
- PhisLoc – физические места расположения материалов;
- Batch – названия партий материалов;
- MatType – тип материала;
- ConType – типы контейнеров;
- TransType – типы транзакций;
- Users – пользователи.

4. Составить запрос, который выводит таблицу, состоящую из двух полей, одно из которых содержит названия типов транзакций, а второе – количество транзакций этого типа. Для этого используйте запрос sql, содержащий функцию Count() по всем полям таблицы Trans, сгруппированный по типам транзакций и связанную с таблицей Trans таблицу TransType. При составлении запроса используйте утилиту «создать запрос». Выберите нужные таблицы и поля. На рис. 11.1 приведено окно формирования запроса.

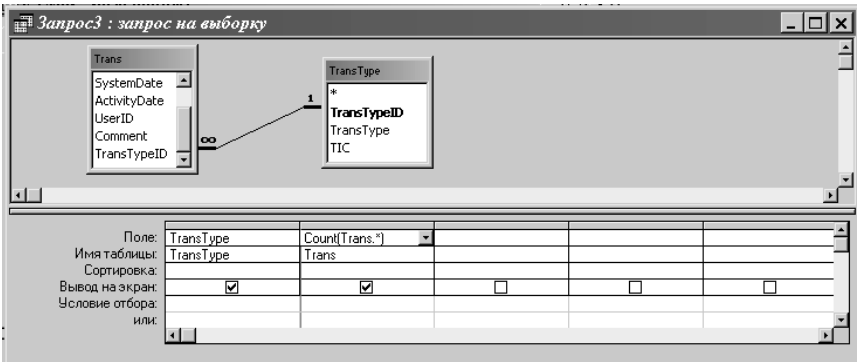

Рис. 11.1. Составление запроса по типам транзакций

Переключите окно запроса в вид SQL (рис. 11.2) и добавьте группировку. Сохраните запрос. Отладьте его, просмотрите результат.

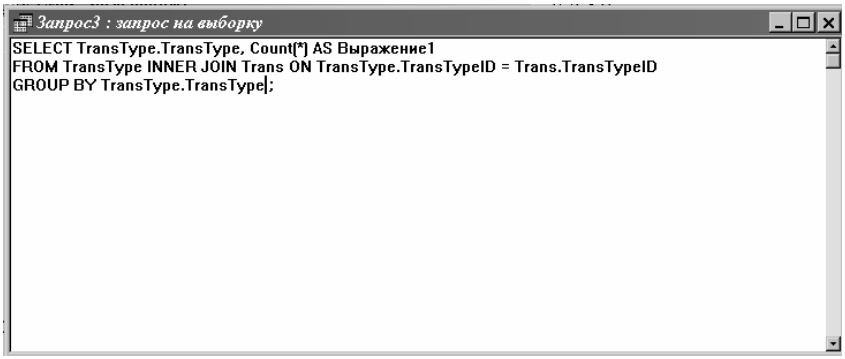

Рис. 11.2. SQL запрос для получения требуемой таблицы

5. Стартуйте Internet Explorer. Подключитесь к СУиК E/Z MAS, той версии, на которую вам укажет преподаватель. Войдите в систему с именем и паролем, полученным у преподавателя. Просмотрите отчеты по транзакциям за последние дни.

Проведите несколько транзакций по перемещению выбранного контейнера между различными ЗБМ, в одной ЗБМ.

Просмотрите отчет и определите свои транзакции и транзакции других пользователей.

6. Создайте новую функцию отчета и включите ее в E/Z MAS. Для этого:

• В странице ReportMenu.asp включите в таблицу новую «кнопку». Организуйте ее ссылку на файл SumOfTrans.asp. Сохраните страницу и просмотрите результат.

• Откройте файл UserTrans3.asp и сохраните его как SumOfTrans.asp. Преобразуйте файл следующим образом.

```
<HTML>
<!--#include FILE="header.asp"-->
<!--#include FILE="connect.asp"-->
<!--#include FILE="CheckDate.asp"-->
<!--#include FILE="print_errors.asp"-->
<H2 ALIGN=CENTER TEXT="0000FF"> Количество транзакций разных типов </H2>
<% sql="SELECT Count(*) AS Expr1, TransType.TransType"
    sql = sql & " FROM TransType INNER JOIN Trans ON TransType.TransTypeID = Trans.TransTypeID"
    sql = sql & " GROUP BY TransType.TransType"
      connect conn, rs
      rs.Open sql,conn,1,3 %>
    <TABLE BORDER BGCOLOR="#CCFFFF">
      <TR>
      <TH> Тип транзакции</TH>
      <TH> Количество транзакций</TH>
      </TR>
      <%do while Not rs.eof%>
      <TR>
      <TD><%=rs("TransType")%></TD>
```

```
<TD><%=rs("Expr1")%></TD>
</TR>
<%
rs.MoveNext
loop
rs.Close%>
</TABLE><BR>
<INPUT TYPE="button" NAME="cmdBut"
VALUE="Возврат назад">
<SCRIPT language=vbscript>
Sub cmdBut_onClick()
  Window.Location.Href="ReportMenu.asp"
end sub </Script>
<!--#include FILE="footer.asp"-->
```

Обратите внимание, что использован запрос, созданный на этапе работы с базой данных в MS Access. Его можно оттуда скопировать.

7. Продемонстрировать преподавателю работающее приложение, получить подпись преподавателя.

Дата _____ Подпись преподавателя _____

Владимир Вениаминович Кондаков

БАЗОВОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРИЗИРОВАННЫХ СИСТЕМ УЧЕТА И КОНТРОЛЯ ЯДЕРНЫХ МАТЕРИАЛОВ

Лабораторный практикум

Редактор Н.В. Егорова
Компьютерная верстка Г.А. Бобровой

Подписано в печать 6.11.2007 Формат 60x84 1/16
Печ.л. 8,75 Уч.-изд.л. 8,75 Тираж 150 экз. Изд. № 3/28
Заказ № 0-608

Московский инженерно-физический институт (государственный университет).
115409 Москва, Каширское ш., 31

Типография издательства «ТРОВАНТ»
г. Троицк Московской области