

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

Развертывание и интеграция программных систем

Методические указания по выполнению лабораторных работ для студентов
всех форм обучения направления 09.03.04 Программная инженерия

Краснодар
2023

Составители: проф. Зайков В.П., ст. преп. Ковтун А.А., доц. Шумков Е.А.

УДК 004.41

Развертывание и интеграция программных систем: методические указания по выполнению лабораторных работ для студентов всех форм обучения направления 09.03.04 Программная инженерия. / Сост.: В.П. Зайков, А.А. Ковтун, Е.А. Шумков; Кубан. гос. технол. ун-т. Каф. Информационных систем и программирования, 2023 – 95 с.

Составлены в соответствии с рабочей программой курса «Развертывание и интеграция программных систем» для студентов всех форм обучения направления профессиональной подготовки 09.03.04 Программная инженерия.

Изложена краткая теория, общие методические указания и последовательность выполнения заданий по лабораторным работам.

Содержание

Часть 1. Практические приемы, инструментарий в развертывании программных систем	4
Введение.....	4
Лабораторная работа №1. Основы работы с VM Virtualbox. Установка серверной ОС на виртуальную машину.	6
Лабораторная работа №2. Управление загрузкой серверной ОС. Добавление ролей. Установка первого контроллера домена.	12
Лабораторная работа №3. Основы администрирования домена: добавление компьютера в домен, работа с учетными записями и группами.	20
Лабораторная работа №4. Администрирование файлового сервера.....	24
Лабораторная работа №5. Администрирование баз данных. Управление системными и пользовательскими БД.	33
Лабораторная работа №6. Настройка служб DNS и DHCP.	45
Часть 2. Технологии интеграции программных (информационных) систем .	55
Введение.....	55
Лабораторная работа №7. Определение потоков данных. Бизнес-модель предприятия, программное и информационное обеспечение	57
Лабораторная работа №8. Классические методы интеграции. Вариант интеграции на уровне платформ (платформы интеграции)	60
Лабораторная работа №9. SOA-интеграция. Проектирование корпоративной ИС с применением SOA	66
Лабораторная работа №10. Интеграция на базе архитектуры с общей шиной данных (ESB)	72
Лабораторная работа №11. Интеграция на базе микросервисов.....	78
Лабораторная работа №12. Практическое освоение понятия ETL. Системы и инструменты интеграции корпоративных данных.	88
Список рекомендуемой литературы	94

Часть 1. Практические приемы, инструментарий в развертывании программных систем

Введение

Развертывание программного обеспечения (англ. Software deployment), в более широком смысле, «развертывание программных систем» – это все те действия, которые делают программную систему (программное обеспечение) готовой к ее использованию. Данный процесс является частью жизненного цикла программных систем (см. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств).

В целом процесс развертывания состоит из нескольких взаимосвязанных действий с возможными переходами между ними. Эта активность может происходить как со стороны производителя, так и со стороны потребителя. Поскольку каждая программная система является уникальной, трудно предсказать все процессы и процедуры во время развертывания. Поэтому «развертывание» можно трактовать как общий процесс, соответствующий определенным требованиям и характеристикам. Развертывание может осуществляться программистом и непосредственно, в процессе разработки программного обеспечения.

Перечислим основные, связанные с этим термином, понятия.

Подсистема хранения (англ. database engine, storage engine).

Конфигурационное управление (англ. software configuration management, SCM).

Сборка (англ. build).

Модуль ядра, загружаемый модуль ядра (англ. loadable kernel module, LKM).

Эталонная реализация (от англ. reference implementation).

Установка программного обеспечения, инсталляция

Сервисная шина предприятия (англ. enterprise service bus, ESB)

Библиотека среды выполнения, библиотека времени исполнения (англ. runtime library, RTL)

Сервер баз данных (БД).

Сервис-ориентированная архитектура (SOA, англ. service-oriented architecture).

Непрерывная интеграция (CI, англ. Continuous Integration).

Инструментальное программное обеспечение.

Сервер приложений (англ. application server).

Двоичная совместимость, бинарная совместимость (англ. binary compatibility).

Насыщенное интернет-приложение (англ. rich internet application, RIA).

Динамический сайт.

Пользовательское пространство.

Загрузчик (англ. loader).

Слой аппаратных абстракций (Hardware Abstraction Layer, (HAL)

Трехуровневая архитектура (трехзвенная архитектура, англ. three-tier).

Снимок файловой системы, или снапшот, или снepsшот (англ. Snapshot).

Механизм копирования при записи (англ. Copy-On-Write, COW).

Открытая система.

Управляемый код (англ. managed code).

Распределительная вычислительная среда (англ. Distributed Computing Environment, сокр. DCE).

Кросс-компилятор (англ. cross compiler).

Уровень абстракции.

Планировщик задач.

Отказоустойчивый кластер (англ. High-Availability cluster, HA cluster).

Система управления веб-содержимым (Web Content Management System или WCMS).

Журналирование (англ. logging)

Связующее программное обеспечение, промежуточное программное обеспечение, программное обеспечение среднего слоя, межплатформенное программное обеспечение (англ. Middleware).

Высокая доступность (англ. high availability).

Репликация (англ. replication).

Просмотр кода (англ. code review), инспекция кода (англ. code inspection).

Пакетное задание (англ. batch job).

Менеджер памяти.

Мультизагрузка (англ. Multi-boot).

Общий ресурс (общий сетевой ресурс).

Редактор исходного кода.

По умолчанию.

Обертка библиотеки (англ. wrapper).

Развертывание и сопровождение программного обеспечения стандартизовано. Для этого имеются национальные стандарты РФ, идентичные международным: ISO/IEC 12207:2008, ИСО/МЭК 12207-2010, ISO/IEC 14764:99, ИСО/МЭК 14764-2002 и др.

Лабораторная работа №1. Основы работы с VM Virtualbox. Установка серверной ОС на виртуальную машину.

1 Цель работы

Научиться использовать среду VM Virtualbox. Получить навыки установки серверной операционной системы.

2 Краткая теория и порядок выполнения работы

Используемое программное обеспечение.

Для выполнения данной лабораторной работы понадобится компьютер с операционной системой Microsoft Windows или Linux Mint и установленным на нем программным обеспечением Oracle VM VirtualBox, а также дистрибутив операционной системы Windows Server 2008 (или Linux Ubuntu) на flash- накопителе или в виде образа DVD-диска (файл с расширением .iso).

В ходе выполнения лабораторной работы установим на виртуальную машину операционную систему Windows Server 2008.

Организацию работы виртуальной машины упрощенно иллюстрирует рис.1.1.

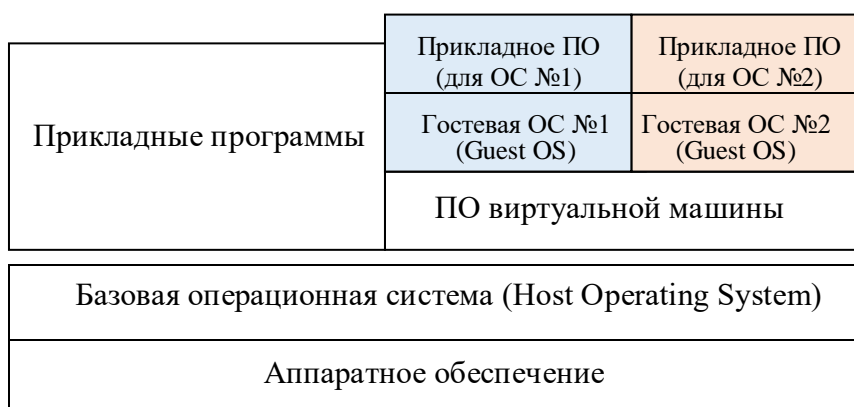


Рисунок 1.1 – Общая схема программно-аппаратной организации виртуальных машин

На компьютере первоначально устанавливается базовая операционная система, после чего устанавливается программное обеспечение виртуальной машины (VM). Оно позволяет установить одну или несколько гостевых операционных систем и запускать в них программы, разработанные для данных ОС. В качестве ПО виртуализации в лабораторных работах (Часть 1) будет использоваться Oracle VM VirtualBox. В ходе выполнения данной работы и последующих лабораторных работ, студентам предлагается проделать определенные «шаги» и продемонстрировать практические навыки преподавателю во время защиты работы. По результатам выполнения работ необходимо каждому студенту индивидуально, оформить отчет в виде краткого эссе на 3-5 страницы, включающего собственные, авторские выводы.

Шаг 1. Создание виртуальной машины

Из меню Пуск (Программы -> Oracle -> VM VirtualBox) запустим консоль управления виртуальными машинами (рис 1.2). Нажав кнопку **New...**, приступим к созданию новой машины.

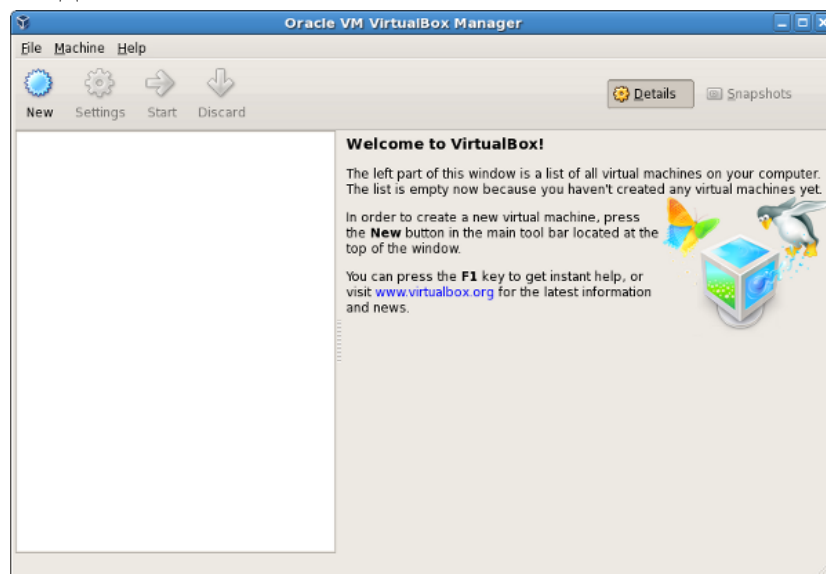


Рисунок 1.2 – Консоль управления виртуальной машиной

При создании новой машины необходимо прежде всего, описать расположение ее файлов (файла «машины» с расширением `.vbox` и файла виртуального жесткого диска с расширением `.vhd`). Поэтому в запущившемся после нажатия кнопки мастере указываем:

- создание новой машины (Create a virtual machine);
- имя и расположение файла (если специально не указывается, для выполнения этой лабораторной в классе используем имя S08, и путь D:\VirtPC_доп_лаб\S08; при самостоятельном выполнении работы можно использовать произвольные пути и имена);
- тип устанавливаемой операционной системы. Если это ОС разработки Microsoft, будут автоматически выставлены рекомендуемый объем оперативной памяти и виртуального жесткого диска. Поэтому в окне выбора ОС вместо значения по умолчанию —Other выберите из выпадающего списка Windows Server 2008;
- в следующих окнах мастера согласимся с предлагаемым объемом выделяемой машине оперативной памяти (512 MB); укажем, что надо создать новый виртуальный жесткий диск (A new virtual hard drive), проверим путь D:\VirtPC_доп_лаб\S08, поменяем предлагаемое название файла диска —S08 Hard Disk.vhd на более короткое —S08.vhd и уменьшим предлагаемый размер с 65 Гб до 40Гб. Размер и имя можно оставить и по умолчанию, изменение настроек предлагается «чтобы потренироваться». Тут надо отметить, что при использовании данного мастера файл виртуального диска создается

минимального размера и увеличивается по мере надобности, т.е. сразу 40 Гб не потребуется.

После сделанных настроек в окне консоли (рис.1.2) появится новая виртуальная машина, которую можно запустить, выделив ее и нажав кнопку Start. При этом может появиться сообщение об ошибке «The virtual machine could not be started because there was not enough memory available on the host» - виртуальная машина не может быть запущена, потому что недостаточно памяти. Это произойдет, например, если на компьютере установлено меньше 1 Гб оперативной памяти или в момент запуска виртуальной машины запущено много других программ. В последнем случае, проблема может быть решена завершением работы временно ненужных запущенных программ. Если же мало физической памяти, то через настройки виртуальной машины (кнопка Settings в консоли) можно попробовать несколько уменьшить размер памяти, выделяемой виртуальной машине. Но надо помнить, что минимально рекомендуемый объем памяти для Windows Server 2008 – это 512 МБ и сильно урезать его без потери работоспособности ОС не получится.

Шаг 2. Настройка виртуальной машины

Среди прочих настроек виртуальной машины возможно отметить параметры сети (рис. 1.3).

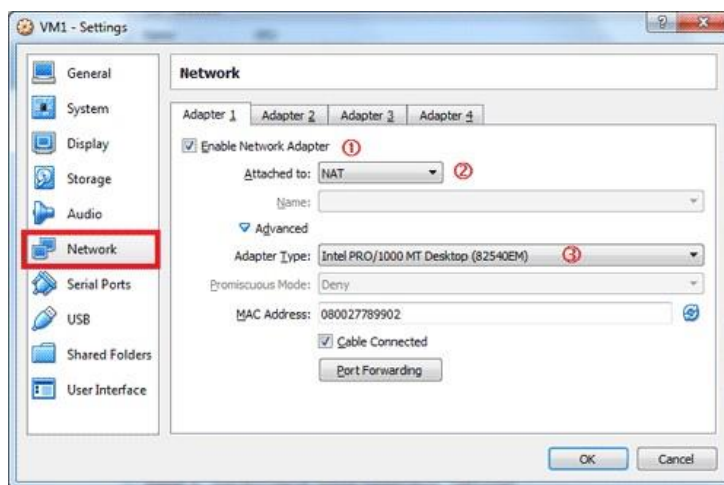


Рисунок 1.3 – Настройка сетевых адаптеров виртуальной машины

Выбрав раздел «Network», можно указать число адаптеров виртуальной машины, а также параметры для каждого адаптера. В частности, можно указать, какой сетевой адаптер физического компьютера будет задействоваться виртуальной машиной. Для локальных соединений (в рамках одного компьютера) можно выбрать тип Local only. А для первого сетевого адаптера также можно указать тип Shared networking. В этом случае используется механизм трансляции сетевых адресов NAT, виртуальный сетевой адаптер получает адрес из зарезервированного диапазона 192.168.x.y

и может «обращаться» к внешним узлам, используя базовую ОС, как NAT-устройство. Этот вариант позволит, например, получить из виртуальной машины доступ в Интернет, если на «физической» машине используется dialup соединение.

Остановимся на конфигурации с одним сетевым адаптером и типом соединения Shared networking.

Шаг 3. Установка Windows Server 2008

Следующая задача – установить операционную систему на виртуальную машину. Если Ваш дистрибутив Windows Server 2008 находится на DVD диске, то установите его в привод и запустите подготовленную виртуальную машину кнопкой **Start** на консоли виртуальных машин. После сообщения с просьбой нажать любую клавишу для подтверждения загрузки с CDROM начнется установка.

В случае, когда используется файл *.iso с образом дистрибутивного диска, последовательность действий будет несколько отличаться. Запустите виртуальную машину и в меню CD выберите пункт «Capture ISO image» (подключить образ диска), после чего укажите расположение файла с образом диска и при необходимости перезапустите машину (меню Action, пункт Reset).

Итак, установка началась. Для имеющих опыт установки операционных систем Microsoft, процесс установки Windows Server 2008 особых сложностей представлять, скорее всего, не будет. Первое окно мастера установки позволяет выбрать язык, региональные настройки и раскладку клавиатуры.

Дальше, в зависимости от версии дистрибутива, будет предложено ввести серийный номер (который, в частности, и укажет какую версию Windows Server 2008 надо устанавливать) или сразу будет показано окно выбора устанавливаемой операционной системы. Дело в том, что с одного и того же дистрибутивного диска может быть установлен Windows Server 2008 в версиях Standard, Enterprise или Datacenter. Каждая из трех версий может быть установлена в режиме полной установки (Full) или установки основных компонентов (Server Core). Режим Server Core – одно из новшеств, появившихся в операционной системе Windows Server 2008. Он позволяет путем отказа от установки средств графического администрирования и ряда модулей операционной системы получить «специализированную» конфигурацию сервера для выполнения ряда функций (файловый сервер, контроллер домена и т.д.), более защищенную и требующую меньшего внимания администратора. Для наших лабораторных работ устанавливаем версию Enterprise (Full installation) и отмечаем галочкой «I have selected the edition of Windows that I purchased» (Выбрана приобретенная версия Windows).

Далее интерес представляет окно выбора раздела для установки. В нем будет отображаться диск виртуальной машины (если помните, мы его задавали размером 40 Гб). Если нажать на ссылку Drive options (Advanced) появятся дополнительные команды, позволяющие в частности, разбить диск на разделы. Нужно отметить, что Windows Server 2008 можно установить только в раздел отформатированный в NTFS. Создайте два логических диска размером 30 Гб (на который будет устанавливаться операционная система) и 10 Гб.

В процессе установки операционная система несколько раз перезагрузится. В самом конце будет выдан запрос на установку пароля администратора. Чтобы в дальнейшем было удобнее работать, предлагается на всех устанавливаемых в классе виртуальных машинах назначить пароль **Serv08Saiu** (по-английски, с соблюдением указанной очередности больших и малых букв).

Для первого входа пользователя понадобится знание следующих команд.

Шаг 4. Изучение команд виртуальной машины

Вместо сочетания клавиш «Alt+Ctrl+Del» надо нажимать «правый Alt+Del».

«Правый Alt+Enter» – переключение между обычным и полноэкранным режимом.

Захваченный виртуальной машиной указатель мыши освобождается нажатием на правую клавишу Alt и выводом курсора из окна виртуальной машины. После установки расширений (см. ниже) эти действия больше не понадобятся, и указатель мыши будет свободно перемещаться между окном виртуальной машины и остальной областью экрана.

После первого входа в систему пользователь увидит окно начальной настройки системы Initial Configuration Tasks. Выглядит оно примерно так, как представлено на рис. 1.4.

В нем в секции 1 (Provide Computer Information) проверьте, правильно ли выставлен часовой пояс: для нас это (GMT+03:00) Москва, Санкт-Петербург, Волгоград. Кроме того, в разделе Provide computer name and domain вы увидите автоматически сгенерированное имя компьютера и рабочую группу Workgroup. Рабочую группу пока оставим как есть, а имя изменим на **S08**.

Итак, мы подготовили виртуальную машину с Windows Server 2008, которую назвали **S08**, и назначили учетной записи **Administrator** пароль **Serv08Saiu**.

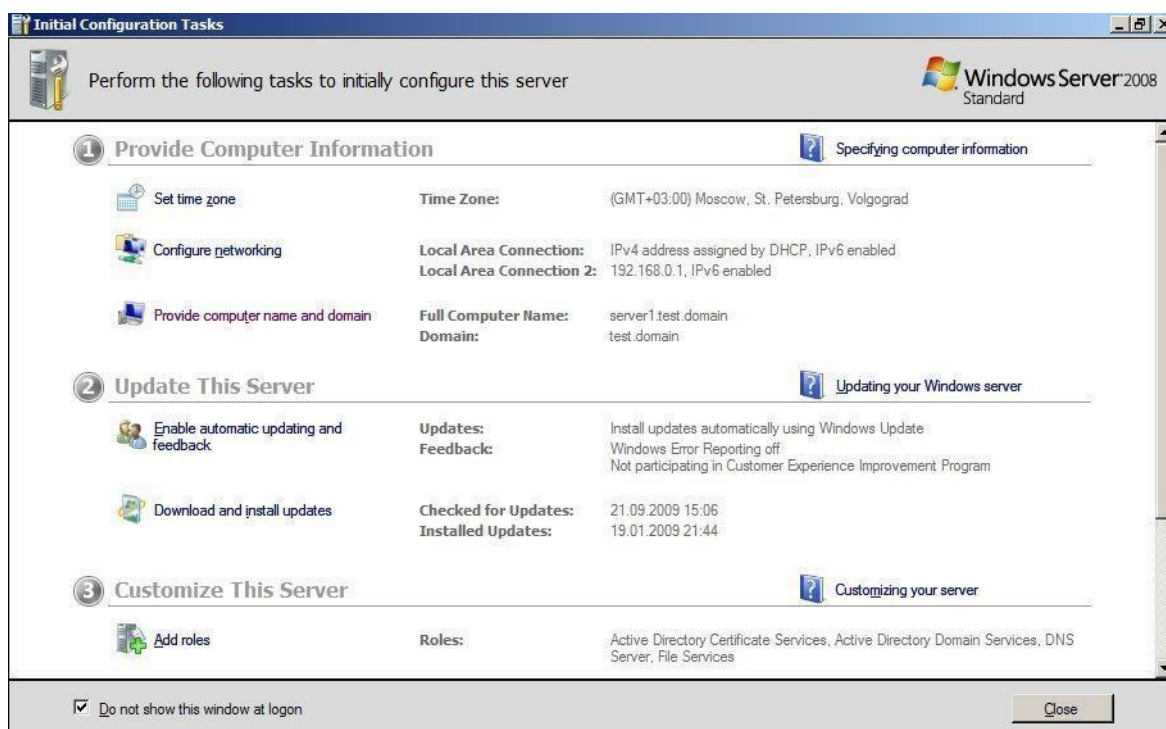


Рисунок 1.4 – Окно Initial Configuration Tasks

Шаг 5. Установка расширения на сервер

Последним пунктом в выполнении данной лабораторной работы является установка на сервер расширения, что добавит ряд удобств в работе с виртуальной машиной. В частности, это передача данных между базовой ОС и гостевыми через буфер обмена, возможность подключения папки на диске «физической» машины в качестве сетевого диска виртуальной машины, возможность свободного перехода указателя из области окна виртуальной машины в область рабочего стола базовой ОС и т.д.

В меню Action окна виртуальной машины выберите пункт Install or Update Virtual Machine Additions (установите или обновите расширения виртуальной машины). Если виртуальная машина загружена и пользователь совершил вход, то в CDRом виртуальной машины будет автоматически подключен «псевдообраз» диска с инсталляционным пакетом расширений. Останется только согласиться с автоматическим запуском программы setup и дождаться окончания установки.

Лабораторная работа №2. Управление загрузкой серверной ОС. Добавление ролей. Установка первого контроллера домена.

1 Цель работы

Научиться:

- добавлять и настраивать роли серверной ОС;
- устанавливать и настраивать контроллер домена.

2 Краткая теория и порядок выполнения работы

Используемое программное обеспечение

Для выполнения данной лабораторной работы понадобится установленный на виртуальную машину Windows Server 2008.

Шаг 1. Знакомство с инструментами управления загрузкой ОС.

Параметры загрузки. Утилита bcdedit. Утилита System Configuration.

Начнем выполнение данной лабораторной работы со знакомства с новыми инструментами управления загрузкой операционной системы. В Windows Server 2003 (также, как и в предыдущих версиях клиентских и серверных операционных систем семейства Windows NT) использовался загрузчик NT loader, который считывал настройки из файла boot.ini:

```
[boot loader] timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS [operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
Professional RU" /noexecute=optin /fastdetect
```

Изменять файл (и параметры загрузки) можно было прямо в текстовом редакторе. В Windows Vista и Windows Server 2008 используется диспетчер загрузки Windows Boot Manager. Параметры загрузки хранятся в бинарном файле конфигурации (Boot Configuration Data – BCD) расположенном в скрытой системной папке с именем \boot на диске, с которого загружается операционная система. Для работы с BCD используется утилита командной строки bcdedit. Некоторые параметры доступны из графической утилиты конфигурирования System Configuration и окна System в панели Управления.

Рассмотрим графические утилиты.

Откройте Панель управления (*Start-> Control Panel*), при необходимости переключитесь к классическому виду (*Classic View*) и откройте окно *System*. В нем перейдите по ссылке *Advanced System Setting* и в разделе *Startup and Recovery* нажмите кнопку *Settings*. В появившемся окне (рис.2.1) можно увидеть список загружаемых операционных систем, выбрать операционную систему, загружаемую по умолчанию, установить интервал ожидания выбора пользователем системы для загрузки.

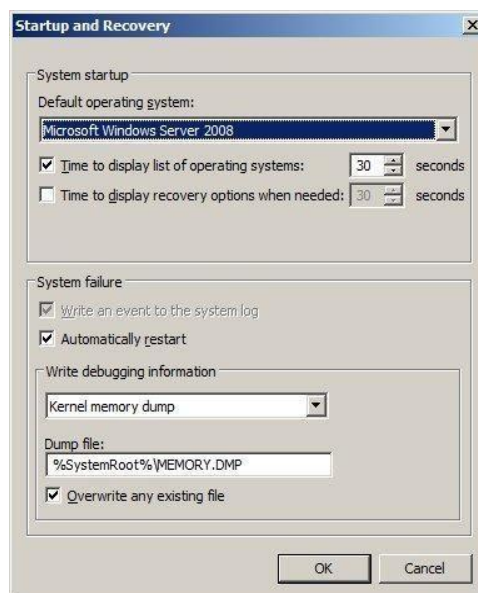


Рисунок 2.1 – Окно Startup and Recovery

Рассмотрим следующую утилиту – *System Configuration* (запускается из раздела *Administrative Tools* главного меню). На вкладке boot (рис.2.2) можно увидеть и изменить дополнительные параметры, например, указать что в определенном варианте загрузки надо вести протокол загрузки (флажок Boot log). Кнопка *Advanced options...* позволяет ограничить число процессоров, используемых ОС, объем оперативной памяти и установить некоторые параметры, необходимые для отладки.

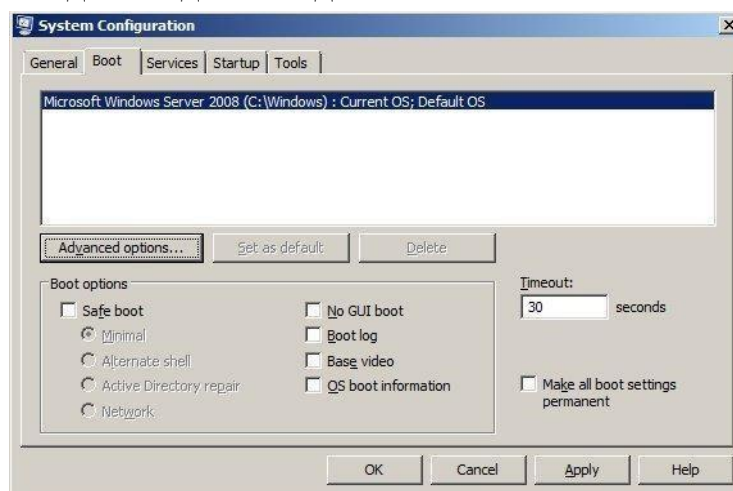


Рисунок 2.2 – Вкладка Boot окна утилиты System Configuration

Примечательно, что создавать новые варианты загрузки можно только с помощью утилиты командной строки *bcdedit*. Запустим ее (Start->Command Prompt -> *bcdedit.exe*). При запуске без ключей будет выведена информация о текущей конфигурации, например, следующего вида:

```
Windows Boot Manager
identifier           {bootmgr}
```

```

device          partition=C:
description     Windows Boot Manager
locale          en-US
inherit         {globalsettings}
default         {current}
displayorder    {current}
toolsdisplayorder {memdiag}
timeout30 Windows Boot Loader

```

```

.....
identifier      {current}
device          partition=C:
path            \Windows\system32\winload.exe
description     Microsoft Windows Server 2008
locale          en-US
inherit         {bootloadersettings}
osdevice        partition=C:
systemroot      \Windows
resumeobject    {51c6bea1-e439-11dd-bb9b-ed314082ca2a}nx OptOut

```

Здесь видно, что в файле содержатся две записи – одна для самого диспетчера загрузки (Windows Boot Manager), другая – для операционной системы Windows Server 2008.

Запуск `bcdedit.exe /?` позволит получить краткую справку по ключам команды. Вот некоторые из них:

<code>/enum</code>	Отображает список всех имеющихся в BCD файле записей. Запуск утилиты без ключей эквивалентен запуску " <code>bcdedit /enum ACTIVE</code> "
<code>/v</code>	Отображает идентификаторы записей в полном виде, вместо использования «хорошо известных» идентификаторов. Например, вместо идентификатора <code>{current}</code> будет что-нибудь вида <code>{51c6bea0-e439-11dd-bb9b-ed314082ca2a}</code>
<code>/copy</code>	Копирует запись
<code>/create</code>	Создает новую запись
<code>/delete</code>	Удаляет запись
<code>/export</code>	Экспортирует содержимое файла BCD в указанный файл
<code>/import</code>	Восстанавливает содержимое BCD из указанного файла
<code>/set</code>	Устанавливает значения параметра для записи

Теперь выполним следующие действия.

1. Создадим копию файла BCD в файл `bcd.bak` в корне диска C:

`bcdedit.exe /export "C:\bccopy"`

Убедимся, что в отличие от `boot.ini` при открытии файла в текстовом редакторе осмысленного текста мы не увидим.

2. Добавим к файлу копию записи с идентификатором `{current}` и назовем ее `test08 bcdedit.exe /copy {current} /d "test08"`. Убедимся, что новая запись появилась.

3. Чтобы было быстрее, из утилиты System Configuration для созданного варианта загрузки укажем, что нужно выполнить загрузку в безопасном режиме (Safe boot, переключатель minimal). Перезагрузим, протестируем работу настройки.

4. Командой `bcdedit.exe /import "C:\bccopy"` вернем файл BCD в начальное состояние.

Шаг 2. Установление, добавление, удаление ролей и компонентов

Сейчас у нас имеется установленная операционная система Windows Server 2008, но для того, чтобы использовать ее для выполнения некоторой серверной функции надо установить роль (role). Роль включает одну или несколько служб (role services), необходимых для выполнения определенной функции. Например, File Services (файловые службы) или Web-server (IIS). Когда роль объединяет несколько служб, то могут устанавливаться или все сразу, или отдельные службы. Дополнительная функциональность может быть получена путем установки программных модулей, называемых компонентами (feature). Пример компоненты – это SMTP Server. Роли и компоненты могут быть как независимыми, так и взаимосвязанными.

Добавить или удалить роли и компоненты можно с помощью оснастки Server Manager (Start -> Administrative tools -> Server Manager). Окно оснастки представлено на рис.2.3.

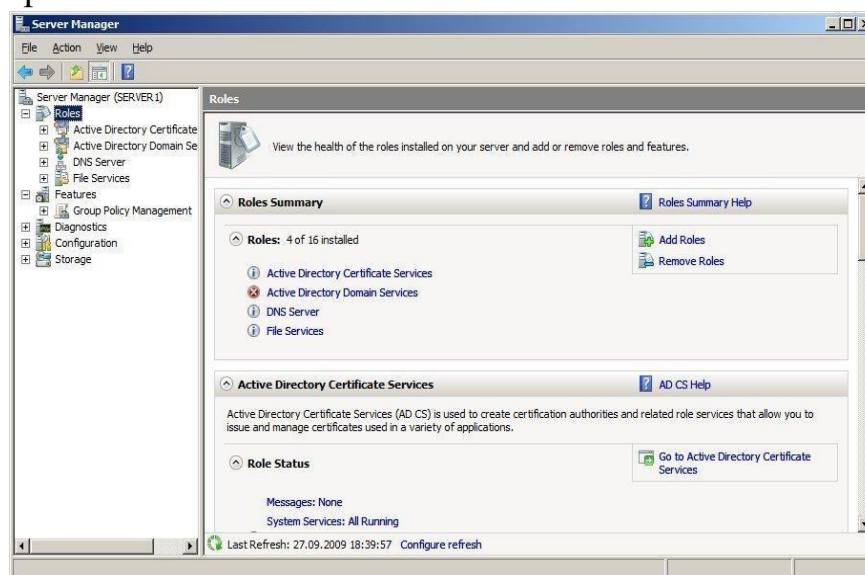


Рисунок 2.3 – Окно административной оснастки Server Manager

Если выделить узел Roles, то увидим список установленных ролей. Добавлять или удалять роли можно, перейдя по ссылкам Add Roles или Remove Roles соответственно. Добавьте роль File Services (не устанавливая дополнительных служб, таких как DFS и т.д.). Теперь наш сервер сможет выполнять роль файлового сервера. Работе с файловыми ресурсами будет посвящена отдельная лабораторная работа. Сейчас же стоит отметить только, то что в данном случае никакого дополнительного конфигурирования после установки роли не потребовалось.

Одна из задач лабораторной работы – сделать сервер контроллером домена Windows. Для этого понадобится установить роль Active Directory Domain Services и выполнить настройку параметров домена.

Домен Windows логически объединяет несколько компьютеров для того, чтобы можно было их централизованно администрировать. Примером административной задачи может быть создание такой учетной записи, чтобы пользователь мог входить под ней, на любой компьютер своего подразделения организации. В этом случае, чтобы такую запись завести только один раз (а не на каждом компьютере), нужно вести единую базу данных с информацией о пользователях и компьютерах. Подобная база называется каталогом, а разработанная Microsoft служба каталога – Active Directory. Серверы, на которых работает служба и которые выполняют проверку пользователей с доменными учетными записями, называются контроллерами домена.

Из оснастки Server Manager добавляем роль Active Directory Domain Services (рис.2.4).

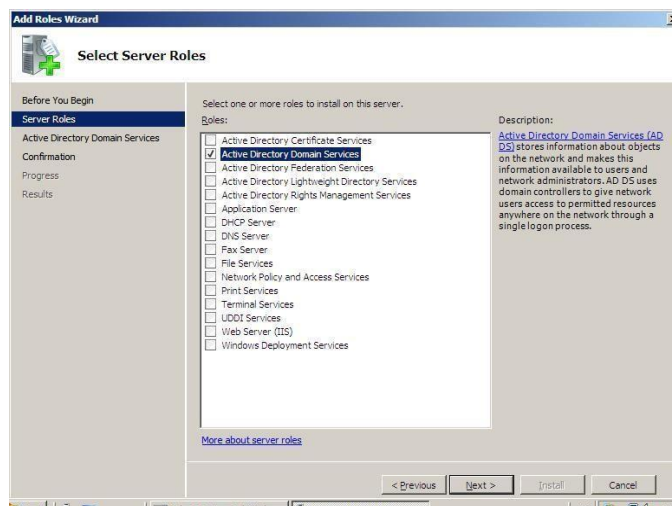


Рисунок 2.4 – Добавление роли Active Directory Domain Services

Когда роль добавлена, потребуется произвести начальное конфигурирование нового контроллера. Делается это запуском утилиты dcpromo из основного меню (Start->Run...-> dcpromo) или по ссылке, которая появится в окне Server Manager после установки роли.

Для понимания дальнейшего необходимо ввести ряд понятий. Как отмечалось, информация об объектах сети хранится в каталоге. Для этого сначала создаются определения объектов, которые помещаются в служебную структуру, называемую схема каталога. Если хотим создать объект нового типа, нужно сначала поместить в схему его определение. Совокупность доменов, использующих единую схему каталога и общую конфигурацию, называют лесом доменов (forest).

В окне мастера, представленном на рис. 2.5, указываем, что создаем новый лес, т.е. тем самым, конфигурируем первый контроллер первого (корневого) домена в нашей организации.

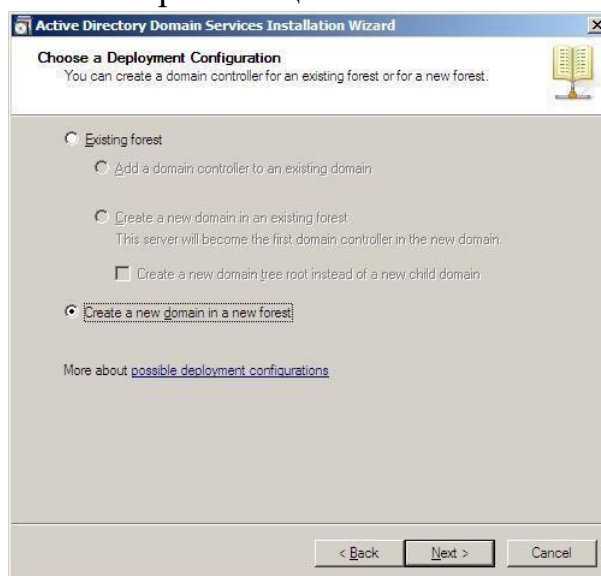


Рисунок 2.5 – Создание нового домена в новом лесе

В следующем окне мастера (рис.2.6) запрашивается имя домена. Обычно имена соотносятся с доменными именами Интернет (например, ftk.spbstu.ru), но для наших лабораторных будем использовать имя **SAIU_Test**.

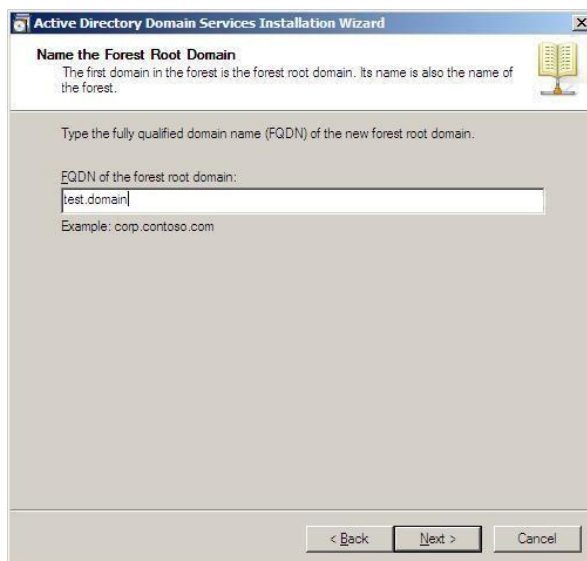


Рисунок 2.6 – Ввод имени домена

Далее, так как в виртуальной сети пока нет DNS сервера, мастер предложит установить DNS сервер (рис.2.7).

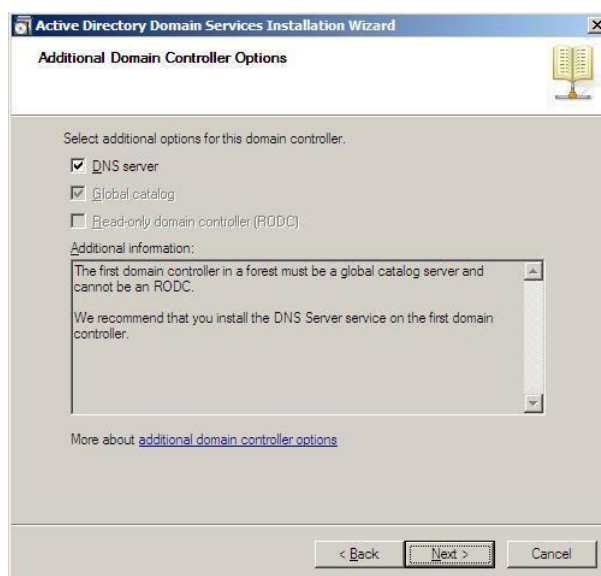


Рисунок 2.7 – Установка DNS сервера

Служба DNS используется для разрешения доменных имен компьютеров в ip-адреса. В домене Windows клиентские компьютеры с помощью DNS получают информацию о контроллерах домена (более подробно об этом в следующих лабораторных), поэтому хотя бы один DNS сервер необходим.

Следующее окно позволяет выбрать функциональный уровень домена (рис. 2.8). Если выбрать уровень Windows Server 2008, то мы получим поддержку всех новых функций, но в роли контроллеров домена смогут выступать только компьютеры с Windows Server 2008 (и, наверное, более новых ОС после их появления).

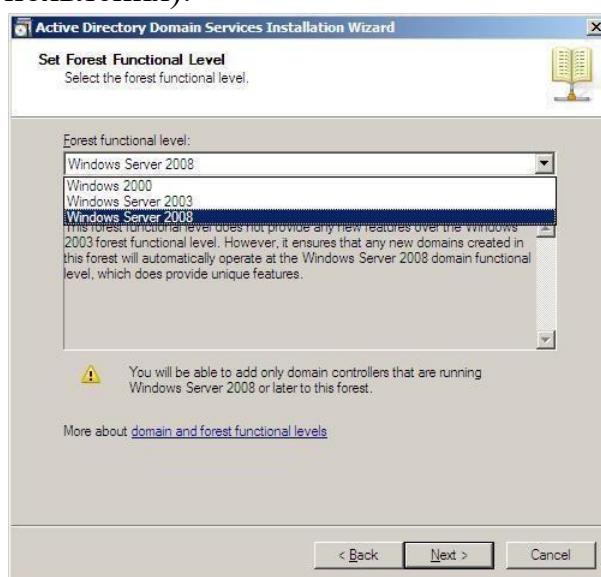


Рисунок 2.8 – Выбор функционального уровня домена

И наоборот, выбор уровня Windows 2000 позволит использовать контроллеры с более старыми версиями серверных ОС (Windows 2000 Server, Windows Server 2003), но не даст использовать некоторые возможности, имеющиеся только в Windows Server 2008. Для лабораторных выберем уровень Windows Server 2008.

Следующее окно мастера позволяет выбрать расположение базы каталога, файлов журнала и служебного каталога SYSVOL (в нем, в частности, хранятся политики и скрипты, запускающиеся при входе пользователя в домен). Можно согласиться с настройками по умолчанию.

В конце надо будет задать пароль для доменной учетной записи **Administrator** (пусть это снова будет **Serv08Saiu**) и перезагрузить нашу виртуальную машину. После перезагрузки зайдите под учетной записью **Administrator**.

Шаг 3. Создание нового пользователя

Для этого нужно запустить административную оснастку Active Directory Users and Computers (Start-> Administrative Tools-> Active Directory Users and Computers) или можно получить к ней доступ из окна Server Manager, как показано на рис.2.9.

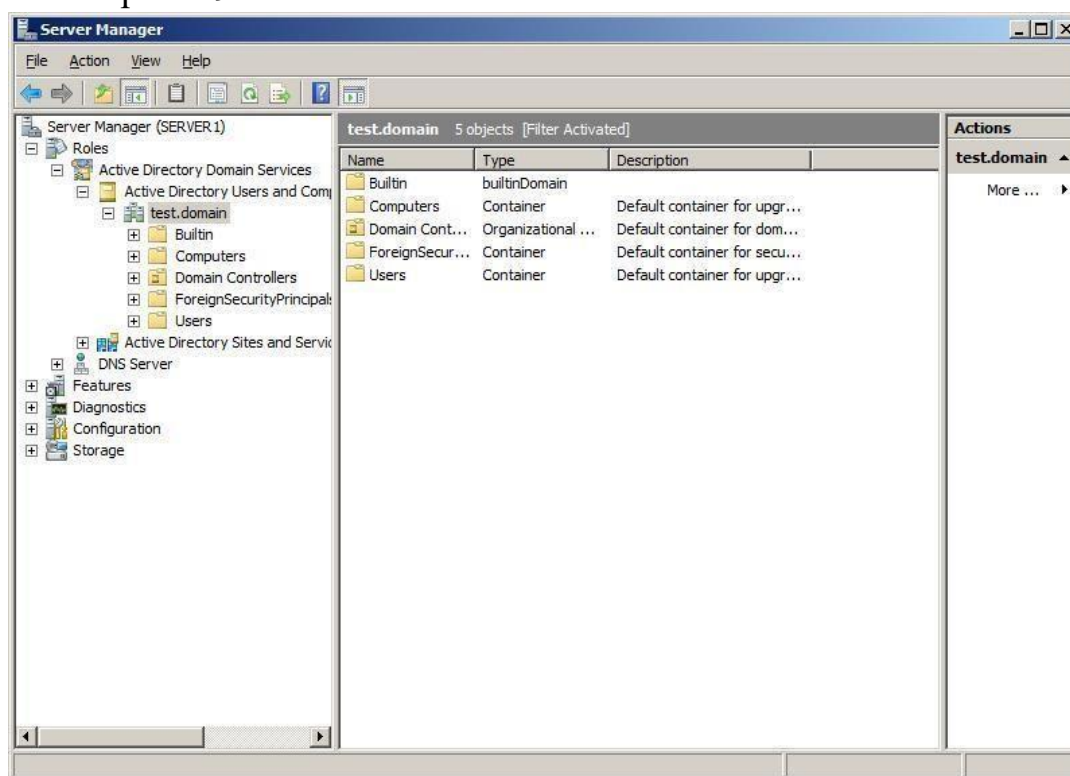


Рисунок 2.9 – Управление доменными учетными записями

Раскройте узел, соответствующий домену, перейдите на контейнер Users и вы увидите список созданных автоматически стандартных групп и учетных записей.

В контекстном меню выберите создание нового пользователя (New>User) пусть имя учетной записи будет **Labos**, пароль **Lab0s123** (имя и фамилию пользователя придумайте сами). Укажите, что пользователю не надо менять пароль при первом входе.

Когда учетная запись создана, попробуйте войти под ней на сервер. Должно появиться сообщение, что действующая политика не позволяет это сделать. Дело в том, что настройки по умолчанию не позволяют обычным пользователям локально входить на контроллер домена.

Созданная учетная запись нам понадобится на следующей лабораторной работе, когда мы добавим в домен рабочую станцию.

Лабораторная работа №3. Основы администрирования домена: добавление компьютера в домен, работа с учетными записями и группами.

1 Цель работы

Освоить основы администрирования домена: добавление компьютера в домен, работа с учетными записями и группами.

2 Краткая теория и порядок выполнения работы

Используемое программное обеспечение

Для выполнения данной лабораторной работы понадобится компьютер с операционной системой Microsoft Windows, или Linux Mint и установленным программным обеспечением Oracle VM VirtualBox, подготовленные виртуальные машины с Windows Server 2008 и Windows Vista.

В ходе работы виртуальная машина с Windows Vista будет добавлена в домен, доменным контроллером которого выступает Windows Server 2008.

Шаг 1. Настройка параметров сети для виртуальных машин

Для выполнения работы нам понадобится настроить параметры сетевых подключений виртуальной рабочей станции и сервера. В случае если компьютер, на котором выполняется лабораторная работа, не позволяет одновременно запустить две виртуальные машины (сервер и рабочую станцию), то для выполнения лабораторной понадобятся два компьютера, подключенные к сети. Эту конфигурацию и рассмотрим. Сначала, до запуска виртуальных машин, сделаем настройки, указывающие, что виртуальная машина будет использовать физический сетевой адаптер. В консоли управления Oracle VM VirtualBox в свойствах виртуальной машины найдем пункт Networking и выберем для использования один из сетевых адаптеров нашего компьютера (рис.3.1).

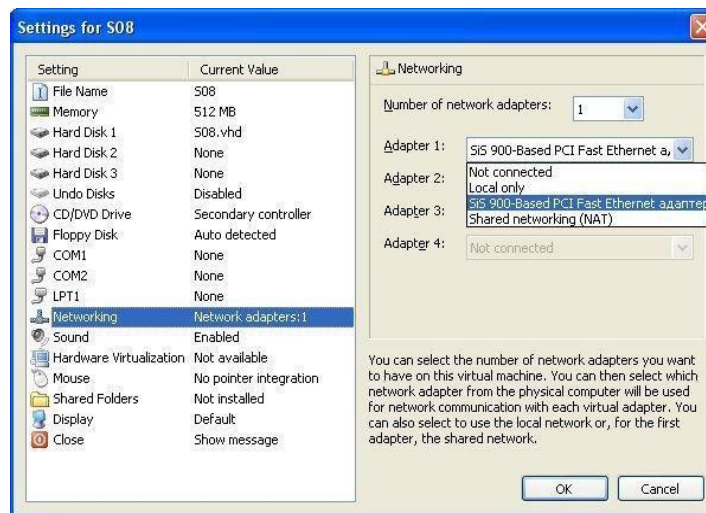


Рисунок 3.1 – Выбор «физического» сетевого адаптера в свойствах виртуальной машины

После этого предлагается студентам разделиться на пары. На одном компьютере запускается виртуальная машина с Windows Server 2008, сконфигурированном для работы в качестве контроллера домена (у нас он назвался S08). На другом компьютере запускаем виртуальную машину с Windows Vista. Входим с правами администратора. Для S08 это учетная запись **Administrator** с паролем **Serv08Saiu**. Для Vista_dor учетная запись **Adm** с паролем **123qwe**.

На виртуальных машинах настроим протокол TCP/IP v4 таким образом, чтобы сервер смог взаимодействовать с рабочей станцией, не мешая проведению лабораторных другими парами. Для этого, откроем параметры сетевого подключения на рабочей станции и сервере. Например, таким образом Start->Control Panel->(при необходимости переключаемся к классическому виду Classic View) Network and Sharing Center -> ссылка Manage Network Connections (Пуск-> Панель управления-> Центр управления сетями и общим доступом->Управление сетевыми подключениями). Находим наше подключение (оно должно быть единственным), открываем его свойства, находим протокол TCP/IP v.4 и кнопкой Properties открываем окно настроек для этого протокола (рис.3.2).

Переключателями укажите, что настройка параметров будет осуществляться вручную (переключатель Use the following IP address).

Для сервера задайте:

Ip адрес 192.168.1yy.1, где yy –двухзначный порядковый номер компьютера в классе (192.168.101.1 для первого);

маска 255.255.255.0;

шлюз - <оставляем пустым> DNS 127.0.0.1.

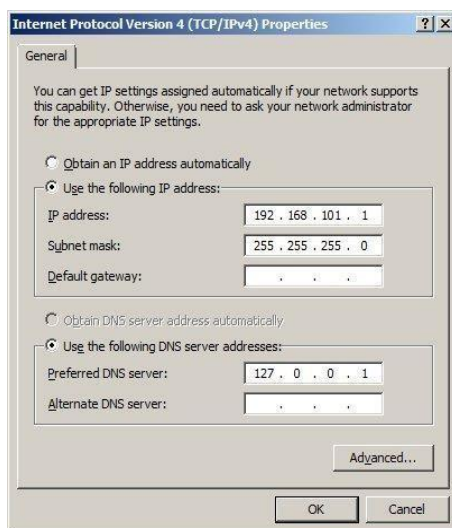


Рисунок 3.2 – Настройка параметров TCP/IP

Для рабочей станции, которая будет подключаться к данному серверу:

Ip адрес – 192.168.1yy.2, где yy – номер, такой же как у сервера (192.168.101.2 для рабочей станции, подключающейся к серверу с адресом 192.168.101.1);

маска – 255.255.255.0;

шлюз – 192.168.1yy.1;

DNS – 192.168.1yy.1.

Таким образом, у нас сервер и рабочая станция и сервер окажутся в одной ip-сети, а другие пары «сервер - рабочая станция» – в других. Чтобы проверить правильность сделанных настроек с рабочей станции выполните команду

ping 192.168.1yy.1 (yy - в зависимости от номера пары, например, ping 192.168.101.1)

Если будет получен ответ, то все сделано правильно, сервер доступен.

Итак, у нас в одной сети есть сервер-контроллер домена и рабочая станция, в домен пока не включенная. Перед выполнением следующих заданий проверьте на рабочей станции состав групп Администраторы и Пользователи (Пуск->Панель управления->(классический вид) Администрирование -> Управление компьютером -> Локальные пользователи и группы).

На сервере откроем оснастку администрирования Active Directory Users and Computers (Start->Administrative tools) и убедимся, что в контейнере Computers ничего нет. Кстати, а где сам сервер S08, он же член домена?! Контроллеры домена находятся в контейнере Domain Controllers.

Шаг 2. Добавление домена

Теперь, администрируя рабочую станцию, добавим ее в домен.

Для этого откроем Панель управления, переключимся к классическому виду и запустим инструмент *Система*. В открывшемся окне найдите раздел *Имя компьютера, имя домена, параметры рабочей группы*, выберите ссылку *Изменить параметры*. В открывшемся окне на вкладке *Имя компьютера* нажмите кнопку *Изменить*. И укажите, что теперь компьютер включен в домен (он у нас называется *saiu_test*). Для завершения этой операции понадобится логин и пароль учетной записи, обладающей правами добавлять пользователей в домен. Это учетная запись *Administrator*. Чтобы явно указать, что учетная запись доменная, используйте полное имя в виде *saiu_test\Administrator*.

После добавления в домен рабочую станцию, необходимо выполнить перезагрузку. В это время проверьте на сервере, что в домене появился новый компьютер.

Войдите на рабочую станцию сначала под доменной учетной записью **Labos** (пароль **Lab0s123** – эту запись мы создавали на прошлой лабораторной работе), потом под локальной записью **Адм**. Посмотрите, как изменился состав локальных групп Пользователи и Администраторы.

На сервере откройте оснастку *Active Directory Users and Computers*. В контейнере *Users* найдите учетную запись **Labos**. Откройте ее свойства (в контекстном меню пункт *Properties*), на вкладке *Account* найдите кнопку *Logon Hours ...* (рис.3.3).

Отредактируйте параметры таким образом, чтобы время выполнения лабораторной работы не попадало в период, разрешенный для входа пользователя. Попробуйте на рабочей станции зайти под пользователем **Labos**.

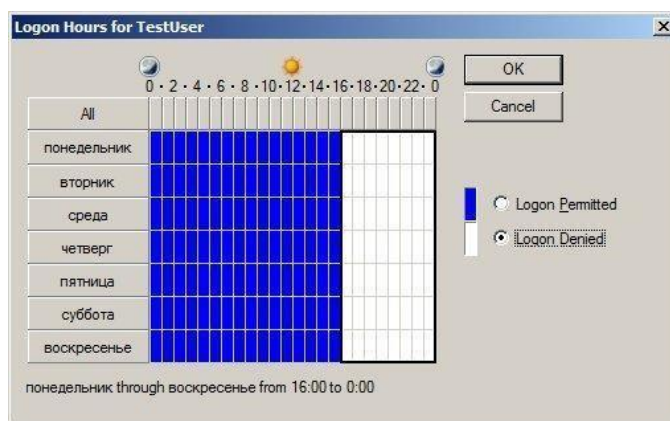


Рисунок 3.3 – Редактирование расписания работы пользователя

Шаг 3. Составление расписания пользователя

В завершение работы, сделайте расписание таким, чтобы пользователь **Labos** мог входить в домен в это время, но при этом, ограничьте перечень

компьютеров, на которые может входить пользователь (вкладка Account кнопка Logon to...) и исключите рабочую станцию Vista_dor. Проверьте работу настройки.

Верните назад сделанные изменения.

При необходимости, поменяйтесь в паре и проделайте лабораторную работу для другого сочетания сервер-рабочая станция.

Лабораторная работа №4. Администрирование файлового сервера.

1 Цель работы

Освоить основы администрирования файлового сервера.

2 Краткая теория и порядок выполнения работы

Используемое программное обеспечение

Для выполнения данной лабораторной работы понадобится компьютер с операционной системой Microsoft Windows или Linux Mint и установленным программным обеспечением Oracle VM VirtualBox, а также две подготовленные виртуальные машины с Windows Server 2008 и Windows Vista.

Шаг 1. Дополнительная настройка виртуальных машин

В ходе выполнения лабораторной работы нам понадобится добавить диски к виртуальной машине. Для этого в консоли VirtualBox нужно выделить виртуальную машину и кнопкой Settings открыть ее параметры (рис. 4.1).

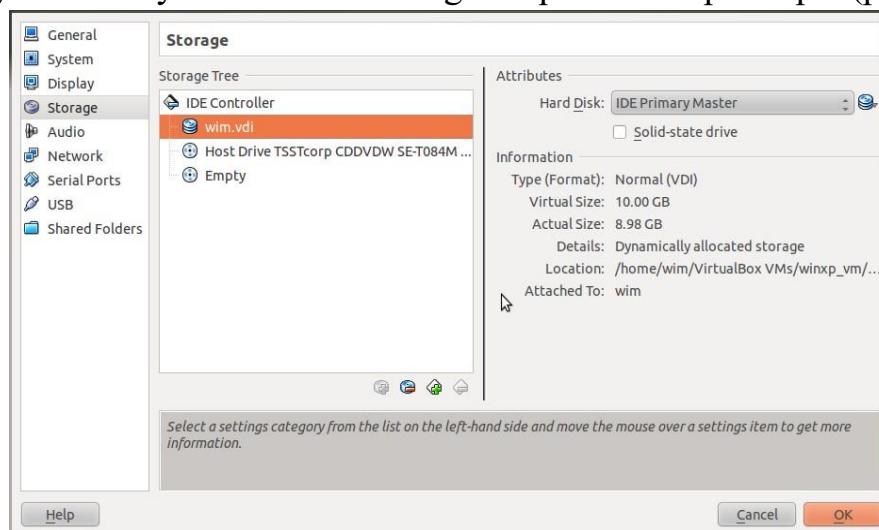


Рисунок 4.1 – Окно параметров виртуальной машины

Найдите список жестких дисков и на месте отсутствующего диска (надпись «None») с помощью мастера, запускаемого нажатием кнопки Virtual Disk Wizard, создайте новый диск. Процедура достаточно понятна и не требует

особых пояснений. При выборе типа диска оставьте настройку по умолчанию Dynamically expanding (динамически увеличивающийся). Для выполнения работы понадобится подключить два диска размером по 10 Гб.

Также иногда требуется подключить к виртуальной машине папку с «физического» компьютера. В виртуальной машине она будет отображаться как сетевой диск, что позволит осуществлять обмен файлами.

Для этого в параметрах уже запущенной виртуальной машины надо зайти в раздел Shared Folders (рис.4.2) и, нажав кнопку Share Folder подключить к виртуальной машине выбранную папку.

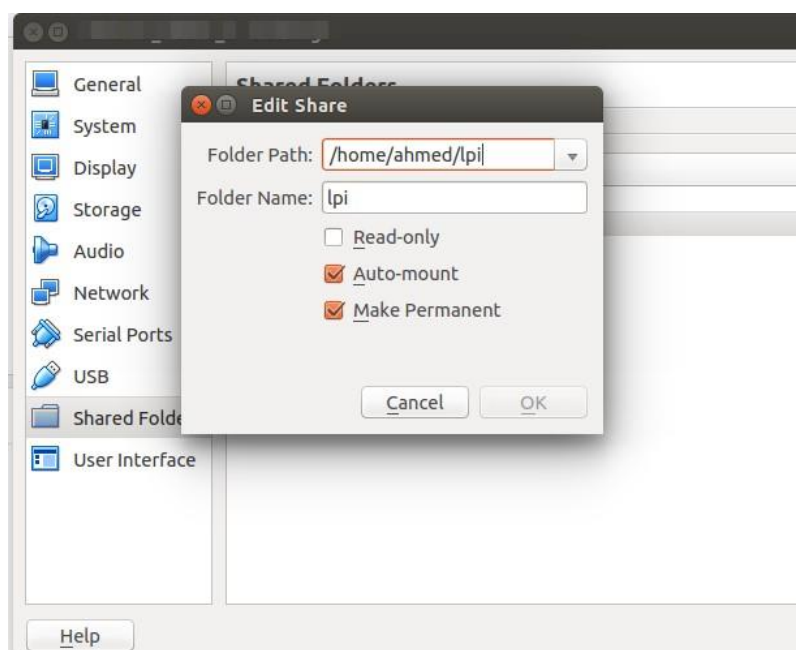


Рисунок 4.2 – Подключение папки к виртуальной машине

Шаг 2. Управление дисками

Рассмотрим ряд практических вопросов, связанных с организацией дискового пространства в Windows Server 2008. Для работы с дисками запустим оснастку Server Manager и в ней откроем узел Storage и там Disk Management (рис.4.3).

Если вы добавили два новых виртуальных диска (см. предыдущий раздел), то при открытии Disk Management появится окно с предложением подключить новые диски к системе и выбрать их тип - MBR или GPT. Диски типа MBR («обычные» для персональных компьютеров на базе процессоров Intel и совместимых) могут иметь размер до 2 терабайт и поддерживают до 4 разделов на логический диск. Новый тип GPT используется для дисков большего объема и позволяет создавать до 128 разделов.

В данной лабораторной работе выберем тип MBR.

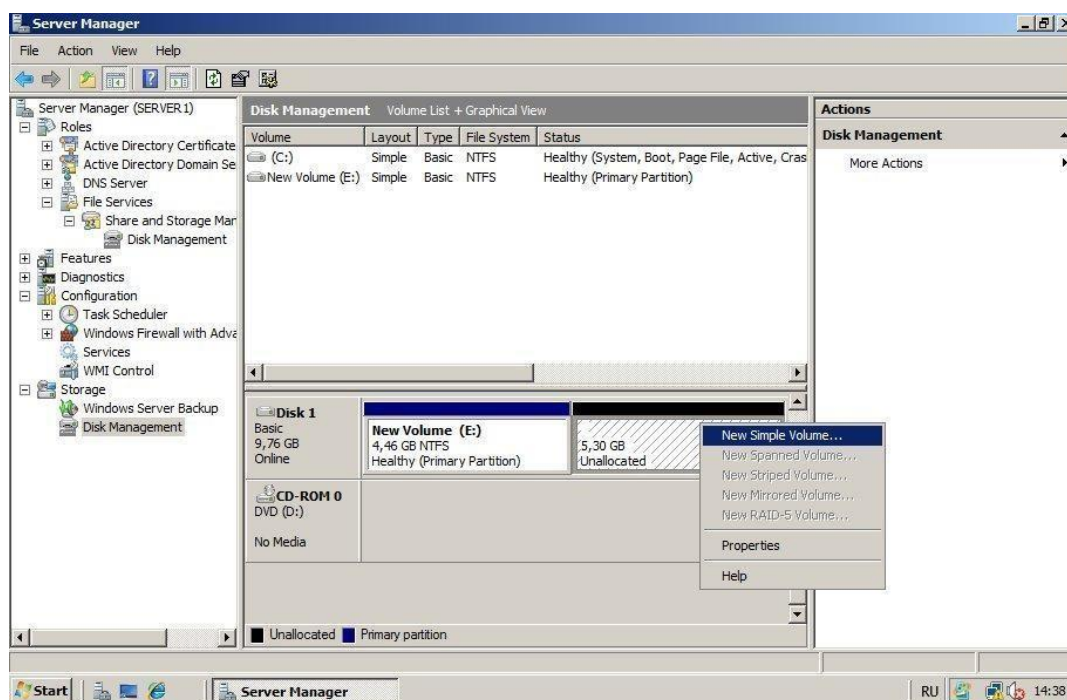


Рисунок 4.3 – Окно оснастки Disk Management

Посмотрите на созданный при установке операционной системы диск.

Если вы внимательно выполнили первую лабораторную работу, то сейчас должны увидеть физический диск с типом Basic, логическим диском C: на 30 гигабайт, куда установлена операционная система, и примерно 10 гигабайтами свободного пространства. Теперь в свободной области диска создадим новый раздел. Для этого в контекстном меню (рис.4.3) выберем пункт New Simple Volume, укажем, что все свободное место отводится под раздел, назначим букву (например, D:) и выберем форматирование в файловую систему NTFS (при этом размер кластера – Allocation Unit Size – лучше оставить по умолчанию, а метку тома можно изменить на более информативную, например, TEST DATA).

Таким образом, нами создан второй основной раздел на диске типа basic. В предыдущих версиях Windows Server оснастка Disk Management также позволяла создавать и расширенные разделы (extended partition). В начале основного раздела находится загрузочный сектор (boot sector) и с него может загружаться операционная система. Расширенный раздел загрузочного сектора не содержит, но может быть разделен на несколько логических дисков (основной раздел содержит только один логический диск). Windows Server 2008 поддерживает работу с расширенными разделами, но создавать их можно только с помощью утилиты командной строки diskpart.exe.

Говоря о новых возможностях Disk Management, нужно отметить возможность уменьшить (shrink) или увеличить (extend) размер существующего раздела диска. Для выполнения этих операций надо щелкнуть

на изображении раздела правой клавишей мыши и из контекстного меню выбрать соответствующую операцию. Расширить раздел на базовом диске можно только в том случае, если непосредственно за ним есть неразмеченная область диска.

Теперь перейдем к рассмотрению динамических дисков. Новые возможности, предоставляемые данным типом дисков – создание одного логического диска на нескольких физических, создание массивов дисков (RAID от англ. Redundant Array of Independent Disks массив независимых дисков с избыточностью). Итак, у нас в виртуальной машине появились два новых диска. По умолчанию они имеют тип Basic. Но их можно преобразовать в тип Dynamic воспользовавшись контекстным меню, появляющимся при щелчке правой клавишей мыши по названию диска (рис.4.4).

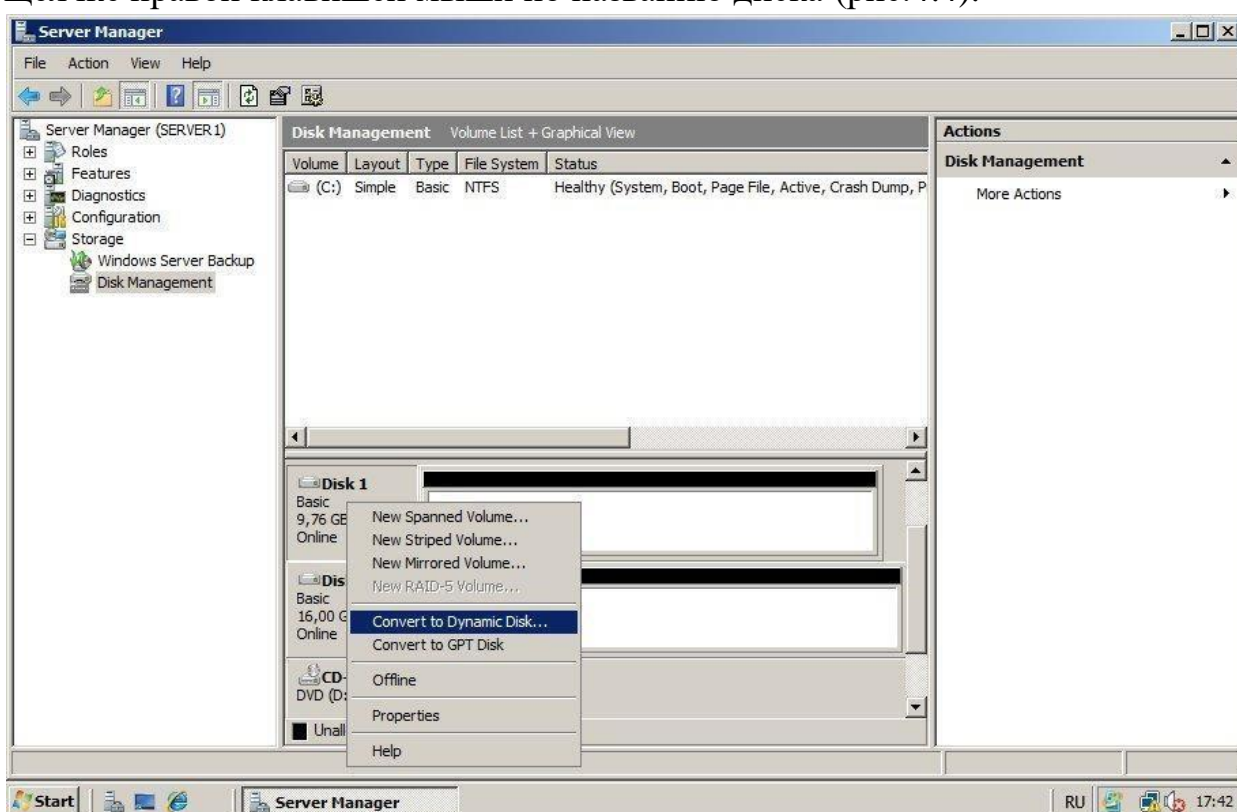


Рисунок 4.4 – Изменение типа диска с basic на dynamic

Безопасно преобразовать к динамическому типу можно любой базовый диск (в том числе, уже размеченный и содержащий данные). Обратное преобразование оснастка Disk Management осуществить не позволяет.

Преобразуем два новых диска к типу Dynamic. На одном из них создадим новый том (тип Simple Volume), занимающий все пространство диска, пусть он будет обозначен буквой E:. На новом логическом диске создайте какие-нибудь папки или файлы. Теперь предположим, что понадобилось увеличить объем логического диска. Для этого задействуем второй «физический диск»: в оснастке Disk Management щелкните правой

клавишей мыши на изображении диска E:, в контекстном меню выберите пункт Extend Volume и с помощью мастера добавьте все пространство второго диска к тому E:. Обратите внимание, что после этой операции тип тома с «простого» (simple volume) изменился на «составной» (spanned volume). Эта конфигурация позволяет более гибко управлять дисковым пространством, но не обладает отказоустойчивостью. Убедимся в этом на следующем примере.

Сначала проверим, что созданные нами папки и файлы остались на диске E:. Теперь выключим виртуальную машину и отключим третий виртуальный диск (на первом у нас операционная система и раздел для данных, а второй и третий диски занимал динамический том). После очередной загрузки виртуальной машины увидим, что диска, созданного нами на динамическом томе, в проводнике нет. Если открыть оснастку Disk Management, то увидим, что составной том неработоспособен (рис.4.5).

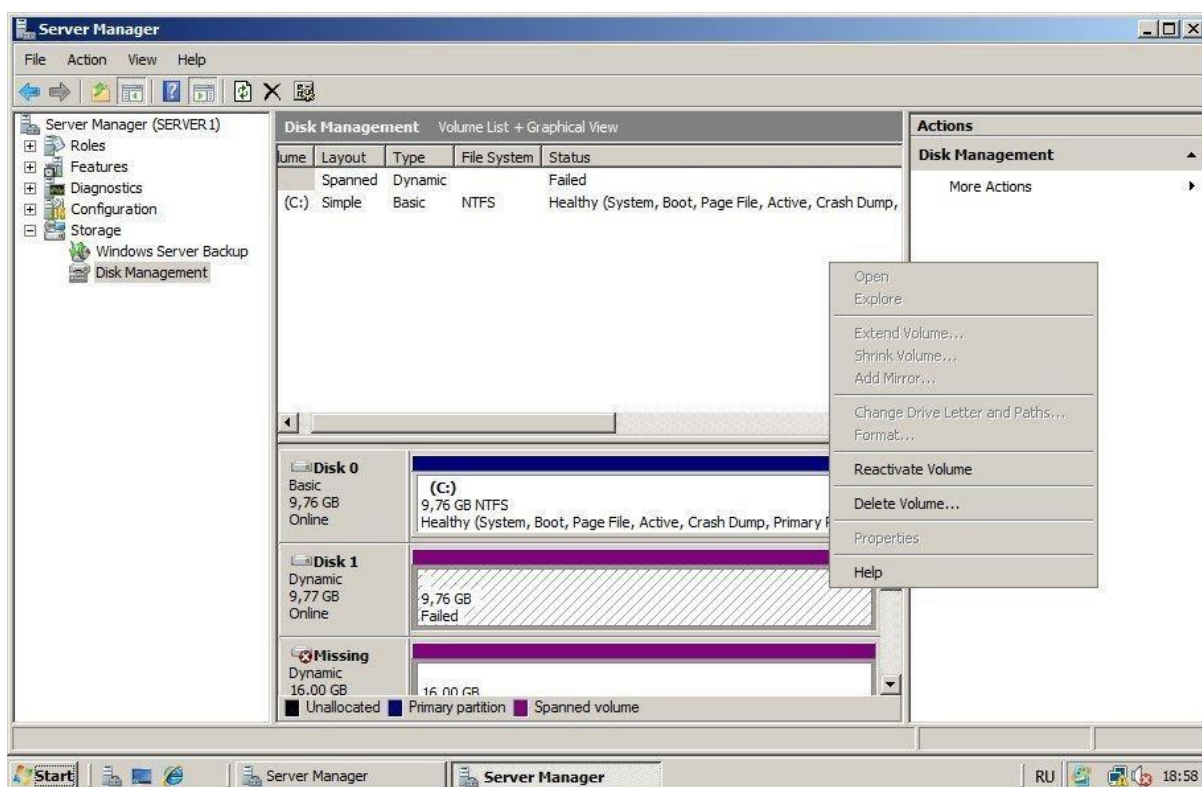


Рисунок 4.5 – Имитация отказа одного из дисков в составном томе

Его можно удалить, чтобы использовать дисковое пространство на работоспособном физическом диске, но это не позволит получить доступ к хранившимся там файлам.

Кроме рассмотренных простого и составного тома, при использовании нескольких физических дисков можно создавать следующие типы томов.

1) Чередующийся (stripped) – соответствует RAID 0, в массив объединяются несколько физических дисков, записываемые файлы делятся на

блоки, которые поочередно записываются на различные диски. За счет этого повышается скорость работы (операции распараллеливаются между несколькими физическими устройствами), но снижается надежность, т.к. выход из строя одного диска приводит к недоступности всех данных;

2) Зеркальный или зеркалируемый (mirrored) – соответствует RAID 1, в массив объединяются два диска, содержимое которых будет идентично - данные записываются одновременно на оба. При отказе одного из дисков, данные могут быть считаны со второго. Но плата за большую надежность – избыточность, т.к. для пользователя доступно только 50% суммарного пространства дисков, входящих в массив;

3) RAID 5 – так называемый чередующийся массив с переходящим контролем четности; в массив объединяют три или более диска, данные записывают блоками, на все диски кроме одного: на него пишутся контрольные суммы для блоков. Все диски по очереди используются для записи данных и контрольных сумм. При отказе одного из дисков все данные можно восстановить по содержимому оставшихся. Обеспечивается отказоустойчивость и избыточность ниже, чем в случае RAID 1.

Важно отметить, что обычно в серверных системах RAID-массивы организуются с помощью аппаратных RAID-контроллеров. Программные реализации дополнительно загружают центральный процессор и поэтому менее предпочтительны.

Для учебных целей на месте разрушенного составного тома создадим зеркальный. Для этого выключим виртуальную машину, снова создадим третий виртуальный диск и загрузимся. На двух дисках создадим зеркальный том и запишем на него какие-нибудь файлы. Выключим виртуальную машину и отключим один из дисков.

Снова загрузимся. Как и в предыдущем случае, в проводнике диск виден не будет. Но с помощью оснастки Disk Management (в меню почти таком же, как на рис.4.5) можно для сбойного тома выбрать опцию Remove Mirror и из неработающего зеркального тома получить работоспособный обычный. Только будьте внимательны при выборе диска, исключаемого из массива. Исключить надо отказавший диск (а не оставшийся работоспособным) иначе данные будут потеряны! А если нужно восстановить «зеркало», то вместо выбывшего из строя, надо установить еще один диск и использовать его для восстановления.

Шаг 3. Настройка квот и работы с сетевыми папками

Для продолжения лабораторной работы нам понадобится только один логический диск (далее будем называть его E:), не считая того, где

расположена операционная система. Остальные можно удалить или оставить для каких-то еще экспериментов.

Предположим, нам нужно разместить на сервере файлы пользователей нашего домена. И для этого будем использовать диск E:. Зачастую при администрировании файлового сервера возникает необходимость отслеживать, сколько дискового пространства используется каждым из пользователей и, при необходимости, ограничивать это значение. Сделать это позволяет механизм дисковых квот.

В проводнике откройте окно свойств диска E: (Start->Computer щелчок правой клавишей на диске E:, пункт Properties в контекстном меню). Перейдите на вкладку Quota и отметьте флажок Enable quota management, установите какую-нибудь небольшую квоту (например, 1 Мб) и пороговое значение для предупреждения администратора (оно должно быть не больше размера квоты, а чаще берется чуть меньше) (рис.4.6). Просмотреть текущее использование дисковых квот можно, нажав на кнопку Quota Entries. Но пока это нам не так интересно.

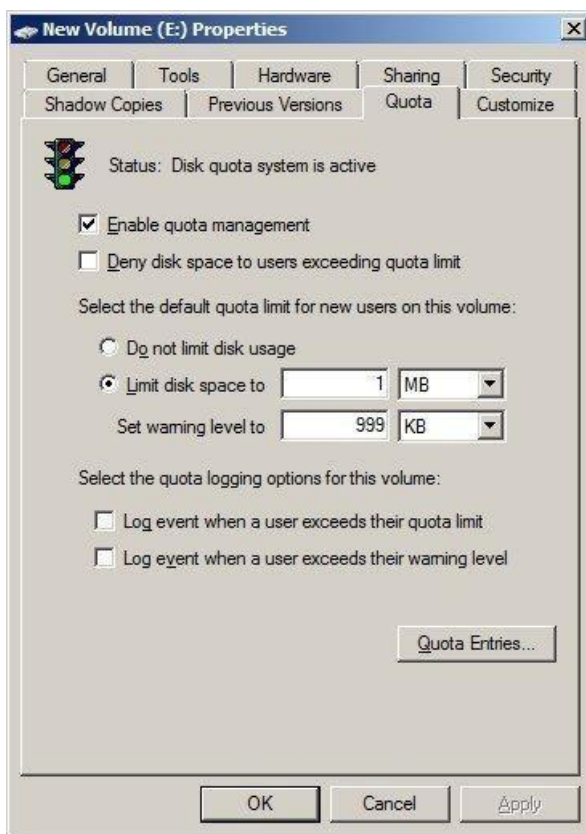


Рисунок 4.6 – Настройка квот для диска

Для дальнейшего выполнения лабораторной работы проверьте, чтобы на сервере была установлена роль File Services (это делалось в ходе выполнения лабораторной работы №2). Если роли нет, то добавьте ее.

Теперь создадим на диске Е: папку и предоставим ее в общий доступ. Для этого откроем свойства папки и перейдем на вкладку Sharing (рис.4.7).

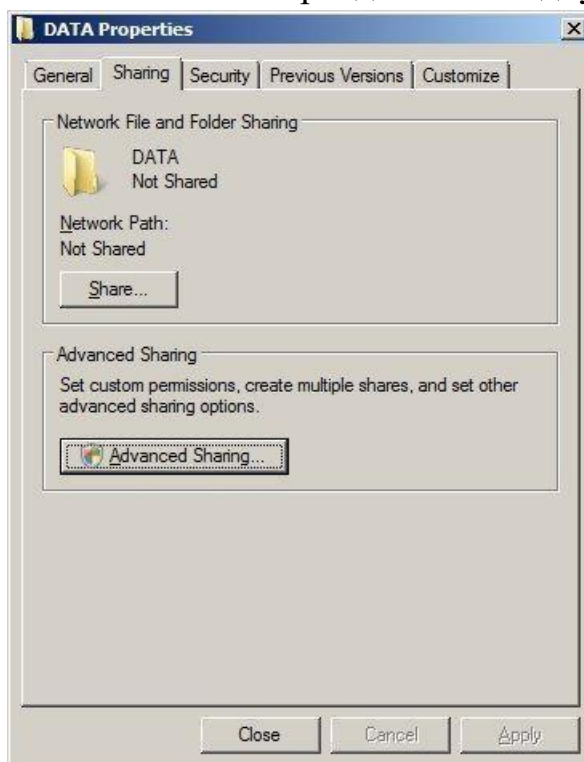


Рисунок 4.7 – Настройка общего доступа для папки

Для более точной настройки общего доступа вместо кнопки Sharing нажмите Advanced Sharing. Отметьте переключатель Share this folder (рис. 4.8) и введите имя общей папки (по умолчанию, подставляется такое же, как имя папки на диске). Если понадобится создать папку, которая не будет отображаться в «Сетевом окружении», то в конце ее имени ставится знак \$. Но в нашем случае этого не требуется.



Рисунок 4.8 – Настройка общего доступа к папке (продолжение).

Теперь нажмите кнопку Permissions, чтобы отредактировать разрешения для пользователей. Вы увидите, что группа Everyone имеет разрешение на чтение (Read). Добавим в список (кнопка Add) доменного пользователя Labos и дадим ему полный доступ к файлам папки (разрешение Full Control). В этой связи следует отметить, что на файловой системе NTFS также устанавливаются разрешения на работу с файлами и папками (их можно просмотреть в свойствах файла или папки на вкладке Security). При доступе к папке локально, работают только разрешения NTFS.

При доступе к папке через сеть срабатывают и разрешения NTFS, и разрешения на общий доступ (итоговое разрешение – наименьшее из двух).

Запускаем рабочую станцию с операционной системой Windows Vista (как и при выполнении лабораторной работы № 3 в нашем учебном классе делимся парами – первый студент на одном компьютере Windows Server, второй - на другом, входящая в этот домен рабочая станция). На рабочую станцию заходим под учетной записью Labos и запускаем Windows Explorer. Открываем узел Network (рис.4.9), находим наш сервер и открываем сетевую папку. Записываем в нее несколько файлов, чтобы их суммарный размер был больше одного Мегабайта.

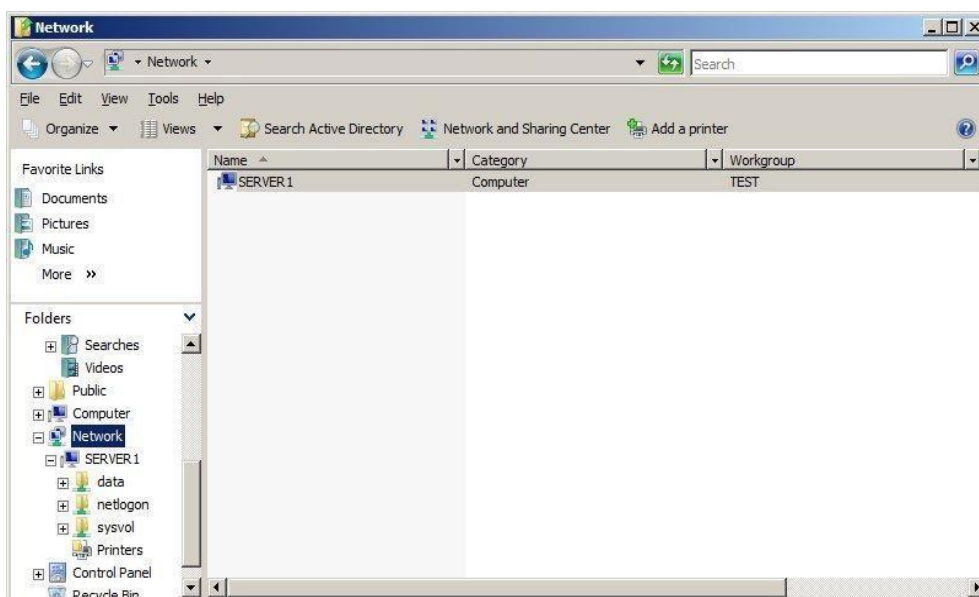


Рисунок 4.9 – Поиск папки в сети

На сервере откроем информацию об использовании квот (кнопка Quota Entries, рис.4.6). Видим, что пользователь Labos превысил лимит, но пока ему писать файлы на диск не запрещается. Срабатывает так называемая мягкая (soft) квота, которая информирует администратора. Если в настройках квот отметить переключатель Deny disk space to users exceeding quota limit, то будет установлена жесткая (hard) квота и пользователь не сможет записывать файлы на этот диск. Установите жесткую квоту, проверьте ее работу.

Лабораторная работа №5. Администрирование баз данных. Управление системными и пользовательскими БД.

1 Цель работы

Изучить основные функции и механизмы, заложенные в СУБД на примере MS SQL Server, позволяющие поддерживать БД в работоспособном состоянии.

2 Краткая теория и порядок выполнения работы

Как и предыдущие, лабораторная работа выполняется пошагово.

Шаг 1. Контрольные точки и их создание

Возможность создания *контрольных точек* была включена в SQL Server с целью фиксации изменений в базе данных или параметрах настройки в заранее заданный удобный момент времени. Если производилась настройка сервера или в его функционирование вносились изменения, потребовавшие перезагрузки управляющей программы, это значит, наступил момент для ручного запуска процедуры создания контрольной точки.

После запуска процедуры создания контрольной точки (независимо от способа ее вызова – вручную или посредством автоматического процесса) все модифицированные страницы памяти сбрасываются на диск. *Модифицированная страница* – это такая страница памяти, которая содержит изменения в данных, еще не зафиксированные в образе базы данных на жестком диске. По умолчанию процедура создания контрольной точки автоматически запускается приблизительно каждые 60 секунд. Реальный промежуток времени между двумя контрольными точками определяется текущей загрузкой сервера, заданными параметрами защиты от сбоев и текущими установками общей настройки производительности SQL Server. Однако этот реальный промежуток времени будет достаточно близок к 60 секундам.

Вероятно, вы уже заметили, что если SQL Server неожиданно прекратил работу в результате случайного сбоя, то на его повторный запуск может потребоваться достаточно много времени. Это происходит по той причине, что SQL Server осуществляет откат и повторное выполнение всех транзакций, которые происходили после создания последней контрольной точки. При этом база данных предварительно приводится в последнее, известное системе, корректное состояние. Оно соответствует именно тому моменту, когда был запущен и успешно завершен процесс создания последней контрольной точки.

Работа SQL Server может быть в любой момент остановлена с помощью выдачи команды shutdown. Если перед выдачей этой команды ввести команду

checkpoint, то все внесенные в результате последних транзакций изменения будут корректно зафиксированы в базе данных до остановки ее работы.

Если по какой-либо причине предстоит остановить работу SQL Server (особенно если требуется указать параметр WITH NOWAIT), то можно избежать продолжительной процедуры последующего запуска системы, предварительно вручную выдав команду CHECKPOINT. Выполнение этой команды ничем не отличается от процедуры создания контрольной точки, вызванной автоматически. Вся информация будет сохранена на диске и запуск системы будет сведен к простой загрузке программ ядра базы данных и предоставления доступа к данным всем клиентским приложениям. Это замечание особенно полезно в тех случаях, когда необходима экстренная остановка сервера – например, при отказе системы электропитания и наличии источника бесперебойного питания, способного обеспечить функционирование сервера на время, достаточное для выполнения цикла создания контрольной точки.

Команда checkpoint выполняется на уровне базы данных и всегда применяется по отношению к текущей базе данных. Если в системе имеется несколько баз данных, эту команду следует выдать для каждой из баз данных в отдельности. Чтобы иметь право выдать команду checkpoint для некоторой базы данных, необходимо иметь статус ее владельца.

Кроме этого, следует упомянуть параметр TRUNCATE LOG ON CHECKPOINT, который также может оказаться довольно полезным. Установка этого параметра вызывает автоматическую очистку журнала транзакций после завершения создания очередной контрольной точки. Таким образом, журнал транзакций будет вестись только в промежутках между созданием контрольных точек. Для установки этого параметра в окне SQL Server Enterprise Manager щелкните правой кнопкой мыши на пиктограмме базы данных и выберите в контекстном меню команду Properties. В раскрывшемся диалоговом окне свойств базы данных перейдите во вкладку Options.

При установленном параметре Truncate log on checkpoint невозможно сделать резервную копию журнала транзакций в процессе выполнения стандартной процедуры резервного копирования базы данных. Обычно это не является проблемой, поскольку стандартное копирование базы данных попрежнему будет выполняться. Однако данное замечание следует учитывать при планировании процедур копирования. Процесс резервного копирования и восстановления системы подробно обсуждается в последующем.

Если для некоторой базы данных установлен параметр Truncate log on checkpoint, то ее журнал транзакций будет каждый раз очищаться в момент

успешной фиксации последней транзакции (при отсутствии в системе репликации данных). Если в системе используется репликация данных, информация в журнале очищается в момент успешного завершения репликации и фиксации в базе данных последней транзакции. Поскольку процедуры репликации данных используют данные журнала транзакций, информация о проведенной транзакции не удаляется из журнала до тех пор, пока сведения о ней не будут доставлены всем подписчикам данного источника.

Шаг 2. Проверка целостности базы данных

```
USE master;  
GO  
EXEC sp_dboption 'Demo_DataBase', 'read only', 'TRUE';
```

Рассмотрим режимы выполнения команды DBCC.

Команда DBCC поддерживает несколько режимов выполнения. В последующих разделах будут рассмотрены те из них, которые используются чаще всего. Кроме того, мы проанализируем, чем они могут помочь в управлении системой SQL Server.

DBCC CHECKALLOC

```
DBCC CHECKALLOC [(имя_базы_данных)[, NOINDEX])] [WITH  
NO_INFOMSGS]
```

Если при вызове команды опустить имя базы данных, SQL Server осуществит проверку текущей базы данных. При выполнении команды с указанием режима CHECKALLOC в отчет помещается детальная информация о системе и существующих в ней объектах базы данных. Эту информацию можно использовать для поиска возможных проблем в системе. Обнаруженные проблемы следует изучать и устранять по отдельности. Помещаемая в отчет информация – это сведения о существующей структуре таблиц, распределении страниц памяти и т.п. Каждое сообщение об обнаруженной ошибке обязательно сопровождается конкретными инструкциями по ее устранению.

Пример.

```
USE Demo_DataBase  
GO  
DBCC CHECKALLOC
```

Результат (фрагмент):

DBCC results for 'Demo_DataBase'.

Table sys.sysrowsetcolumns Object ID 4.

Index ID 1, partition ID 262144, alloc unit ID 262144 (type In-row data). FirstIAM (1:139). Root (1:66). Dpages 5.

Index ID 1, partition ID 262144, alloc unit ID 262144 (type In-row data). 7 pages used in 0 dedicated extents.

Total number of extents is 0.

...

CHECKALLOC found 0 allocation errors and 0 consistency errors in database 'Demo_DataBase'.

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

DBCC CHECKDB

При вызове команды DBCC в режиме CHECKDB, на наличие ошибок проверяется каждая таблица и связанные с ней страницы данных, индексы и указатели. Индексы и страницы данных контролируются на предмет корректности установленных связей. Кроме того, индексы дополнительно анализируются на соответствие заданному порядку сортировки. Для каждой страницы проверяется обоснованность данных, целостность указателей и корректность ее формата. Дополнительно в режиме CHECKDB контролируется правильность связей между текстом, графикой и страницами типа NTEXT, а также корректность их размера. Если выполняется вызов команды DBCC в режиме CHECKDB, то повторно запускать ее в режимах CHECKALLOC или CHECKTABLE не потребуется. Команда DBCC в режиме CHECKDB имеет следующий синтаксис:

DBCC CHECKDB [(*имя_базы_данных*[/, NOINDEX])] [WITH NO_INFOMSGS]

Если опустить параметр имени базы данных, будет выполнена проверка текущей базы данных.

Результат (фрагмент) выполнения команды DBCC CHECKDB без параметров для текущей БД Demo_DataBase:

DBCC results for 'Demo_DataBase'.

Service Broker Msg 9675, State 1: Message Types analyzed: 14.

Service Broker Msg 9676, State 1: Service Contracts analyzed: 6.

Service Broker Msg 9667, State 1: Services analyzed: 3.
Service Broker Msg 9668, State 1: Service Queues analyzed: 3.
Service Broker Msg 9669, State 1: Conversation Endpoints analyzed: 0.
Service Broker Msg 9674, State 1: Conversation Groups analyzed: 0.
Service Broker Msg 9670, State 1: Remote Service Bindings analyzed: 0.
DBCC results for 'sys.sysrowsetcolumns'.
There are 557 rows in 5 pages for object "sys.sysrowsetcolumns".
DBCC results for 'sys.sysrowsets'.
There are 83 rows in 1 pages for object "sys.sysrowsets".
DBCC results for 'sys.sysallocunits'.
There are 96 rows in 1 pages for object "sys.sysallocunits".
DBCC results for 'sys.sysfiles1'.
There are 2 rows in 1 pages for object "sys.sysfiles1".
DBCC results for 'sys.sysshobtcolumns'.
There are 557 rows in 5 pages for object "sys.sysshobtcolumns".
DBCC results for 'sys.sysshobts'.
There are 83 rows in 1 pages for object "sys.sysshobts".
...

CHECKDB found 0 allocation errors and 0 consistency errors in database 'Demo_DataBase'.

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

DBCC CHECKTABLE

Если проверка всей базы данных в целом занимает слишком много времени, можно воспользоваться режимом CHECKTABLE. В этом случае в команде DBCC задается режим CHECKTABLE и указывается перечень имен таблиц базы данных, которые следует проверить. Будет выполнен анализ состояния только указанных таблиц, что позволит сократить общее время проверки.

DBCC CHECKFILEGROUP

При выполнении команды DBCC в режиме CHECKFILEGROUP проверяется каждая таблица и связанные с ней страницы данных, индексы и указатели. Индексы и страницы данных контролируются на корректность установленных связей. Кроме того, индексы дополнительно анализируются на соответствие заданному порядку сортировки. Для каждой страницы проверяется обоснованность данных, целостность указателей и корректность ее формата. Дополнительно в режиме CHECKFILEGROUP контролируется правильность связей между текстом, графикой и страницами типа NTEXT, а также корректность их размера.

Команда DBCC в режиме CHECKFILEGROUP имеет следующий синтаксис:

```
DBCC CHECKFILEGROUP [([{'filegroup_name' | filegroup_id | 0 } ] [ ,
NOINDEX ])]
[ WITH { [ ALL_ERRORMSG ] [ NO_INFOMSGS ] } [ , [ TABLOCK ] ]
[ , [ ESTIMATEONLY ] ] ]
```

Операторы, выводящие информацию о БД

DBCC CONCURRENCYVIOLATION	DBCC SHOW_STATISTICS
DBCC INPUTBUFFER	DBCC SHOWCONTIG
DBCC OPENTDBCC INPUTBUFFERRAN	DBCC SQLPERF
DBCC OUTPUTBUFFER	DBCC TRACESTATUS
DBCC PROCCACHE	DBCC USEROPTIONS

Операторы проверки целостности и корректности данных объектов БД

DBCC CHECKALLOC	DBCC CHECKFILEGROUP
DBCC CHECKCATALOG	DBCC CHECKIDENT
DBCC CHECKCONSTRAINTS	DBCC CHECKTABLE
DBCC CHECKDB	

Операторы очистки и восстановления объектов БД

DBCC CLEANBTABLE	DBCC INDEXDEFRAG
DBCC DBREINDEX	DBCC SHRINKDATABASE
DBCC DROPCLEANBUFFERS	DBCC SHRINKFILE
DBCC FREEPROCCACHE	DBCC UPDATEUSAGE

Miscellaneous Statements

DBCC dllname (FREE)	DBCC TRACEOFF
DBCC HELP	DBCC TRACEON

Шаг 3. Изучение команд UPDATE STATISTICS и RECOMPILE

Высокая производительность системы SQL Server в значительной степени обеспечивается за счет интеллектуальных процессов обработки сохраняемых в таблицах данных. Необходимый анализ информации осуществляется различными методами, однако самым рациональным из них является изучение сохраняемых в системе реальных данных с целью определения оптимальных путей доступа к ним.

Чтобы лучше уяснить, о чем идет речь, давайте в качестве примера рассмотрим маршрут следования к вашему собственному дому. Весьма вероятно, что вы уже нашли самый оптимальный и быстрый путь между

местом работы и местом жительства. И раз этот маршрут уже найден, то чаще всего вы будете пользоваться именно им, даже если кто-то предложит вам воспользоваться иным путем. Ведь вы уже потратили достаточно времени на анализ различных возможных маршрутов и выбрали из них самый оптимальный.

Для того чтобы в сложившийся маршрут были внесены изменения, необходимы достаточно серьезные причины. Это может быть новая дорога, реконструкция уже существующих путей или что-либо в этом роде, способное оказать влияние на время, за которое вы добираетесь домой.

В системе SQL Server подобная аналогия вполне справедлива. При создании хранимой процедуры SQL Server анализирует логику ее оператора SELECT совместно с любыми прочими условиями, зависящими от обрабатываемых данных. На основе этого анализа выбирается оптимальный маршрут обращения к данным при выполнении хранимой процедуры. После того как маршрут определен, информация о нем запоминается, поэтому при каждом следующем выполнении хранимой процедуры используемый в ней алгоритм доступа к данным будет оптимальным. "Маршруты" к данным в

SQL Server прокладываются на основе существующих в системе индексов таблиц. Именно эти элементы анализирует система, определяя наиболее эффективный способ доступа к требуемой информации.

Если в процессе работы в таблицу будет добавлено достаточно большое количество строк данных (превышающее 20% ее исходного размера), следует выполнить обновление статистических данных об этой таблице.

Используемая для этой цели команда имеет следующий синтаксис:

```
UPDATE STATISTICS table | view
[
  {
    { index | statistics_name }
    | ( { index | statistics_name } [ ,...n ] )
  }
]
[ WITH
  [
    [ FULLSCAN ]
    | SAMPLE number { PERCENT | ROWS } ]
    | RESAMPLE
    | <update_stats_stream_option> [ ,...n ]
  ]
[ [ , ] [ ALL | COLUMNS | INDEX ]
```

```

    [ [ , ] NORECOMPUTE ]
];
<update_stats_stream_option> ::=
    [ STATS_STREAM = stats_stream ]
    [ ROWCOUNT = numeric_constant ]
    [ PAGECOUNT = numeric_constant ]

```

Ниже приведен пример команды обновления статистики для таблицы Authors:

```
update statistics Authors
```

В команде можно потребовать выполнить пересчет статистики как для одного определенного индекса или столбца данных, так и для всей таблицы в целом. Если указывается отдельный индекс или столбец данных, то это делается так, как показано выше, с одновременным указанием имени таблицы, в которой этот индекс или столбец располагается.

Периодически SQL Server по собственной инициативе использует промежутки времени, когда процессор простаивает, для обработки таблиц с целью поддержания актуальности статистических данных, что способствует повышению эффективности обработки информации в системе.

Режим ручного обновления статистики также сохранен, причем соответствующая команда дополнена новыми параметрами. Так, при указании в команде обновления статистики параметра FULLSCAN SQL Server выполнит полное сканирование указанного индекса или таблицы. Наличие в команде параметра SAMPLE потребует от SQL Server использовать технологию выборок данных на основе указанного количества или процента строк. В этом случае SQL Server должен будет убедиться, что в качестве образца выбрано статистически значимое число значений. Если указанное число или процент строк недостаточно велики для выполнения этого требования, система автоматически увеличит заданное значение. Последний из параметров INDEX, COLUMN или ALL указывает, для каких объектов должна собираться статистика – индексов, столбцов или и тех, и других. При опускании этого параметра статистика собирается только для индексов.

Для прекращения автоматического сбора статистики (хотя я и не знаю, зачем это может кому-либо потребоваться) следует ввести команду UPDATE STATISTICS с параметром NORECOMPUTE. Указание этого параметра блокирует выполнение процедур автоматического сбора статистики в системе SQL Server.

Статистика по определенной таблице, индексу или столбцу может быть удалена, для чего предназначена команда DROP STATISTICS, имеющая приведенный ниже синтаксис:

DROP STATISTICS *таблица.столбец* [...*таблица.столбец*]

Последним шагом на пути обновления индексов с целью использования новых статистических данных в хранимых процедурах является уведомление SQL Server о необходимости пересчитать маршруты, используемые для извлечения информации из базы данных. Это достигается посредством перекомпиляции хранимых процедур, работающих с данными, статистические сведения о которых были обновлены только что выполненными командами UPDATE STATISTICS.

Обычно хранимые процедуры компилируются в момент первого их вызова после перезапуска системы SQL Server. Существуют и другие ситуации, в которых выполняется автоматическая перекомпиляция хранимых процедур, однако простая выдача команды UPDATE STATISTICS к ним не относится. В качестве примера можно указать ситуацию, когда ликвидируется индекс, ранее использовавшийся в некоторой хранимой процедуре. Иногда можно заметить, что при первом вызове некоторой хранимой процедуры время реакции системы оказывается существенно большим, чем обычно. При этом все последующие вызовы этой процедуры выполняются уже с нормальной скоростью. Это происходит по той причине, что оптимизатор выполнил перекомпиляцию хранимой процедуры с целью учета новых статистических данных, полученных для используемых в запросе таблиц или индексов.

Чтобы перекомпилировать хранимые процедуры, необходимо установить для таблицы соответствующий флажок, предназначенный для извещения системы о том, что любые работающие с этой таблицей и находящиеся в процедурном кэше копии хранимых процедур следует считать недействительными. В результате SQL Server перезагрузит и перекомпилирует все соответствующие процедуры при первом же их вызове. Для установки данного флажка используется следующая команда:

sp_recompile <*имя_таблицы*>

Здесь параметр *имя_таблицы* определяет имя той таблицы, для которой следует перекомпилировать все обращающиеся к ней хранимые процедуры. В случае успешного выполнения команда выводит простое сообщение, уведомляющее, что все связанные с указанной таблицей хранимые процедуры будут перезагружены и перекомпилированы.

Пример.

EXEC sp_recompile 'authors'

GO

Результат:

Object 'authors' was successfully marked for recompilation.

Если не перекомпилировать те хранимые процедуры, которые связаны с результатами выполнения команды `update statistics`, то ожидаемое улучшение производительности достигнуто не будет вплоть до очередной остановки и перезагрузки сервера. Только в этом случае все хранимые процедуры будут автоматически заново перекомпилированы.

Выполнение команд `UPDATE STATISTICS` и `RECOMPILE` не может нанести системе никакого вреда, однако делать это рекомендуется в то время, когда активность пользователей в системе относительно невысока. Особенно важно придерживаться этого правила при работе с крупными базами данных. Время, которое будет затрачено на получение новых статистических данных, может существенно отразиться на производительности системы по обработке запросов пользователей. Лучше всего выполнять процедуры обновления статистики как часть регулярных манипуляций, связанных с плановым обслуживанием системы. Например, имеет смысл выполнять эту процедуру не реже, чем раз в месяц для каждой из интенсивно используемых в системе таблиц с невысоким уровнем изменения данных. В случае, когда система только запускается в работу и производится интенсивное заполнение данными ее таблиц, может потребоваться выполнять процедуры обновления статистики гораздо чаще – возможно, даже ежедневно, – если поток новых данных достаточно велик.

Шаг 4. Импорт/экспорт данных в SQL Server.

Последовательность действий для экспорта данных в другую БД SQL Server (создание копии БД) иллюстрируется рис.5.1 – 5.7.

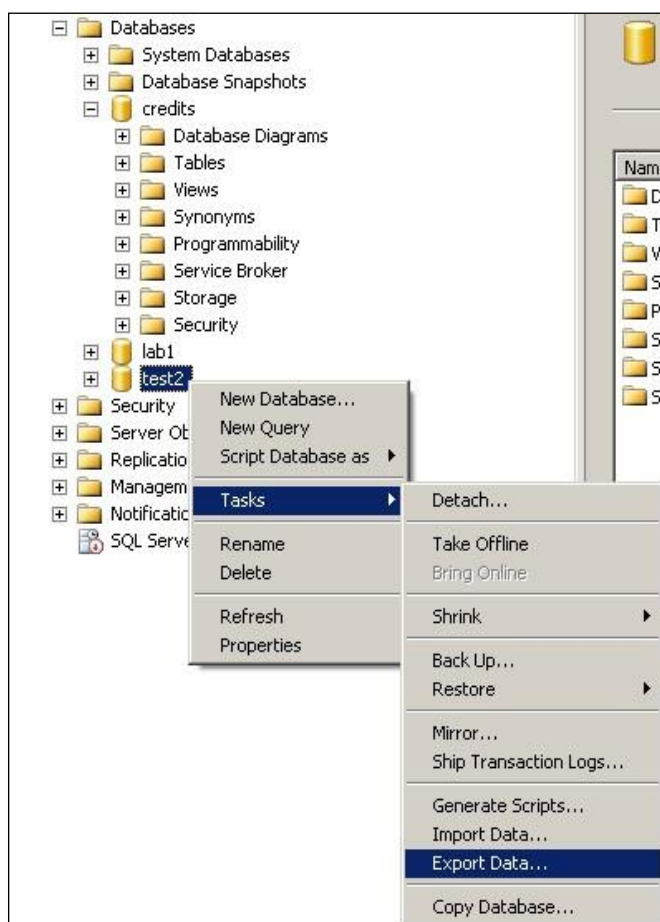


Рисунок 5.1 – Первое действие для экспорта данных в другую БД

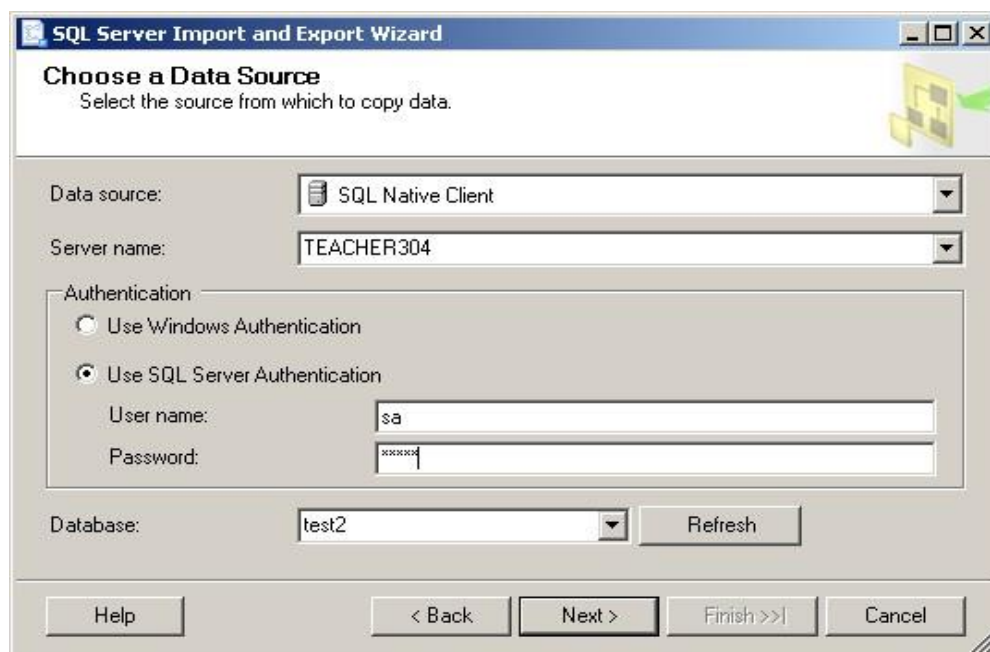


Рисунок 5.2 – Второе действие для экспорта данных в другую БД

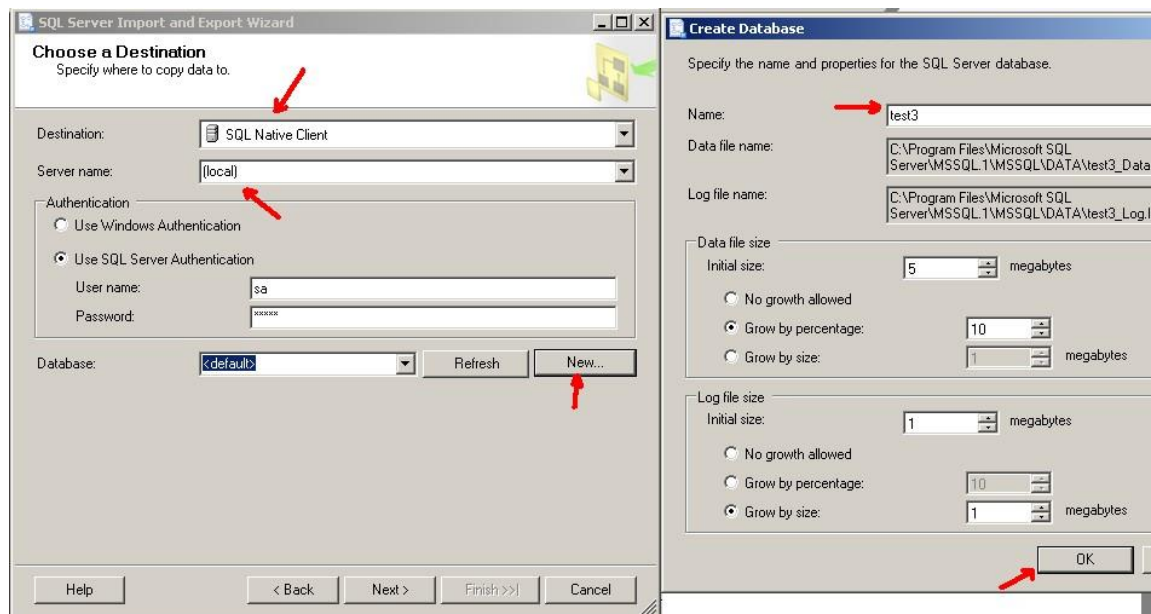


Рисунок 5.3 – Третье действие для экспорта данных в другую БД



Рисунок 5.4 – Четвертое действие для экспорта данных в другую БД

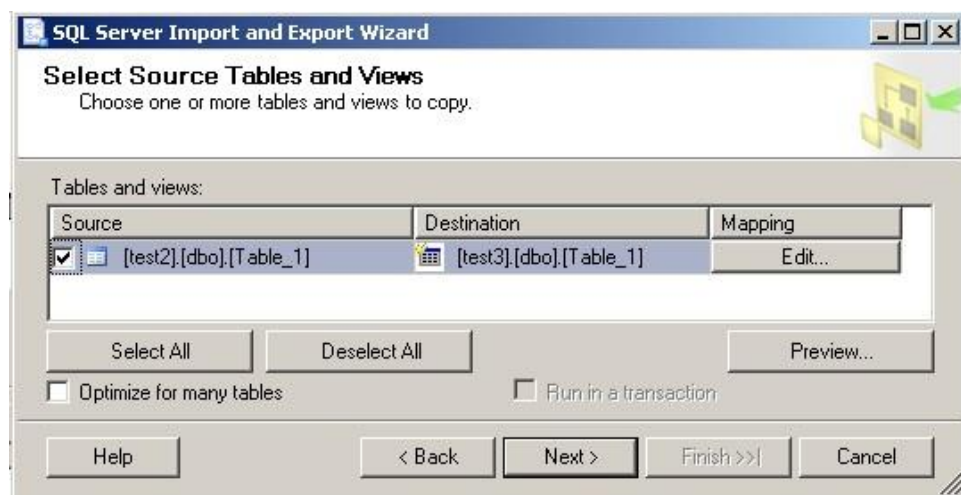


Рисунок 5.5 – Пятое действие для экспорта данных в другую БД



Рисунок 5.6 – Шестое действие для экспорта данных в другую БД

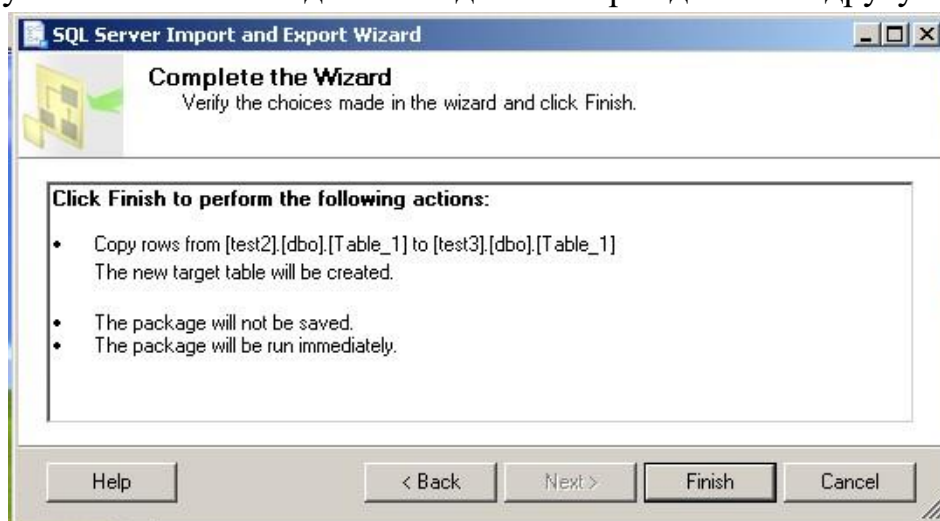


Рисунок 5.7 – Седьмое действие для экспорта данных в другую БД

Экспорт данных в другую СУБД выполняется аналогично только на шаге, показанном на рисунке 5.3, только в разделе Destination указать в качестве приемника нужную СУБД.

Лабораторная работа №6. Настройка служб DNS и DHCP.

1 Цель работы

Освоить основы настройки и администрирования служб DNS и DHCP.

2 Краткая теория и порядок выполнения работы

Используемое программное обеспечение

Для выполнения данной лабораторной работы понадобится компьютер с операционной системой Microsoft Windows или Linux Mint и установленным программным обеспечением Oracle VM VirtualBox, а также две подготовленные виртуальные машины с Windows Server 2008 и Windows Vista.

Шаг 1. Изучение основ организации DNS сервера

Во второй лабораторной работе вы уже установили DNS сервер.

DNS — это система именования (и поддерживающие ее службы) компьютеров и сетевых служб, которая сопоставляет имена компьютеров и сетевые адреса, организуя их в доменную иерархическую структуру. Система именования DNS используется в сетях TCP/IP, например, в интернете и в большинстве корпоративных сетей, для поиска компьютеров и служб по понятным именам. Когда пользователь вводит DNS-имя компьютера в приложении, DNS находит имя компьютера и другую связанную с ним информацию, например, его IP-адрес или службы, которые он предоставляет в сети. Этот процесс называется разрешением имен.

DNS находит имена компьютеров и служб и сопоставляет их с числовым адресом, который требуется операционным системам и приложениям для идентификации компьютера в сети.

Служба доменных имен — это главный метод разрешения имен в Windows Server. При использовании DNS необходимо развертывание роли сервера доменных служб Active Directory.

Имя DNS состоит из двух или более частей, разделенных точками (.), рис.6.1. Последняя часть имени (справа) называется доменом верхнего уровня (TLD – top-level domain).

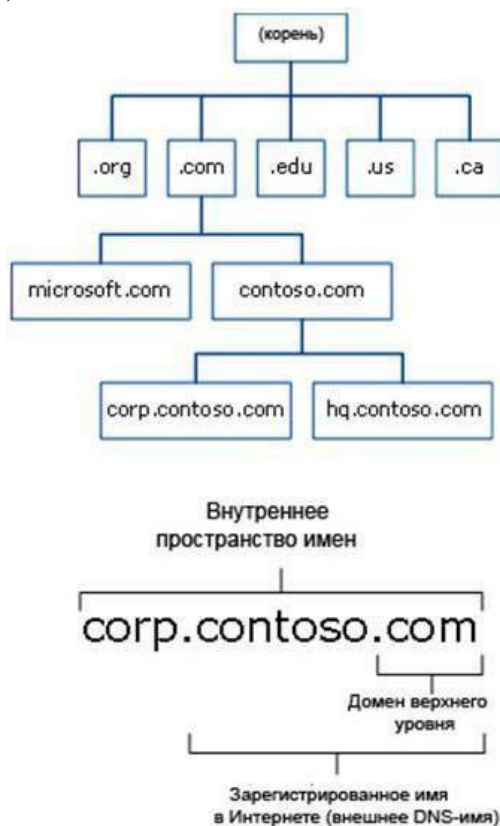


Рисунок 6.1 – Организация пространства имен DNS

Другие части имени — это дочерние домены домена верхнего уровня или другого дочернего домена. Имена TLD бывают функциональными или географическими. Дочерние домены, как правило, указывают на организацию, владеющую доменным именем.

Шаг 2. DNS-запись. Обратный DNS-запрос

Как частный случай, DNS может хранить и обрабатывать и обратные запросы, определения имени хоста по его IP адресу — IP адрес по таблице соответствия преобразуется в доменное имя, и посылается запрос на информацию типа PTR.

Для этого используются уже имеющиеся средства DNS. Дело в том, что с записью DNS могут быть сопоставлены различные данные, в том числе и какое-либо символьное имя. Существует специальный домен `in-addr.arpa`, записи в котором используются для преобразования IP-адресов в символьные имена. Например, для получения DNS-имени для адреса 11.22.33.44 можно запросить у DNS-сервера запись `44.33.22.11.in-addr.arpa`, и тот вернёт соответствующее символьное имя. Обратный порядок записи частей IP-адреса объясняется тем, что в IP-адресах старшие октеты расположены в начале, а в символьных DNS-именах старшие (находящиеся ближе к корню) части расположены в конце.

При запросе осуществляется считывание записи PTR, содержащей искомое доменное имя. Если запись отсутствует, то IP-адрес считается не имеющим обратного DNS. DNS-запись `in-addr.arpa` выглядит так:

`78.56.34.12.in-addr.arpa. IN PTR domain.ltd.`

Это будет означать, что имени хоста `domain.ltd` соответствует IP-адрес `12.34.56.78`.

Понятие записей ресурсов

Объекты в иерархии DNS идентифицируются с помощью *записей ресурсов (Resource Record)*. Они используются для выполнения основных операций поиска пользователей и ресурсов внутри указанного домена и уникальны для содержащего их домена.

Рассмотрим некоторые типы записей.

1. *Начальная запись зоны SOA (Start of Authority)*. Указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, тайминги кеширования зонной информации и взаимодействия DNS-серверов.

2. *Записи хостов (записи A, Address Record)*. Наиболее часто встречающийся тип записей ресурсов. Содержит имя хоста и соответствующий ему IP-адрес.

3. *Записи сервера имен (Name Server – NS)*. Указывают, какие компьютеры в базе данных DNS являются серверами имен – то есть DNS-серверами для конкретной зоны. Для каждой зоны может существовать только одна запись типа SOA, но может быть несколько NS – записей.

4. *Запись AAAA (IPv6 address record)* связывает имя хоста с адресом протокола IPv6.

5. *Запись CNAME (canonical name record)* или *каноническая запись имени* позволяет присваивать хосту мнемонические имена. Мнемонические имена, или псевдонимы, широко применяются для связывания с хостом какой-либо функции, либо просто для сокращения имени.

6. *Запись MX (mail exchange)* определяет почтовый сервер - машину, которая обрабатывает почту для данного домена.

7. *Запись SRV (server selection)* используется для поиска серверов, обеспечивающих работу тех или иных служб в данном домене.

8. *Запись PTR (pointer)* или *запись указателя* связывает IP хоста с его каноническим именем.

Шаг 3. Изучение понятия зоны

Зона - логический узел в дереве имен. Рассмотрим типы зон в DNS.

- *Зона прямого просмотра (forward lookup zone)* создается в DNS по умолчанию во время установки. Она необходима для преобразования доменного имени в IP-адрес и информацию о ресурсах. Большинство записей в прямой зоне типа A.

- В дополнение к зоне прямого просмотра мы в этой лабораторной работе создадим *зону обратного просмотра (reverse lookup zone)*, которая реализует обратный DNS-запрос. Развертывание зоны обратного просмотра обычно улучшает производительность DNS и существенно повышает успешность DNS-запросов. Зона обратного просмотра состоит почти целиком из записей типа PTR (Pointer).

- *Первичная зона (Primary zone)*. В DNS (без интегрированной Active Directory) один сервер служит первичным DNS – сервером зоны, и все изменения, выполняемые в данной зоне, выполняются на этом конкретном сервере. Один DNS – сервер может содержать несколько зон, будучи первичным для одной зоны и вторичным для другой. Однако если зона является первичной, все запрошенные изменения для данной зоны должны выполняться на сервере, содержащем основную копию зоны.

- *Вторичная зона (Secondary zone)* создается для обеспечения резервирования и разгрузки первичной зоны. Однако каждая копия базы данных DNS доступна только для чтения, т.к. все модификации записей

выполняются в первичной зоне. Один сервер DNS может содержать несколько первичных и вторичных зон.

- *Зона-заглушки (Stub zone)* представляет собой зону, которая не содержит никакой информации о членах домена, а служит только для переадресации запросов к списку назначенных серверов имен для различных доменов. Поэтому она содержит только записи NS, SOA и связанные записи (glue records – записи A, которые используются в сочетании с конкретной записью NS для преобразования IP-адреса конкретного сервера имен). Сервер, содержащий зону-заглушку какого-либо пространства имен, не управляет зоной. Она используется для ускорения работы.

Шаг 4. Примеры. Выполнение контрольного задания

Для выполнения практического задания в данной лабораторной работе необходимо будет задействовать рабочую станцию и сервер. На рабочую станцию заходим под пользователем labos (пароль Lab0s123).

На сервере с помощью оснастки DNS Manager (Start-> Administrative Tools-> DNS Manager) можно посмотреть, какие типы записей были созданы при установке DNS, а также убедиться в том, что по умолчанию была создана зона прямого просмотра. Для этого нужно щелкнуть по нашему серверу S08, далее выбрать папку Forward Lookup Zones (зоны прямого просмотра), а там выбрать домен Saiu_test (рис. 6.2).

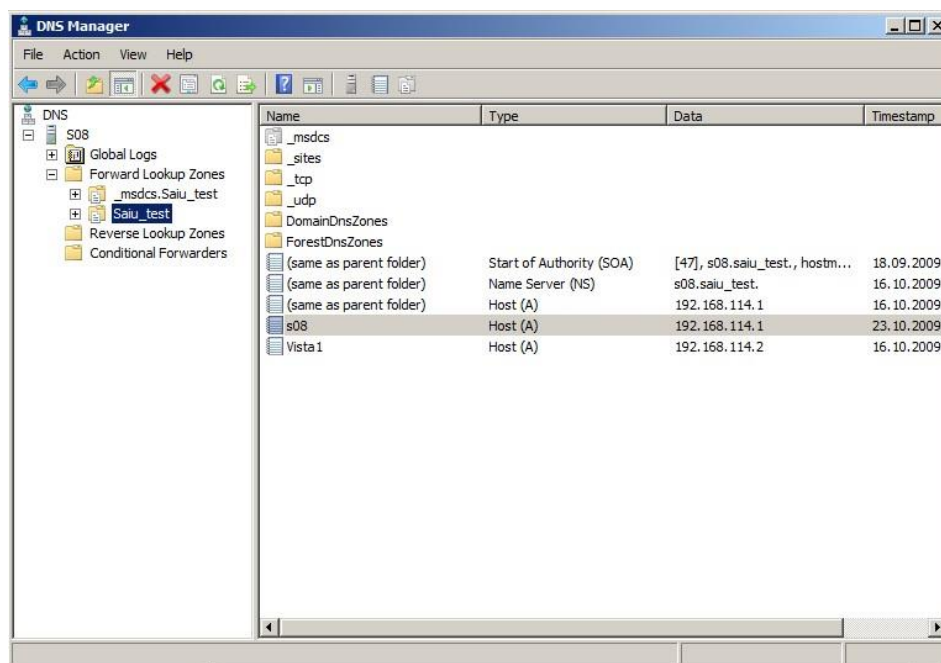


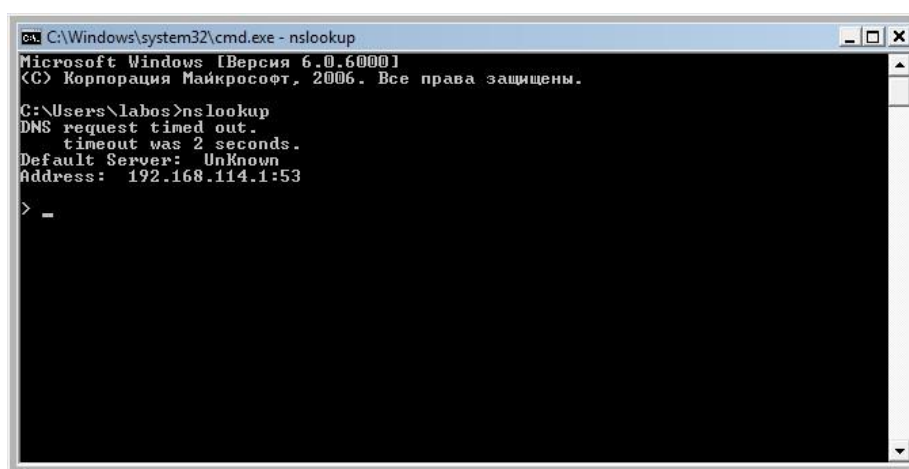
Рисунок 6.2 – Просмотр записей и зон DNS

Примечание: в примерах данной работы используется IP-адрес для сервера и рабочей станции, как и в предыдущих лабораторных работах: 192.168.1уу.1 и 192.168.1уу.2, где уу – номер компьютера. В данном случае,

номер компьютера равен 14 и в последующих примерах везде будет использоваться именно он.

Как видим, были созданы начальная запись зоны (SOA), записи хостов (A) и запись сервера имен (NS).

На рабочей станции воспользуемся командой nslookup (name server lookup - поиск на сервере имён), которая предоставляет пользователю интерфейс командной строки для обращения к системе DNS, а также позволяет задавать различные типы запросов и запрашивать произвольно указываемые сервера (рис. 6.3). В ответ на приглашение «>» можно ввести имя нашего виртуального сервера – s08.saiu_test и получить его ip-адрес – 192.168.114.1. Если ввести ip-адрес, то в имя он не разрешится.



```
cmd. C:\Windows\system32\cmd.exe - nslookup
Microsoft Windows [Версия 6.0.6000.1
(C) Корпорация Майкрософт, 2006. Все права защищены.
C:\Users\labos>nslookup
DNS request timed out.
    timeout was 2 seconds.
Default Server: UnKnown
Address: 192.168.114.1:53
> _
```

Рисунок 6.3 – Пример работы команды nslookup

Данный пример подтверждает, что на сервере есть зона прямого просмотра (IP-адрес определяется правильно - Address: 192.168.114.1). Но на нет зоны обратного просмотра или в ней для искомого адреса, т.е. для него не создана запись (это можно также проверить на сервере, посмотрев в оснастке DNS папку Reverse Lookup Zones).

Создадим псевдоним (запись CNAME), чтобы рабочая станция смогла обратиться к серверу и по альтернативному имени. Например, мы планируем на сервере s08 организовать web-сервер, и хотим, чтобы к нему можно было обратиться www.saiu_test. Для этого на оснастке DNS Manager правой кнопкой мыши щелкаем по домену Saiu_test и из списка выбираем New Alias (CNAME)... (рис. 6.4).

Задаем имя псевдонима www и хост s08.Saiu_test.

Теперь в списке записей прямой зоны появилась запись типа CNAME с именем www. На рабочей станции теперь можно проверить доступность псевдонима www.saiu_test с помощью команды nslookup www.saiu_test.

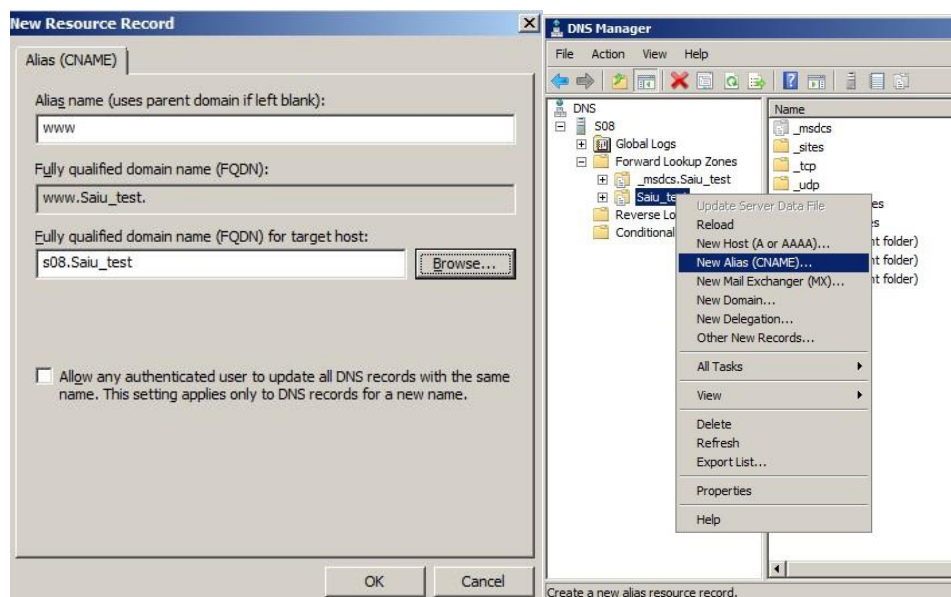


Рисунок 6.4 – Создание псевдонима записи типа CNAME

Теперь настроим обратное разрешение имен. Для этого понадобится создать обратную зону. В DNS Manager на вкладке Action выбираем New Zone... Создаем Reverse Lookup Zone и изменяем только: имя во вкладке Reverse Lookup Zone Name на 114.168.192.in-addr.arpa (рис.6.5) для сети 192.168.114.0 (или другое имя при использовании другого адреса сети). И при вопросе о Dynamic Update выбираем Allow only secure dynamic updates (разрешить только безопасные динамические обновления).

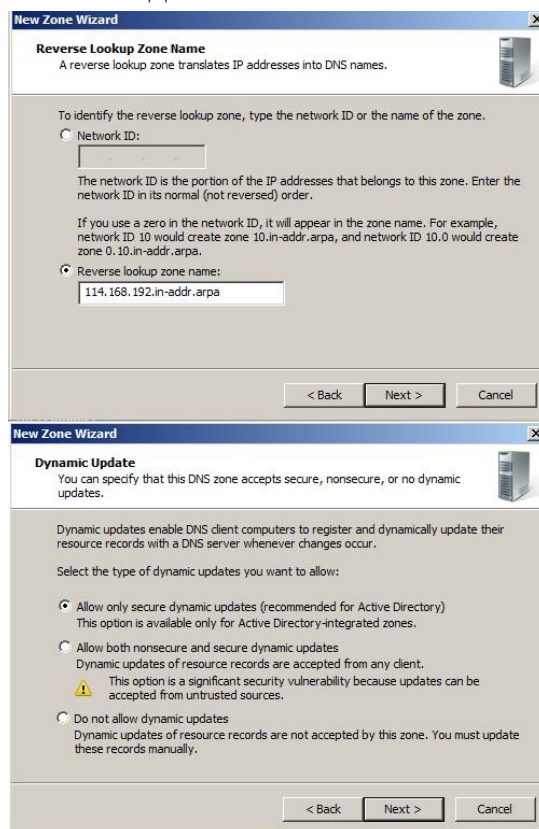


Рисунок 6.5 – Создание обратной зоны

Теперь в папке обратных зон появилась новая обратная зона с заданным именем. Но рабочая станция все равно не может получить имя сервера, поскольку не создана запись указателя (PTR), связывающая IP хоста с его каноническим именем. Для этого на сервере выполняем команду `ipconfig` с ключом `/registerdns` (рис.6.6).



Рисунок 6.6 – Обновление регистрации DNS-клиента

Примечание. В операционных системах Microsoft Windows `ipconfig` — это утилита командной строки для вывода деталей текущего соединения и управления клиентскими сервисами DHCP и DNS. Утилита `ipconfig` позволяет определять, какие значения конфигурации были получены с помощью DHCP, APIPA или другой службы IP-конфигурирования либо заданы администратором вручную.

Таблица 6.1 – Ключи команды `ipconfig`

Ключ	Описание
<code>/all</code>	Отображение полной информации по всем адаптерам.
<code>/release адаптер]</code>	Отправка сообщения DHCPRELEASE серверу DHCP для освобождения текущей конфигурации DHCP и удаления конфигурации IP-адресов для всех адаптеров (если адаптер не задан) или для заданного адаптера. Этот ключ отключает протокол TCP/IP для адаптеров, настроенных для автоматического получения IP-адресов.
<code>/renew [адаптер]</code>	Обновление IP-адреса для определённого адаптера или если адаптер не задан, то для всех. Доступно только при настроенном автоматическом получении IP-адресов.
<code>/flushdns</code>	Очищение DNS кэша.
<code>/registerdns</code>	Команда обновления регистрации DNS-клиента
<code>/displaydns</code>	Отображение содержимого кэша DNS.
<code>/showclassid адаптер</code>	Отображение кода класса DHCP для указанного адаптера. Доступно только при настроенном автоматическом получении IP-адресов.
<code>/setclassid адаптер [код_класс a]</code>	Изменение кода класса DHCP. Доступно только при настроенном автоматическом получении IP-адресов.
<code>/?</code>	Справка.

Выполнив эту команду на сервере и клиентской станции, можно проверить, что в обратной зоне появились две новые записи типа PTR (рис.6.7), которые позволяют по ip-адресу узнать имя. Их также можно было создать и в оснастке DNS.

Name	Type	Data	Timestamp
(same as parent folder)	Start of Authority (SOA)	[3], s08.saiu_test., hostma...	static
(same as parent folder)	Name Server (NS)	s08.saiu_test.	static
192.168.114.1	Pointer (PTR)	s08.saiu_test.	23.10.2009
192.168.114.2	Pointer (PTR)	vista1.saiu_test.	23.10.2009

Рис.6. Записи типа PTR:192.168.114.1 – IP-адрес сервера, 192.168.114.2 – IP-адрес рабочей станции.

В частности, теперь команда nslookup, введенная на рабочей станции, показывает не только IP-адрес сервера DNS (как было на рис.6.3), но и его имя - s08.saiu_test.

Шаг 5. Добавление роли DHCP

DHCP (*Dynamic Host Configuration Protocol* — протокол динамической конфигурации узла) — это сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к *серверу DHCP*, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок.

Для использования протокола TCP/IP в сети администратор должен задать для каждого из компьютеров по меньшей мере три параметра - IP-адрес, маску подсети и адрес используемого по умолчанию шлюза. При этом, каждый компьютер должен иметь уникальный IP-адрес. Кроме того, присвоенный адрес должен находиться в диапазоне подсети, к которой подключено устройство. В большой сети иногда бывает трудно определить, к какой же из подсетей подключен тот или иной компьютер. Однако DHCP-сервер "знает", из какой подсети приходит запрос на получение IP-адреса, и назначит клиенту правильный адрес. Если в сети используются Windows Internet Naming Service (WINS) и Domain Name Service (DNS), то IP-адреса WINS и DNS-серверов также может указывать DHCP-сервер.

Установим и сконфигурируем DHCP-сервер.

На сервере S08 из оснастки Server Manager добавляем роль DHCP. На вкладке DHCP Scopes добавляем адреса, которые могут быть использованы. Пусть, распределяемые адреса будут начинаться с адреса 192.168.114.2 и заканчиваться адресом 192.168.114.100. Назовем эту группу адресов «192.168.114.1» (по адресу сервера). Задаем маску и шлюз соответственно: 255.255.255.0 и 192.168.114.1. Активируем этот набор адресов (рис.6.8).

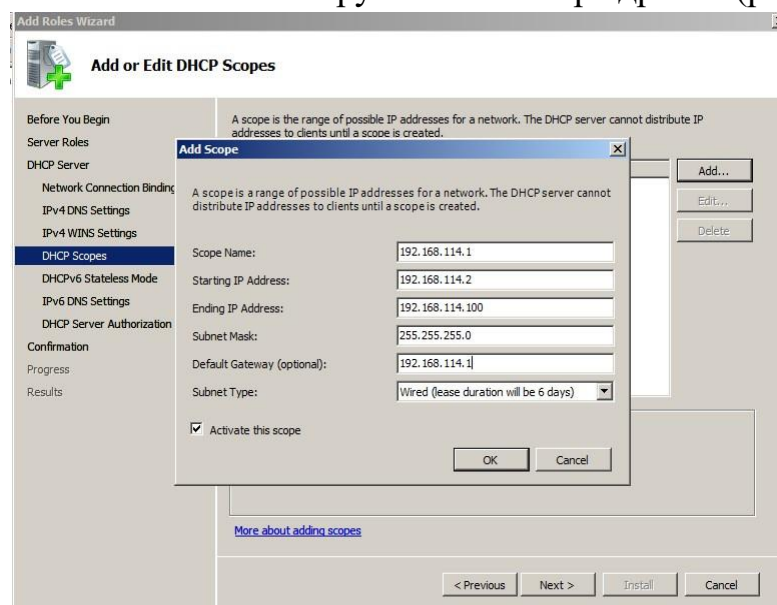


Рисунок 6.8 – Конфигурирование DHCP-сервера в процессе установки

Далее продолжаем установку так, чтобы получились параметры, аналогичные представленным на рис.6.9.

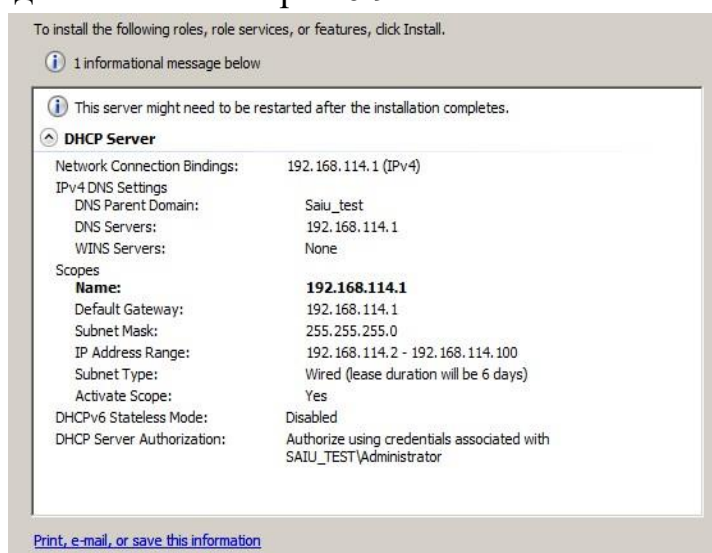


Рисунок 6.9 – Отчет об установке роли DHCP

Когда роль DHCP на сервере установлена, на рабочей станции нужно сделать так, чтобы получать IP-адрес и DNS автоматически, а не прописывать их вручную. Это указывается в настройках протокола TCP/IP v.4 в свойствах сетевого интерфейса. Сделайте подобную настройку и протестируйте ее.

Часть 2. Технологии интеграции программных (информационных) систем

Введение

Интеграция – это процесс объединения и совместной работы информационных систем и программных приложений. Интеграция ИС, ПС предполагает их взаимодействие с целью обмена данными и синхронизации информации. Учитывая современные тенденции к распределенности информации с сохранением ее целостности, задача интеграции рассматриваемых систем, на сегодня, является одной из важнейших.

Выделяют следующие уровни интеграции:

- физический- конвертация данных из различных источников в единый формат их физического представления;
- логический - реализация единой глобальной схемы, которая описывает совместное представление данных из различных источников с учетом их структурных и поведенческих свойств без учета семантики;
- семантический - поддержка единого представления данных с учетом их семантических свойств в едином контексте.

Перечислим основные факторы, влияющие на интеграцию.

- Ускорение процессов. Развитие организации требует все чаще и чаще менять структуры данных, бизнес-процессы, не говоря уже о дизайне и пользовательском интерфейсе, который просто постоянно находится в изменении. Вот, как раз в таких динамичных областях, где “изменчивость” является самой сутью и природой системы, задача интеграции усугубляется и превращается в серьезную проблему.
- Распределенность. Организации становятся все более крупными, а решаемые задачи все более комплексными, появляется логическая, организационная и географическая рассредоточенность.
- Гетерогенность. В крупном программном проекте, почти никогда нет возможности придерживаться платформ и инструментов от одного производителя, поэтому приходится учитывать и поддерживать особенности нескольких платформ.
- Наследственность. Невозможность полностью отказаться от легаси систем, морально устаревших технологий, старого аппаратного обеспечения, которые, кстати, иногда дают вполне хорошие показатели по надежности и производительности, но уж никак не способствуют интеграции.
- Хаотичность. Не всегда есть возможность полностью формализовать, специфицировать и структурировать данные, и часть модели

остается “слабо-связанной”, не поддающейся или слабо поддающейся машинной обработке, анализу, индексации, обсчету.

- **Обусловленность.** Информационные системы ограничены не только техническими рамками, но и привычками людей, особенностями законодательства, множеством других факторов, не зависящих от разработчиков.

- **Интерактивность.** Потребитель информации постоянно повышает свои ожидания о скорости реакции системы, быстродействии и оперативности доставки информации. Большинство процессов выполняются в реальном времени.

- **Высоконагруженность.** На сложность интеграции влияют: количество пользователей в системе, интенсивность потока обработки данных, объемы данных и ресурсоемкость вычислений.

- **Межсистемная интеграция.** Задачи стыковки не ограничены рамками организации, все чаще нужно интегрироваться с партнерами, клиентами, поставщиками, подрядчиками и даже государственными структурами.

Существует два базовых подхода к интеграции.

- 1) Новая система получается централизованной (возможно на базе уже существующей системы), а интегрируемые объекты становятся подсистемами.

- 2) Используется архитектура брокера (посредника, не являющегося центром) и системы остаются независимыми. Главный плюс – универсальность, т.к., всегда можно добавить дополнительный программный модуль.

Под интеграцией приложений можно понимать стратегический подход к объединению информационных систем, обеспечивающий возможность обмена информацией и поддержания распределенных бизнес-процессов.

Интеграция информационных систем дает предприятию такие несомненные конкурентные преимущества, как:

- ведение бизнеса в режиме реального времени с использованием событийно-управляемых сценариев;

- владение достоверной, полной и своевременно полученной информацией.

Главная задача интеграции — обеспечить эффективный, надежный и безопасный обмен данными между различными программными продуктами, изначально не предназначенными для совместной работы. Как правило, требования бизнеса эволюционируют быстрее, чем способы их поддержки информационными технологиями.

Лабораторная работа №7. Определение потоков данных. Бизнес-модель предприятия, программное и информационное обеспечение

1 Цель работы

На основе изучения информационных потоков по составленной бизнес-модели предприятия и состава программных продуктов провести аудит и предложить мероприятия по интеграции ПО.

2 Краткая теория

К началу 2020-х годов на больших и средних российских предприятиях наблюдается «типичная картина»: в корпоративных информационных системах используется большое количество программных продуктов различных годов выпуска и разных производителей. Это произошло во многом в следствии того, что изначально не было единого продукта для информационного сопровождения бизнеса и ПО «наслаивалось» друг на друга. Главная проблема – сведение информации в единый источник для анализа, а также постоянные «перекачки» данных из одной информационной системы в другую. Кроме того – «задвоение» данных и другие неприятные моменты.

Корпоративные информационные потоки – это упорядоченные потоки информации, обеспечивающее ведение предприятием своей финансово-хозяйственной деятельности (бизнес-деятельности).

Выделяют следующие виды потоков по классификации.

1. По отношению к элементам (подразделениям) организационной структуры управления предприятием – горизонтальные и вертикальные.
2. В зависимости от источника их происхождения и среды движения – внутренние и внешние.
3. По отношению к логистической системе движения товаров (услуг) – входные и выходные.

Так, вертикальные информационные потоки имеют место между работниками или группами работников, находящимися на различных уровнях иерархии, например, между начальником и подчиненным.

Горизонтальные информационные потоки чаще всего имеют неформальный характер, они являются самыми эффективными, с коммуникативной точки зрения. В них сохраняется примерно 90% сведений. Потеря информации при передаче таким путем минимальна и это объясняется тем, что работникам предприятия, находящимся на одном уровне служебной иерархии, психологически легче понять друг друга, поскольку они решают однотипные задачи и сталкиваются со сходными проблемами при решении.

Внешние информационные потоки отражают отношения между предприятием и экономическими и политическими субъектами, действующими за его пределами. Они определяют взаимодействие между предприятием, его реальными и потенциальными клиентами, а также конкурентами. Предприятие должно постоянно следить за основными компонентами внешней среды, к которым относятся экономические, технологические, политико-правовые, социально-культурные и физико-экологические факторы.

Внутренние информационные потоки - это потоки информации между различными службами и уровнями предприятия, например, документ на отпуск материалов с заводского склада в цех предприятия, документ об отпуске материальных ресурсов из одного цеха другому, документ о приемке на склад готовой продукции, изделий и деталей. Внутренние информационные потоки четко характеризуют этапы процесса: снабженческий, внутрипроизводственный, сбытовой.

В результате разделения и оценки (качественной, количественной) информационных потоков, обобщенно задачу интеграции программного обеспечения информационной системы (или более глобально, ее составляющих информационных подсистем) можно сформулировать следующим образом:

«Необходимо интегрировать N-информационных подсистем (или программных продуктов) в рамках одной (единой) корпоративной информационной системы, находящихся под влиянием факторов интеграции (приведены во Введении) с минимизацией количества прослоек, конвертеров, брокеров и интерфейсов между ними».

Если решать эту задачу напрямую, то между N-системами будет $N(N-1)/2$ связей, то есть, при двухстороннем взаимодействии $N(N-1)$ интерфейсов. Под интерфейсом можно понимать все что угодно, от веб-сервиса до оффлайн-процесса, запускаемого, например, раз в сутки и делающего целый ряд сложных операций по синхронизации баз (запросы, обработку, экспорт, загрузку по FTP, передачу сигнала другой части системы, чтобы та приняла переданные данные и выполнила свою часть работы, а потом уведомила о результатах и передала необходимые данные обратно).

Решение этой задачи должно базироваться на предварительном проведении анализа информационных потоков, для чего целесообразно построить бизнес-модель процессов, отражающих деятельность рассматриваемого предприятия. Необходимо, выделив основные программные продукты информационной системы, проанализировать их

соответствие процессам в данной модели, иными словами, провести программный аудит.

3 Порядок выполнения работы

1. В качестве объекта исследования в данной и последующих лабораторных работах (№№ 8...12) рассматривается гипотетическое предприятие с корпоративной информационной системой, построенной в соответствии с ее предметной областью (табл.7.1).

Таблица 7.1 – Варианты объектов исследования (соответствуют порядковому номеру (или второй цифре в нем) студента в группе)

№ варианта	Объект исследования (информационная система)
1	Предприятие оптово-розничной торговли
2	Предприятие автосервиса
3	Предприятие гостиничного типа
4	Предприятие по оказанию страховых услуг
5	Предприятие платных лечебных услуг
6	Предприятие туризма
7	Предприятие транспорта (пассажирские перевозки)
8	Предприятие общественного питания
9	Предприятие по продаже бытовой техники
0	Предприятие строительной отрасли

2. Согласно своему варианту, необходимо определить состав программного обеспечения, соответствующий сфере деятельности предприятия (3-4 программного продукта, информационной подсистемы). Обязательным во всех вариантах является выбор продукта «1С: Предприятие».

Сформулировать постановку задачи.

Пример.

На предприятии оптово-розничной торговли используются три независимые информационные подсистемы:

«Складская система» (учет и анализ товародвижений на складе),

«CRM-система» (учет и анализ продаж и других взаимоотношений с клиентами),

«Бухгалтерская система» (бухгалтерский учет и финансовый анализ), построенная на основе 1С: Бухгалтерия.

Между информационными подсистемами нет информационного обмена.

Задача: Необходимо провести аудит программного обеспечения данного предприятия и предложить обоснованные меры к интеграции ПО с наименьшим количеством вендоров и технологий.

3. Построить бизнес-модель предприятия, в которой должны быть отражены три группировки процессов: основных (технологических), управленческих и обеспечивающих.

4. Выполнить декомпозицию, подробное описание процессов в рамках построенной модели. Рекомендуется отразить результат таблицей, присвоив соответствующие индексы процессам и группам, в которые они входят.

5. Изучить основные функциональные возможности программных продуктов (по п.2).

6. Провести программный аудит. Выяснить, к каким процессам имеет отношение тот или иной программный продукт.

7. Определить процессы, которые не «охвачены» рассматриваемыми программными продуктами. Сформулировать предложения («как охватить?»).

8. Определить проблемы интеграции программных продуктов (информационных подсистем). Сформулировать концепцию интеграции.

9. Составить отчет по проведенному исследованию.

Лабораторная работа №8. Классические методы интеграции. Вариант интеграции на уровне платформ (платформы интеграции)

1 Цель работы

Рассмотреть применение возможных методов интеграции информационных подсистем, ПО которых работает на различных ОС.

2 Краткая теория

Интеграция информационных систем заключается в интеграции одного или нескольких программных компонентов интегрируемых информационных подсистем систем с использованием следующих методологических подходов:

- интеграция платформ;
- интеграция данных;
- интеграция приложений;
- интеграция бизнес – приложений.

С практической точки зрения, наиболее распространенными вариантами интеграции программного обеспечения (приложений) можно выделить:

1) Вариант стандартизации, предполагающий как можно большее использование международных, государственных и отраслевых стандартов, а если каких-то не хватает, то следует использовать корпоративные стандарты, имеет смысл продвигать их в соответствующих организациях для скорейшего развертывания программного обеспечения.

2) Вариант интеграции на уровне брокеров. Среди преимуществ этого варианта следует отметить универсальность (практически всегда можно создать дополнительный программный модуль, который будут обращаться в обе системы, разными способами (например, в одну через базу данных, через RPC). Недостатки: сложность, трудоемкость, а, следовательно, высокая стоимость разработки, внедрения и владения.

3) Вариант(ы) интеграции на уровне данных, когда несколько приложений могут обращаться в одну базу данных или в несколько баз данных, связанных репликациями. Преимущества: низкая стоимость интеграции, а при использовании одной СУБД это становится очень заманчивым решением. Недостатки: если база данных не экранирована хранимыми процедурами и не имеет необходимых ограничений целостности (в виде указания каскадных операций и триггеров), то разные приложения могут приводить данные в противоречивые состояния. Если же база экранирована и целостность обеспечивается, то и в этом случае, в параллельно работающих с одной БД приложениях, будут дублирующиеся части кода, выполняющие одинаковые или похожие операции. Кроме того, при изменениях структуры базы мы будем отдельно переписывать код всех приложений, с ней работающих.

4) Вариант(ы) интеграция на уровне сервисов — это интеграция, основанная на фиксации интерфейсов и форматов данных с двух сторон и позволяющая наладить быструю отработку межкорпоративной бизнес-логики. Недостатки: присутствует фиксация, а если структуры или процессы изменяются, то образуются проблемы и генерируются узко специализированные (частные) программные решения.

5) Вариант интеграции на уровне пользователя. Является крайним (нежелательным) вариантом, означающим неавтоматизированную интеграцию, когда пользователи перемещают данные между системами через копипаст, файлы, почту и др. Такие методы в настоящих лабораторных работах не рассматриваются, хотя, следует сказать, что они, довольно часто применяются в тот период жизненного цикла программных систем, когда программные системы функционально «не готовы», а развитие предприятия происходит интенсивными темпами.

Методы интеграции могут быть классифицированы и по архитектурному признаку.

1) Вертикальная интеграция. В соответствии с этим подходом, системы интегрируются по принципу функциональных экспертиз. Например, выделены две информационные подсистемы: оперативный учет и бухгалтерской учет. При этом, бухгалтерский учет, объективно находится по

вертикали выше оперативного учета. В этом примере подсистема оперативного учета поставляет данные в подсистему бухгалтерского учета.

2) Интеграция «многие ко многим» (звезда, снежинка). В рамках данного подхода каждая из используемых в компании подсистем может при необходимости обращаться к функционалу любой другой подсистемы, при этом каждая из подсистем может также использоваться любой другой подсистемой. Такой тип отношений между элементами называется «многие ко многим». В этом случае имеет место практически неограниченные возможности интеграции подсистем между собой (естественно, если подсистемы технологически позволяют делать это).

3) Горизонтальная интеграция. Данный подход заключается в использовании специализированного «промежуточного» (middleware) ПО, так называемой, корпоративной сервисной шины. Основная задача этого ПО заключается в хранении репозитория функционала корпоративных приложений, подключенных к ней, и обеспечение возможности использования этих функций другими приложениями, также подключенными к этой шине. Взаимодействие между приложениями могут, например, происходить в форме обмена сообщениями или вызова опубликованных функций в виде Веб-сервисов. Подключение системы к шине производится путем создания специального адаптера для каждой системы. После этого «опубликованные» функции системы становятся доступными другим подключенным системам. Преимуществом данного подхода является то, что сами системы могут заменяться в рамках существующей спецификации опубликованных функций. При этом никаких изменений в других системах не требуется. Кроме того, подключение новой системы в достаточной степени стандартизировано и упрощено.

Большая часть из выше перечисленных методов будет реализовываться в последующих лабораторных работах, а в настоящей (№8) – рассматривается решение одного базовых классических методов интеграции, а именно, интеграция на уровне платформ. Суть, которой, кратко заключается в следующем.

Платформа интеграции – это тип программной платформы, которая используется для интеграции различных приложений и источников данных, позволяя бизнесу быстро и легко передавать данные между ними. Это позволяет предприятиям создавать более сложные бизнес-приложения, которые могут безопасно и эффективно охватывать несколько платформ. Обычно платформы интеграции предоставляют широкий спектр функций, включая сопоставление данных, оркестрацию, управление бизнес-правилами и унифицированный мониторинг.

Интеграционные платформы очень важны, поскольку они позволяют предприятиям соединять различные системы и приложения, упрощая управление данными и обеспечивая точность информации. Они помогают предприятиям быстро и безопасно передавать данные между разрозненными приложениями, создавая единый подход к управлению данными и оптимизируя работу для конечных пользователей.

Платформа интеграции работает, обеспечивая плавное соединение и обмен данными между различными программными приложениями, системами или сервисами. Общая схема создания и работы платформы интеграции включает пять этапов.

Этап 1. Идентификация источника данных.

Платформа интеграции начинается с определения источников данных, которые необходимо подключить. Эти источники могут включать в себя различные приложения, базы данных, API, облачные сервисы и многое другое.

Этап 2. Проектирование и картирование рабочего процесса.

Пользователи (или администраторы) определяют рабочий процесс интеграции, в котором описывается последовательность действий, преобразований данных и триггеров, которые должны произойти при перемещении данных между источниками. Кроме того, это включает в себя сопоставление полей данных из источника в цель для обеспечения совместимости.

Этап 3. Преобразование данных и маршрутизация.

Когда данные передаются из одного источника в другой, платформа интеграции применяет к данным необходимые преобразования, такие как преобразование форматов, агрегирование информации или применение бизнес-правил. Платформа маршрутизирует данные в соответствии с определенным рабочим процессом.

Этап 4. Интеграция, выполнение и мониторинг.

Платформа интеграции выполняет рабочий процесс интеграции, обеспечивая передачу данных в соответствии с заданной логикой. Во время этого процесса платформа отслеживает ошибки, исключения и успешное завершение. Он предоставляет оповещения в режиме реального времени и журналы для любых возникающих проблем.

Этап 5. Синхронизация данных и обновления.

Поскольку данные продолжают передаваться между различными источниками, платформа интеграции гарантирует, что изменения и обновления, внесенные в одну систему, точно отражаются в других. Это может включать синхронизацию в реальном времени или запланированные обновления, в зависимости от требований интеграции.

На всех этих этапах платформа интеграции предоставляет следующие ключевые функции.

- Разъемы и адаптеры. Платформа предлагает готовые разъемы или адаптеры для различных приложений, баз данных и систем, обеспечивая бесперебойную связь.
- Отображение и преобразование данных. Пользователи могут определить, как данные должны быть преобразованы и сопоставлены между источниками, чтобы обеспечить последовательный и точный обмен данными.
- Автоматизация рабочего процесса. Платформа автоматизирует последовательность действий, сокращая ручное вмешательство и обеспечивая своевременную передачу данных.
- Обработка ошибок. Платформа обнаруживает и управляет ошибками и исключениями, которые могут возникнуть во время интеграции, предлагая ведение журнала, оповещения и корректирующие действия.
- Безопасность и соответствие. Платформа обеспечивает безопасность и соответствие данным, шифруя данные во время передачи и соблюдая отраслевые правила.
- Мониторинг и аналитика. Администраторы могут отслеживать производительность интеграции, отслеживать потоки данных и собирать информацию о процессах интеграции.
- Масштабируемость. Платформы интеграции предназначены для обработки больших объемов данных и удовлетворения растущих потребностей бизнеса.
- Управление API. Многие платформы позволяют организациям управлять API-интерфейсами и предоставлять их для внешнего или внутреннего использования.
- Централизованное управление. Платформа предоставляет централизованный интерфейс для настройки, управления и мониторинга нескольких интеграций.

Интеграционная платформа – это также, промежуточный слой между интеграционным ПО и целевой системой, призванный повысить эффективность усилий разработчиков. Разработчики взаимодействуют с платформой, а не с API напрямую. При этом кодирование сокращается до минимума, а результат достигается тот же или даже лучший.

Таким образом, платформа интеграции – это жизненно важный инструмент в современной ИТ-среде, предоставляющий предприятиям экономичный и безопасный способ неконфликтного использования приложений и данных. Обеспечивая плавную интеграцию между различными информационными подсистемами, платформа интеграции позволяет

предприятиям лучше сотрудничать и оставаться конкурентоспособными в постоянно меняющейся цифровой среде. Благодаря своим мощным функциям и простоте использования платформа интеграции является отличным выбором для компаний, стремящихся максимизировать свою производительность и эффективность.

3 Порядок выполнения работы

Общая постановка задачи: необходимо интегрировать N-информационных подсистем систем, которые характеризуются следующим:

- имеются программные продукты с несовместимыми лицензиями;
- наблюдается существенная технологическая и концептуальная разница в ПО;

- продукты в составе ПО работают на различных платформах.

1. В соответствии со своим вариантом предприятия:

- поставить цель и задачи интеграции (увеличение прибыли, снижение издержек, сокращение трудозатрат для выполнения задач и др.);

- рассмотреть применение возможных методов интеграции информационных подсистем (программных продуктов) в единую систему;

- выделить основные факторы, влияющие (положительно и отрицательно) на процесс интеграции в информационной системе предприятия.

Привести по одному-двум примерам реализации того или иного метода.

2. Определить назначение и основные характеристики интеграционной платформы.

3. Описать работу интеграционной платформы, которую можно применить на рассматриваемом предприятии (например, платформы интеграции API, Azure или другой). Описание произвести на основе пяти этапов создания и работы интеграционной платформы (приведенных выше), с иллюстрацией примерами.

4. Пояснить примерами позитивное влияние интеграционной платформы на бизнес рассматриваемого предприятия:

- сокращение расходов на выполнение регуляторных обязательств;
- снижение рисков и управление ими;
- обеспечение организационных слияний и поглощений;
- создание целостного представления клиентской базы предприятия;
- обеспечение разумных инвестиций в технологии модернизации данного бизнеса.

5. Рассчитать стоимость платформенной интеграции. В качестве обязательных статей калькуляции использовать затраты на оплату труда,

приобретение (дополнительно) аппаратных средств, программных ресурсов, амортизационные отчисления. Обосновать выбор значений статей калькуляции.

6. Привести примеры (написать) интерфейсов для решения задачи интеграции в информационной системе рассматриваемого предприятия.

7. Составить отчет по проведенному исследованию.

Лабораторная работа №9. SOA-интеграция. Проектирование корпоративной ИС с применением SOA

1 Цель работы

- Разработать компонентную бизнес-модель на основе сервис-ориентированной архитектуры информационной системы предприятия;
- Определить состав сервисов и выполнить их программную реализацию.

2 Краткая теория

Существует ряд определений понятия сервис-ориентированной архитектуры (SOA, англ. service-oriented architecture):

- модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабо связанных (англ. loose coupling) заменяемых компонентов, оснащенных стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам;
- архитектурный стиль для создания ИТ-архитектуры предприятия, использующий принципы ориентации на сервисы для достижения тесной связи между бизнесом и поддерживающими его информационными системами (подсистемами);
- парадигма, предназначенная для проектирования, разработки и управления дискретных единиц логики (сервисов) в вычислительной среде.

Проектирование и разработка приложений на основе SOA-подхода предполагает определение целесообразного набора сервисов (в частности, путем добавления к существующим, новых), построения и реализации компонентной бизнес-модели деятельности предприятия в целях:

- сокращения издержек, за счет упорядочивания процесса разработки;
- расширения повторного использования кода;
- повышения уровня независимости от используемых платформ, инструментов и языков разработки;
- повышения масштабируемости создаваемых программных систем;
- улучшения управляемости информационных систем (подсистем).

Основные характерные черты архитектуры SOA:

1. Такая архитектура не привязана к какой-либо определенной информационной технологии.
2. Использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним.
3. Организация сервисов как слабосвязанных компонентов для построения программных систем.

Концепция проектирования и разработки SOA включает в себя ряд принципов:

- сочетаемость приложений, ориентированных на пользователей;
- многократное использование бизнес-процессов;
- независимость от набора технологий;
- автономность в смысле независимой эволюции (развития ИС), масштабируемости и развертывания.

Таким образом, можно сказать, что SOA – это набор архитектурных принципов, не зависящих от технологий и программных продуктов. Архитектура SOA не привязана к какой-либо определенной технологии и может быть реализована с использованием широкого спектра технологий, включая такие технологии как REST, RPC, DCOM, CORBA или веб-сервисы. SOA может быть реализована, используя один из этих протоколов и, например, может использовать дополнительно механизм файловой системы для обмена данными.

Программные комплексы, разработанные в соответствии с сервис-ориентированной архитектурой, обычно реализуются как набор веб-служб, взаимодействующих по протоколу SOAP, но существуют и другие реализации (например, на базе jini).

Главное, что отличает SOA – это использование независимых сервисов с четко определенными интерфейсами, которые для выполнения своих задач могут быть вызваны неким стандартным способом, при условии, что сервисы заранее ничего не знают о приложении, которое их вызовет, а приложение – не знает, каким образом сервисы выполняют свою задачу.

Интерфейсы компонентов в сервис-ориентированной архитектуре инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом, обеспечивая комбинирование и многократное использование компонентов для построения сложных распределённых программных комплексов, обеспечивая независимость от используемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем.

При проектировании программных (информационных) систем с использованием подходов SOA необходимо:

- разрабатывать и моделировать сервисы на разных уровнях абстракции;
- отделять инструкции от реализации;
- обеспечивать гибкость системы;
- обеспечивать соответствие системы бизнес – требованиям.

Организация по стандартизации OASIS утвердила эталонную модель сервис-ориентированной архитектуры Reference Model for Service-Oriented Architecture 1.0 (SOA-RM). Модель призвана ввести четкую техническую терминологию SOA для разработчиков и архитекторов. Работавший над моделью технический комитет OASIS определяет SOA-RM как «абстрактную базу для понимания основных объектов и взаимосвязей между ними в сервис-ориентированной среде и для разработки согласованных стандартов и спецификаций, поддерживающих такую среду. Модель унифицирует концепции SOA и может быть использована архитекторами для разработки SOA или в обучении SOA».

На рис. 9.1 представлена эталонная модель SOA, предполагающая использование в информационной системе предприятия самой широкой группировки сервисов (программных компонентов).



Рисунок 9.1 – Эталонная модель SOA

Самым распространенным видом сервисов в SOA являются web-сервисы. Для web-сервисов, обычно используется одна из спецификаций: SOAP, REST или GraphQL. Для обмена сообщениями между сервисами используется платфо-независимый язык (например, XML или JSON).

В рамках архитектуры SOA, web-сервисы, это не просто API общего назначения, всего лишь предоставляющие CRUD-доступ к базе данных через HTTP. В каких-то случаях эта реализация и может быть полезной, но ради целостности данных необходимо, чтобы пользователи понимали лежащую в основе реализации модель и соблюдали бизнес-правила. Концепция SOA подразумевает, что web-сервисы являются ограниченными контекстами бизнес-субдоменов (business sub-domain) и отделяет реализацию от решаемых веб-сервисами задач.

Архитектура SOA основывается на открытых стандартах и поддерживает платформенно-независимую бизнес-интеграцию, но она нуждается в общей технологии представления данных, на которой будет базироваться ее инфраструктура. Эта инфраструктура должна поддерживаться всеми участвующими сторонами и, чтобы служить основой для взаимопонимания.

В центре этой инфраструктуры находится технология XML в силу ряда причин:

- XML является фундаментом практически всех стандартов web-сервисов, в том числе XML Schema, SOAP, WSDL (Web Services Description Language) и UDDI (Universal Description, Discovery, and Integration). Эти стандарты опираются на основополагающую концепцию основанных на XML представлений - поддерживаемый во всем мире формат обмена информацией между провайдерами сервисов и инициаторами запросов в SOA.

- Использование XML решает проблему работы с различными форматами данных в различных приложениях, работающих на разных платформах.

- Преимущество XML заключается в простоте представления, являющегося по своей природе текстовым, гибким и расширяемым.

Приведем примеры стандартов, основанных на XML и используемых в SOA.

SOAP. Этот простой основанный на XML протокол позволяет приложениям обмениваться информацией по транспортным протоколам, таким как HTTP. Благодаря использованию XML протокол SOAP является платформенно-независимым, пригодным для использования в Интернете, читабельным, структурированным и текстовым. Благодаря всем этим преимуществам SOAP является рекомендованным и самым широко используемым коммуникационным протоколом для web-сервисов. Этот протокол является также основным коммуникационным протоколом для основанных на SOA решениях.

WSDL. Это документ, написанный на XML и описывающий web-сервис. Он определяет месторасположение сервиса и отображаемые им операции (или методы), позволяющие обращаться к этому сервису. WSDL-файл описывает четыре главные вещи:

1. Сервисы, доступные через интерфейс web-сервиса, такие как список имен методов и сообщений-атрибутов.
2. Тип данных сообщений.
3. Адрес сервиса, используемый для его вызова.
4. Реестры сервисов.

Реестр сервисов представляет собой каталог сервисов, доступных в системе SOA. Он содержит физическое месторасположение сервисов, версии и их срок действия, а также документацию по сервисам.

Реестр сервисов является одним из основных строительных блоков архитектуры SOA и его роль выражает в следующем.

А. Реализует SOA слабое связывание. Храня месторасположения конечных точек сервисов, он устраняет тесное связывание, приводящее к жесткой привязке потребителя к провайдеру. Он также облегчает потенциальные сложности замены одной реализации сервиса другой при необходимости.

В. Позволяет системным аналитикам исследовать корпоративный портфель бизнес-сервисов. Исходя из этого, они могут определить, какие сервисы доступны для автоматизации процессов с целью удовлетворения актуальных бизнес-потребностей, а какие нет. Это в свою очередь позволяет узнать, что нужно реализовать и добавить в портфель, формируя каталог доступных сервисов.

С. Может выполнять функцию управления сервисами, обязывая подписывающиеся сервисы быть согласованными. Это помогает гарантировать целостность руководства (governance) сервисами и стратегий.

Использование подхода, принципов архитектуры SOA, позволяет несколько иначе, определить понятие «бизнес-процесс».

Традиционно, бизнес-процесс определен как набор взаимосвязанных задач, относящихся к деятельности, имеющей функциональные границы. Бизнес-процессы имеют начальные и конечные точки и являются повторяемыми.

Термин бизнес-процесс часто употребляется в среде SOA. В парадигме SOA бизнес-процесс управляет потоком сервисов. Бизнес-процесс управляет потоком событий, вызывает и координирует сервисы и создает контекст для их взаимодействия. Бизнес-процесс, будучи отделенным от реализации сервисов, заботится о ходе деятельности. Такое разделение задач позволяет

больше сконцентрироваться на создании процесса и облегчает изменение процесса в соответствии с новыми требованиями.

3 Порядок выполнения работы

1. Определить основные (программные) компоненты информационной системы для своего варианта предприятия (лабораторная работа № 7). Изобразить компоненты условной схемой, с отражением связи между ними. Число компонентов: от 3 до 5.

2. Отобразить модель полученной системы с помощью диаграммы компонентов на языке UML.

3. Для каждого компонента определить группировки сервисов.

4. Определить состав сервисов (не менее 3) по каждой группировке:

- Сервисы для бизнес-инноваций и оптимизации процессов;
- Инфраструктурные сервисы;
- Сервисы для разработки;
- Управляющие сервисы.

5. Построить модель SOA, взяв за основу рис.9.1, указав в ней выделенные в предыдущем пункте сервисы.

6. Сформировать каталог сервисов. Результат отобразить скриншотом.

7. В качестве примеров, реализовать программно 5-7 веб-сервисов, работающих в синхронном и асинхронном режиме (вопрос - ответ).

8. Отобразить схемой пример интеграции нескольких (2-3) бизнес-процессов (рис.9.2):



Рисунок 9.2 – Схема-шаблон для отображения примера интеграции бизнес-процессов

9. Составить отчет по проведенному исследованию.

Лабораторная работа №10. Интеграция на базе архитектуры с общей шиной данных (ESB)

1 Цель работы

Провести исследование вариантов ESB-решения для информационной системы исследуемого предприятия.

2 Краткая теория

В начале 2000-х годов на рынке программного обеспечения стали появляться решения, сформировавшие кластер под названием «Сервисная шина масштаба предприятия» (Enterprise Service Bus, ESB), или сокращенно «Шина Данных».

Шина Данных (ESB) – это концепция, система, элемент архитектуры IT-ландшафта, используемый для решения задачи интеграции разрозненных информационных систем (подсистем) предприятия в единый программно-аппаратный комплекс с централизованным управлением передачи информации и одновременным применением сервис-ориентированного подхода (SOA).

ESB-системы выполняют следующие основные функции:

- подключение по различным протоколам;
- маршрутизация запросов и сообщений;
- преобразование данных.

Архитектура ESB-системы строится на 3-х компонентах (понятиях):

- 1) набор коннекторов;
- 2) очередь сообщений;
- 3) платформа.

Коннекторы используются для подключения к различным системам и обеспечивают прием и отправку данных.

Очередь сообщений (Message Queue, MQ) служит для организации промежуточного хранения сообщений в ходе их доставки.

Платформа обеспечивает связь коннекторов с очередью, а также организацию асинхронной передачи информации между источниками и приемниками с гарантированной доставкой сообщений и возможностью трансформации. В состав платформы входит средство разработки, позволяющее не только задать правила маршрутизации, но также, при необходимости, определить собственные коннекторы, в том числе, с использованием внешних процедур, реализованных на языках Java, C, C++, C#, Python и др.

Архитектура ESB-системы, построенная для информационной системы конкретного предприятия, может быть названа как «ESB-решение». Преимущества ESB-решений:

- широкий набор коннекторов и масштабируемость решения;
- гибкая маршрутизация данных;
- гарантированная доставка информационных сообщений;
- организация безопасного канала передачи;
- централизованное управление и др.

Пример организации ESB-решения для предприятия типа «торгово-складской комплекс» приведен на рис. 10.1.

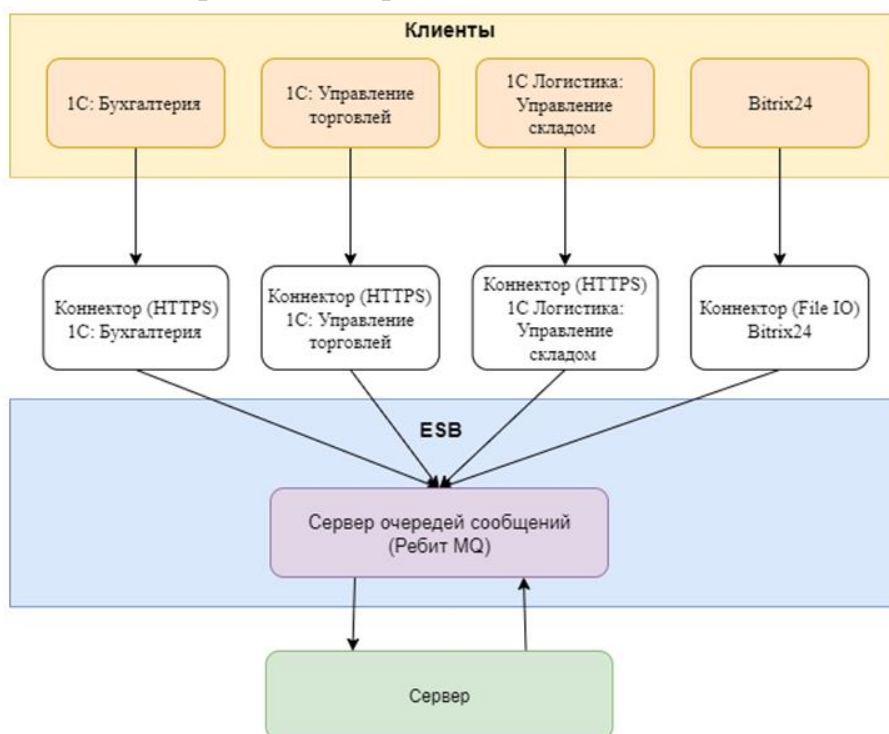


Рисунок 10.1 – Пример ESB-решения

Использование одного сервера обусловлено тесной связанностью системы, межсервисные взаимодействия будут излишне усложнять систему.

Наибольшее распространение получили следующие «типовые» решения: Integration Bus (IBM), Oracle Enterprise Service Bus (ESB), Oracle Service Bus (OSB), BizTalk (Microsoft), ActiveMatrix Service Bus (TIBCO), JBoss Fuse ESB (Red Hat) и некоторые другие.

Например, Oracle Enterprise Service Bus – это основной компонент сервисно-ориентированной архитектуры Oracle, которая обеспечивает слабосвязанный фреймворк для взаимного обмена сообщениями. Служба ESB разрабатывается и конфигурируется при помощи среды Oracle JDeveloper и интерфейса Oracle ESB Control. После этого регистрируется на сервере ESB Server. ESB Server поддерживает мультибиндинг протоколов обмена

сообщениями, включая HTTP/SOAP, JMS, JCA, WSIF и Java, включая синхронную/асинхронную, запрос/ответ или публикация/подписка модели. В настоящее время ESB Server не поддерживает Remote Method Invocation.

Довольно часто при построении композитных приложений приходится сталкиваться с ситуацией, когда различные типы приложений рассчитаны на различные стандарты и схемы интеграции. Также не редка ситуация, когда изменение интеграционных механизмов существующих приложений невозможно или трудоемко по ряду причин: отсутствие разработчика, отсутствие исходного кода и т.д. Интеграционная (также, «корпоративная») шина DATAREON ESB позволяет объединять такие приложения в единое целое, скрывая различия в интеграции на уровне механизмов и настроек типовых коннекторов, что приводит взаимодействие приложений к единой управляемой схеме интеграции.

Таким образом, интеграционная шина данных DATAREON ESB предназначена для построения композитных приложений, использующих различные стандарты и технологии взаимодействия, построенные по разным принципам. Примечательно, что особое внимание с ее помощью уделяется интеграции приложений на платформе «1С: Предприятие» (см. далее).

В DATAREON ESB существуют следующие типы коннекторов:

1. Коннектор SOAP-сервисов.
2. Коннектор REST-сервисов.
3. Коннектор MS SQL.
4. Коннектор IBM DB2.
5. Коннектор Oracle.
6. Коннектор PostgreSQL.
7. Коннектор SharePoint.
8. Коннектор OData 1С.
9. Коннектор TCP.
10. Коннектор Siemens Teamcenter.
11. Коннектор SAP.
12. Коннектор File.
13. Коннектор Pick to Light.
14. Коннектор SFTP.
15. Коннектор Biometry.
16. Коннектор Kardex.
17. Коннектор 1С 7.7.
18. Коннектор 1С 8.x.
19. Коннектор Active Directory.
20. Коннектор ADO.NET.
21. Коннектор RabbitMQ.
22. Коннектор Apache ActiveMQ.
23. Коннектор IBM MQ.
24. Коннектор SMTP.
25. Коннектор IMAP.
26. Коннектор ЛОЦМАН PLM.

Список доступных коннекторов постоянно расширяется. Их актуальный список можно узнать на сайте компании DATAREON (или в частности, по ссылке: <https://sofros.ru/contact/>).

Все коннекторы имеют возможности параметрической настройки подключения к системе-источнику и взаимодействию с ней (рис. 10.2).

Вместе с тем, в составе DATAREON ESB присутствует механизм, позволяющий самостоятельно разрабатывать различные коннекторы на языке Java или языках платформы .Net (рис.10.3).

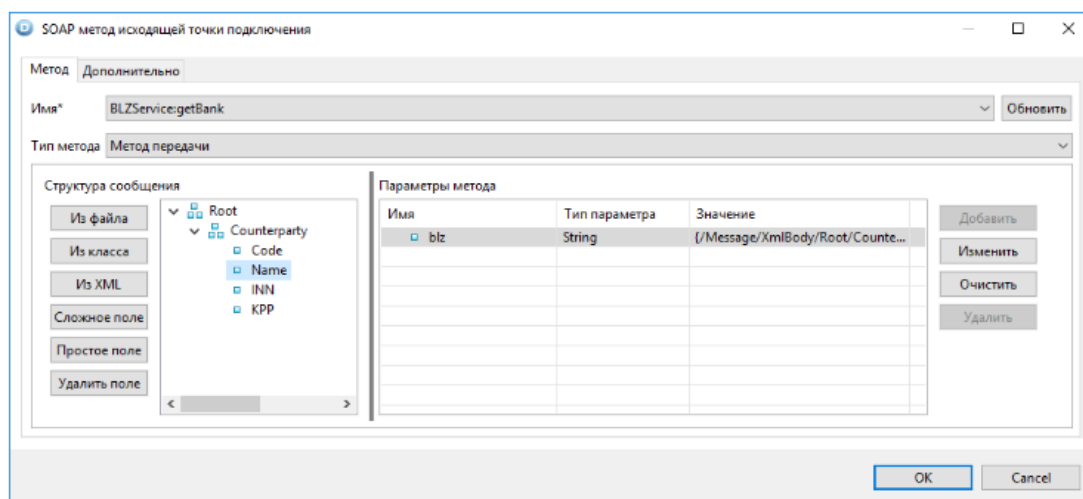


Рисунок 10.2 – Настройка коннекторов DATAREON ESB (метод исходящей точки подключения)

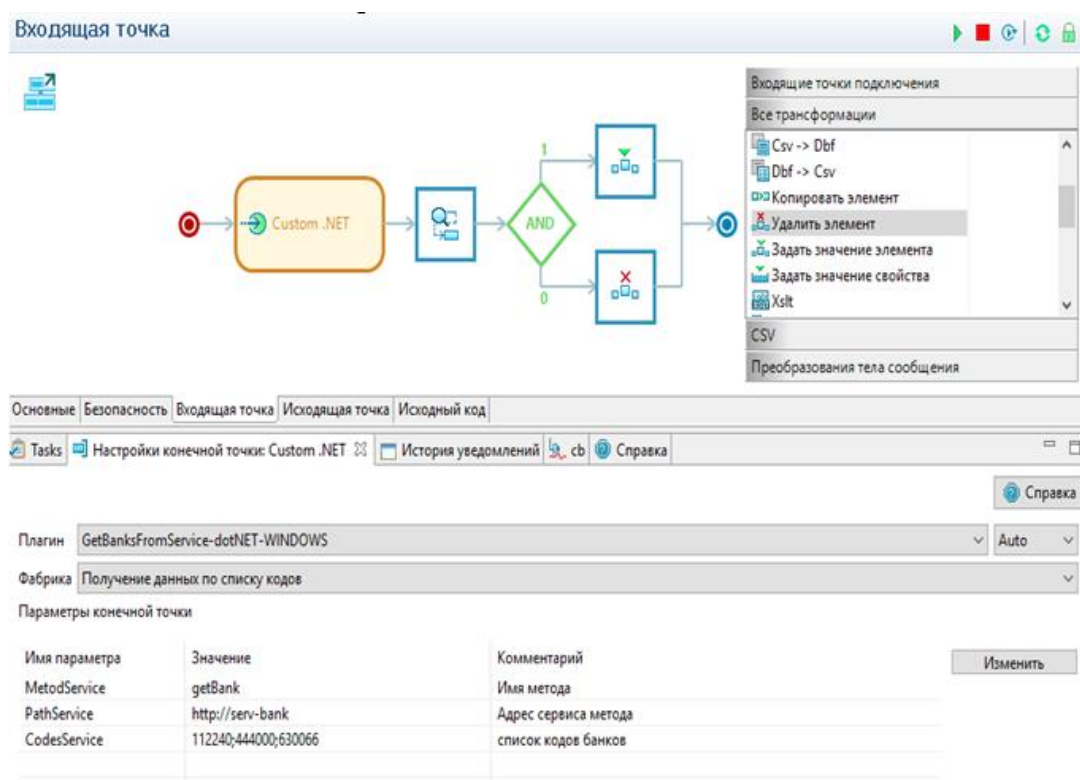


Рисунок 10.3 – Интерфейс DATAREON ESB для разработки коннектора

Кратко скажем о взаимодействии с продуктами на платформе «1С».

Особое внимание фирма DATAREON уделяет программным продуктам, реализованным на платформе «1С». В поставку включена специальная подсистема, написанная на языке V8, которая встраивается в любую систему на платформе «1С» и обеспечивает все необходимые механизмы для интеграции решения с DATAREON ESB. DATAREON ESB предоставляет возможность централизованного автоматического встраивания и обновления

данной подсистемы в конфигурации 1С без необходимости снятия их с поддержки. Правила обработки данных для конфигураций на платформе «1С» создаются и хранятся централизованно в DATAREON ESB. Распространение и обновление обработчиков в системах на платформе «1С» также выполняется централизованно в автоматическом режиме без необходимости модификации самой конечной системы.

Реализованы удобные мастера, которые позволяют создавать обработчики для 1С: (рис. 10.4).

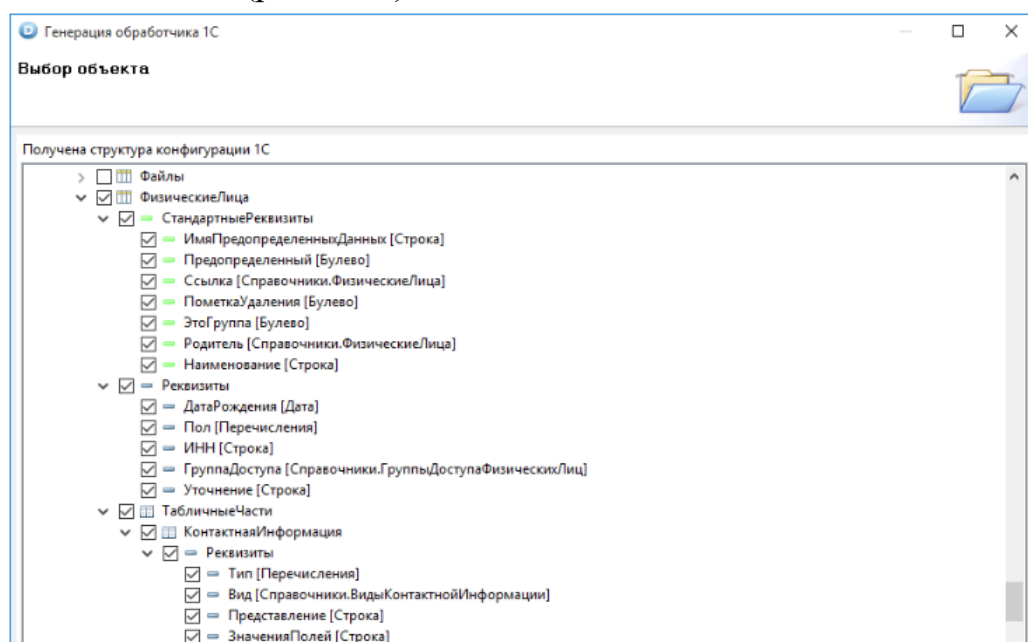


Рисунок 10.4 – Создание обработчика DATAREON ESB в 1С: (фрагмент конфигурации)

Все реализованные интеграционные сценарии учитывают особенности лицензионной политики фирмы «1С», в частности те, которые запрещают прямой доступ к данным системы на платформе 1С через СУБД.

В завершение обзора теоретических основ Шины Данных ESB скажем о возможностях DATAREON ESB в плане централизованного управления интеграционным ландшафтом. Для этого используется экосистема Eclipse.

Использование Eclipse предоставляет пользователю широчайшие возможности по расширению доступного функционала DATAREON ESB. Центр Управления предоставляет мощные и удобные инструменты проектирования потоков данных, разработки алгоритмов трансформации, развитые средства администрирования и контроля.

Центр управления DATAREON ESB может быть интегрирован со средой разработки «1С:Enterprise Development Tools», также построенной на платформе Eclipse, что делает работу в DATAREON ESB еще более удобной для разработчиков на платформе «1С».

В DATAREON ESB присутствует множество визуальных инструментов настройки. Например, мастер настройки и управления информационными потоками (рис.10.5).

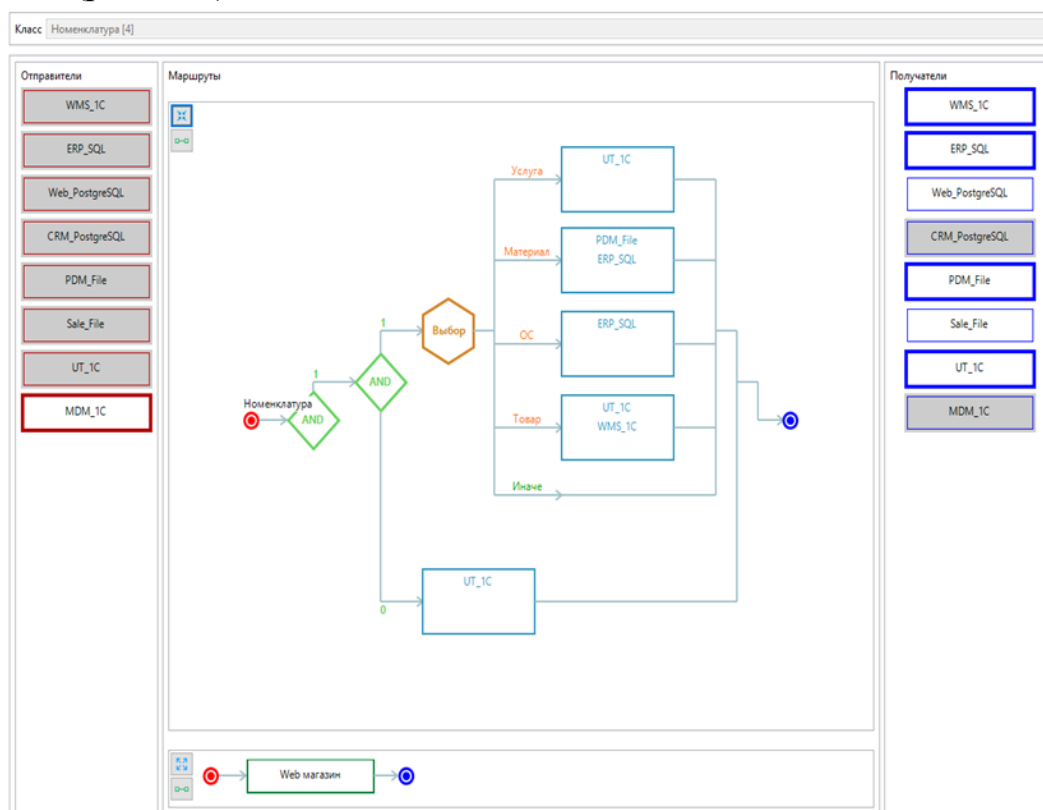


Рисунок 10.5 – Фрагмент работы с мастером настройки и управления информационными потоками в DATAREON ESB

Таким образом, с помощью мастера настройки и управления информационными потоками могут быть реализованы различные пользовательские сценарий подключения к системам-источникам (отправителям сообщений) и системам-потребителям (получателям сообщений).

3 Порядок выполнения работы

1. В соответствии с предприятием, исходными данными по составу программного обеспечения (лабораторная работа № 7) разработать ESB-решение с топологией «одна шина, один сервер». В качестве примера можно взять схему, представленную на рис.10.1. Обосновать, какой тип коннектора (коннекторов) наилучшим образом подходит для данного варианта.

2. Сделать описание возможностей и одного из типовых расширенных коннекторов DATAREON ESB согласно своему варианту (табл. 10.1). Привести схему (с описанием) алгоритма отправки сообщений с помощью анализируемого типового коннектора.

Таблица 10.1 – Выбор типового расширенного коннектора для описания его возможностей

	Номер варианта студента				
	1	2	3	4	5
Коннектор	Active Directory	HTTP – коннектор	Коннектор MS SQL	GIT – коннектор	ADO.NET – коннектор
	6	7	8	9	0
Коннектор	e-mail – коннектор	TCP BRIDGE – коннектор	FTP – коннектор	Коннектор RabbitMQ	Коннектор 1C 8.x

По согласованию с преподавателем может быть выбран другой тип коннектора (из перечня DATAREON ESB).

Для выполнения этого пункта задания можно воспользоваться ссылкой: <https://sofros.ru/products/vozmozhnosti-tipovykh-rasshirennykh-konnektorov/>.

Сделать вывод о возможности, целесообразности и эффективности использования данного типа коннектора для решения задачи интеграции ESB применительно к рассматриваемому предприятию.

3. Провести краткое исследование (пример) технологии создания и использования обработчика DATAREON ESB в 1C: применительно к программному продукту 1C:, который используется в программном обеспечении ИС рассматриваемого предприятия. Привести пример (фрагмент) создания обработчика по аналогии с рис.10.4.

4. Кратко писать возможности Центра управления DATAREON ESB. Составить примерный список получателей-отправителей. Разработать схему управления информационными потоками по аналогии с рис. 10.5.

5. Составить альтернативный вариант схемы ESB-решение с топологией «одна шина, несколько серверов» (также, применительно к своему предприятию) и кратко описать ее работу (пример взаимодействия между приложениями).

6. Составить отчет по проведенному исследованию.

Лабораторная работа №11. Интеграция на базе микросервисов

1 Цель работы

Спроектировать и программно реализовать архитектуру микросервисов программного обеспечения рассматриваемого предприятия.

2 Краткая теория

Впервые понятие «микросервис» появилось в начале 2000-х, а сама концепция такой архитектуры сформировалась к началу 2010 года. В 2014 году

технологии внедрили такие крупные компании, как Netflix, Amazon и Twitter. Сегодня микросервисный подход используется все более активно.

Архитектурный стиль микросервисов – это подход, при котором единое приложение строится как набор небольших сервисов, каждый из которых:

- а) работает в собственном процессе и
- б) коммуницирует с остальными, используя легковесные механизмы (как правило, HTTP).

Микросервисы строятся вокруг определенных бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Существует абсолютный минимум централизованного управления этими сервисами. Сами по себе эти микросервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

Микросервисная архитектура помогает ускорить разработку программного продукта, а также, сделать ее гибкой и управляемой. Проект из неделимого целого (монолитная архитектура) превращается в систему связанных между собой блоков – микросервисов.

Для перехода от монолитной архитектуры к микросервисной, прежде всего, необходимо вынести каждую из частей первой в отдельный сервис (микросервис) – рис.11.1.

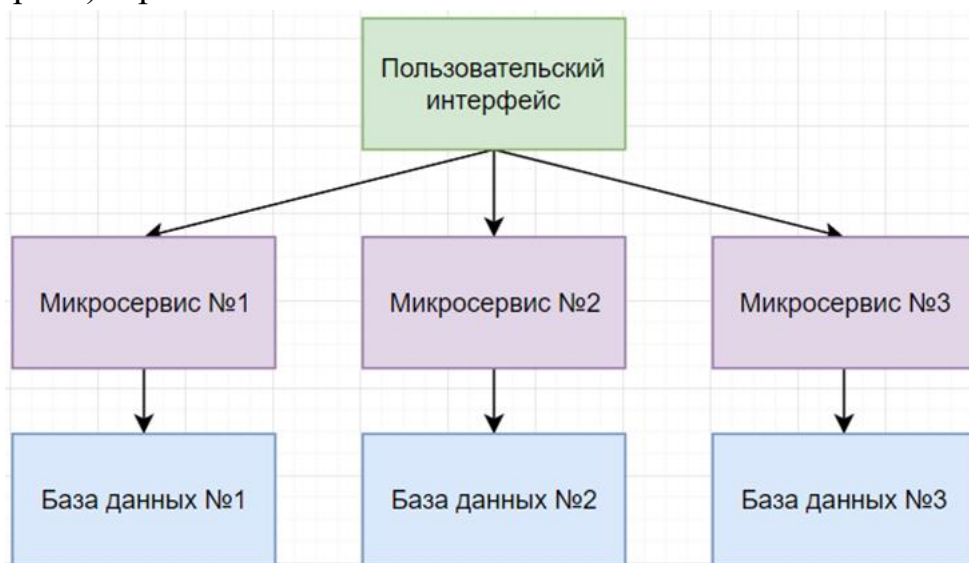


Рисунок 11.1 – Первый шаг перехода к микросервисной архитектуре от монолитной

Микросервисы позволяют достигнуть малой связности частей программного комплекса, что позволяет создать гибкое приложение, готовое к масштабированию. Рис.11.1 отражает общую схему так называемых, Enterprise-приложений. Дословно, Enterprise означает корпоративное, приложение, как правило, включающее три основные части:

- 1) пользовательский интерфейс (состоящий как правило из HTML страниц и javascript-a);
- 2) базу данных (как правило реляционной, со множеством таблиц);
- 3) сервер.

Так, серверная часть обрабатывает HTTP запросы, выполняет доменную логику, запрашивает и обновляет данные в БД, заполняет HTML-страницы, которые затем отправляются браузеру клиента. Любое изменение в системе приводит к пересборке и развертыванию новой версии серверной части приложения.

Таким образом, «микросервисное» Enterprise-приложение состоит из трех соответствующих слоев:

- слой пользовательского интерфейса, с помощью которого пользователи (например, посетители web-сайта предприятия) могут взаимодействовать с системой;
- слой микросервисов, которые будут взаимодействовать между собой и с базой данных (чтобы выполнять запросы пользователя);
- слой данных, через который осуществляется доступ к БД.

Практическое использование таких архитектур приложений позволяет «переиспользовать код» и сделать ПО более комплексным и расширяемым.

Рассмотрим формирование микросервисной архитектуры на примере.

Пример. Пусть информационная система предприятия по ремонту персональных компьютеров и торговле компьютерными комплектующими, содержит три продукта 1С: и программную подсистему CRM системы Bitrix24. Тогда, в первом приближении, можно построить следующую архитектуру (рис. 11.2).

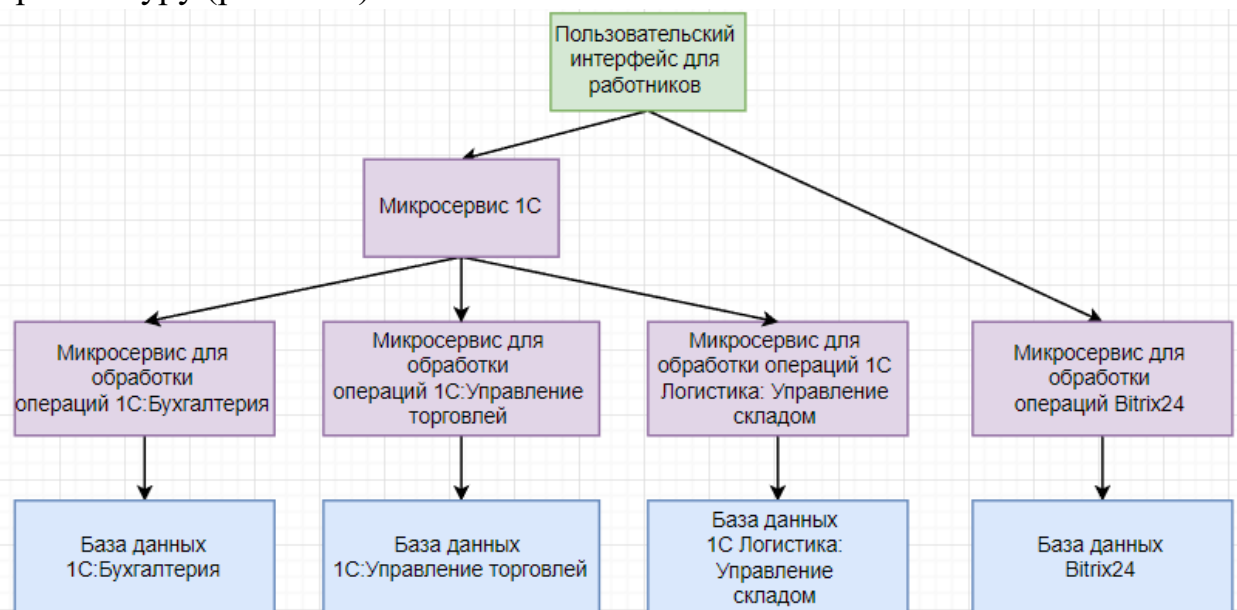


Рисунок 11.2 – Пример формирования микросервисной архитектуры

Выполним описание микросервисов, представленных на рис.11.2, на примерах с использованием ряда методов и функций.

1. Микросервис(ы) для обработки информации по операциям 1С: Бухгалтерия.

Будем использовать метод Recommend(), который позволит выводить результирующую сборку данных по помеченным как *Рекомендованные для продажи запчастей на персональные компьютеры*.

```
class RecommendationService(
    recommendations_pb2_grpc.RecommendationsServicer
):

    def Recommend(self, request, context):
        if request.category not in books_by_category:
            context.abort(grpc.StatusCode.NOT_FOUND, "Category not found")

        books_for_category = books_by_category[request.category]
        num_results = min(request.max_results, len(books_for_category))
        books_to_recommend = random.sample(
            books_for_category, num_results
        )

        return RecommendationResponse(recommendations=books_to_recommend)

    def serve():
        server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
        recommendations_pb2_grpc.add_RecommendationsServicer_to_server(
            RecommendationService(), server
        )
        server.add_insecure_port("[::]:50051")
        server.start()
        server.wait_for_termination()

if __name__ == "__main__":
    serve()
```

2. Микросервис для вывода информации по клиентам при взаимодействии с 1С: Управление торговлей.

Используем метод GetClientInfo(), который вызывается для получения информации по записаному как «агром» пользователю. Заголовок содержит элемент locale, устанавливающий русский язык для ответных сообщений, и

элемент token – авторизационный токен, выданный OAuth-сервером предприятия с согласия пользователя.

```
POST /v4/soap/ HTTP/1.1
Content-Length: 686
Content-Type: text/xml; charset=utf-8
SOAPAction: "API#GetClientInfo"
Host: api.direct.yandex.ru

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns0="API"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <locale>ru</locale>
    <token>0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f</token>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns0:GetClientInfo SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <params soapenc:arrayType="xsd:string[]">
        <xsd:string>agrom</xsd:string>
        <Login>agrom</Login>
      </params>
    </ns0:GetClientInfo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3. Микросервис для получения информации по складу при использовании 1С: Управление складом. Данный микросервис является масштабируемым, так как, при превышении лимита вместимости базы данных происходит динамическое увеличение вместимости БД.

```
функция ПолучитьОстатки(Товар, Склад)
    СтруктураОтбор = Новый Структура;
    СтруктураОтбор.Вставить("Номенклатура", Товар);
    СтруктураОтбор.Вставить("Склад", Склад);
    СтруктураОстатков = РегистрыНакопления.ТоварыНаСкладах.Остатки(ТекущаяДата(), СтруктураОтбор,
    "Номенклатура, Склад", "ВНаличии") ;
    Если СтруктураОстатков.Количество() > 0 Тогда
        Возврат СтруктураОстатков[0].ВНаличии;
    Иначе
        Возврат 0;
    КонецЕсли;
конецфункции
```

4. Микросервис для динамического изменения данных в CRM-форме Bitrix24.

Если нужно разместить на странице несколько кнопок, которые будут заполнять значения одной CRM-Формы, которые будут приходить по ajax - нужно выполнить следующее:

- добавить на страницу скрытую кнопку с классом, b24-web-form-
rорur-btn-X, где X = ID Битрикс формы;
- использовать функцию reinitB24Dvsform() - в данном случае на вход
идёт значение поля формы LEAD_TITLE, можно вызывать в скрипте, или по
событию onclick.

Программный код данного микросервиса:

```
<?php

// JS код
/**
 * Инициализируем настройки формы, ОБЯЗАТЕЛЬНО указываем ref
 */
function initB24CrmDvsForm(b24val) {
    return { "id": "8", "lang": "ru", "sec": "y8awav", "type": "button", "click": "", "ref": "https://ACCOUNT.bitrix24.ru/bitrix/js/crm/form_loader.js", "fields": {
        'values': b24val
    } };
}
var b24paramsload = initB24CrmDvsForm({ });
/**
 * Стандартное добавление скрипта и переменных для скрипта Б24
 */
(function(w,d,u,b){ w['Bitrix24FormObject']=b;
    w[b] = w[b] || function(){ arguments[0].ref=u;
        (w[b].forms=w[b].forms||[]).push(arguments[0]) };
    if(w[b]['forms']) return;
    s=d.createElement('script');
    r=1*new Date();
    s.async=1;
    s.src=u+'?'+r;
    h=d.getElementsByTagName('script')[0];
    h.parentNode.insertBefore(s,h);
})(window,document,b24paramsload.ref,'b24form');
b24form(b24paramsload);
/**
 * Реинициализируем форму, частично повторяем функцию init у Bitrix24FormLoad
er
 */
function reinitB24Dvsform(nVal) {
    if(!window.Bitrix24FormObject || !window[window.Bitrix24FormObject])
        return;
    if(!window[window.Bitrix24FormObject].forms)
        return;
    // Уничтожаем форму
    Bitrix24FormLoader.unload(b24paramsload);
```

```

// Пересоздаём параметры формы
b24paramsload = initB24CrmDvsForm({'LEAD_TITLE': nVal});
// Иницилируем форму с новыми данными
Bitrix24FormLoader.params = b24paramsload;
Bitrix24FormLoader.init();
// Открываем попап
Bitrix24FormLoader.showPopup(b24paramsload);
}
?>

```

Далее. Если поставить задачу, то ее микросервисное архитектурное решение можно отразить следующей общей схемой (рис. 11.3).

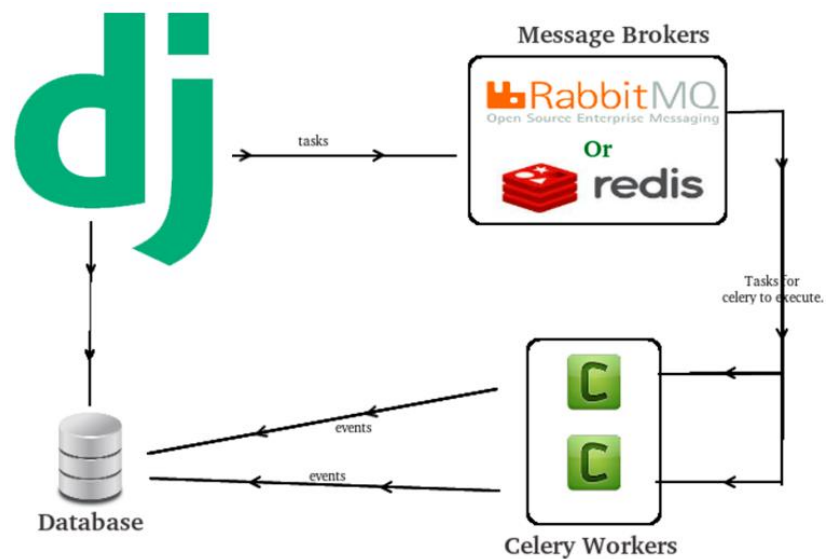


Рисунок 11.3 – Архитектура (общего вида) микросервисов для выбранной (поставленной) задачи

Пояснение к рисунку.

Message Brokers - архитектурный паттерн в распределённых системах; приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником.

Django - свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

Celery Workers - распределенная очередь задач для систем UNIX. Она позволяет выгрузить работу из приложения на Python. Как только пользователь интегрирует Celery в свое приложение, то сразу может отправлять трудоемкие задачи в очередь задач Celery. Таким образом, веб-приложение может продолжать быстро реагировать на запросы пользователей,

в то время как Celery асинхронно выполняет операции, требующие больших затрат в фоновом режиме.

Database - совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

3 Порядок выполнения работы

1. Используя исходные данные по информационной системе рассматриваемого предприятия составить схему микросервисной архитектуры (по аналогии с рис. 11.2).

2. Выполнить конкретизацию и описание микросервисов, используя вышеприведенный пример.

3. Используются результаты выполнения лабораторной работы №9, составить таблицу соответствия микросервисной архитектуры архитектуре SOA.

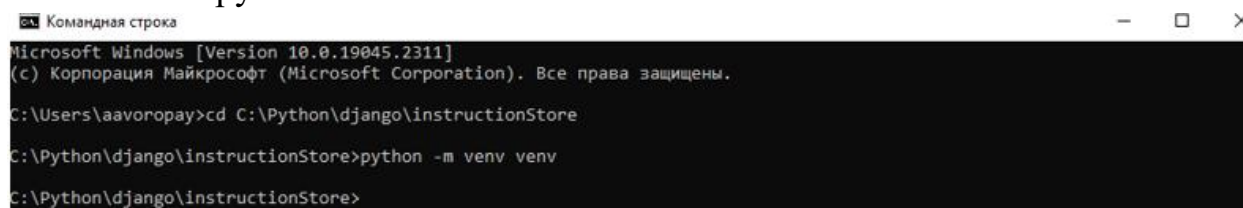
4. Поставить задачу. Конкретизировать ее архитектурное (микросервисное) решение с помощью рис. 11.3.

5. Реализовать спроектированную архитектуру программно. Один из микросервисов должен быть масштабируемым. Микросервисы должны работать в синхронном и асинхронном режимах работы.

Рекомендуется данный пункт работы выполнить в следующей последовательности.

5.1 Создать каталог, где будет формироваться Django-проект.

5.2 Перейти по созданному пути в командной строке и создать локальное окружение:



```
Командная строка
Microsoft Windows [Version 10.0.19045.2311]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\aaavoropay>cd C:\Python\django\instructionStore
C:\Python\django\instructionStore>python -m venv venv
C:\Python\django\instructionStore>
```

В папке проекта появится свое виртуальное окружение. В этом можно убедиться, заметив созданную папку с подкаталогами и файлами venv.

5.3 Запустить среду разработки, например, PyCharm. Выбирать путь расположения файла созданный каталог.

5.4 Установить необходимые библиотеки с помощью pip модуля (Django, Celery и прочие).

5.5 Создать проект командой – `django-admin startproject <название проекта>` и далее создаем программный продукт, описанный в задаче.

В разрабатываемой задаче масштабируемым микросервисом является непосредственно проект Django и Celery Worker, в то время как Redis не является масштабируемым.

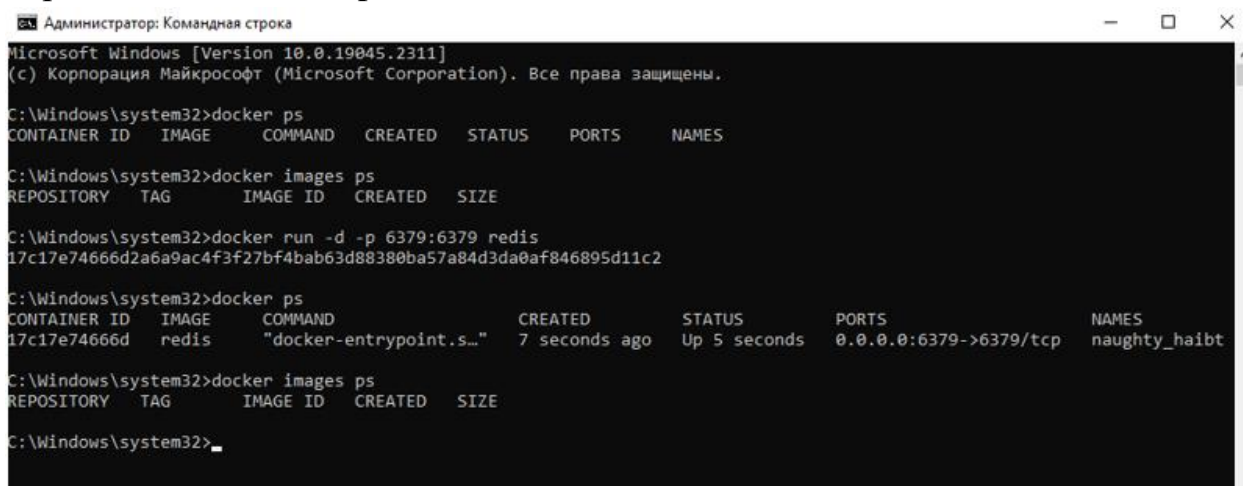
Django проект работает благодаря Celery как в синхронном так и асинхронном порядке.

6. Разработать тест надежности работы микросервисов.

Тестом надежности микросервисов будет являться непосредственно запуск всех микросервисов для работы над поставленной задачей.

Рекомендуется выполнить данный пункт в следующей последовательности.

6.1 Необходимо запустить приложение Docker, с помощью которого будем запускать брокер сообщений (Redis). Предварительно надо скачать образ Redis с сайта <https://hub.docker.com>.

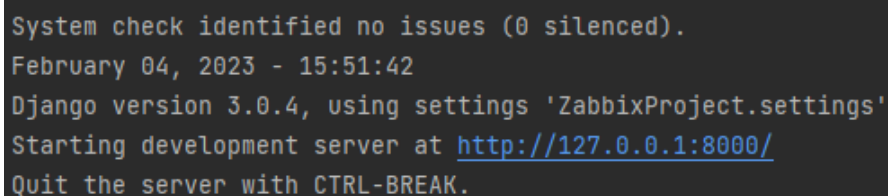


```
Администратор: Командная строка
Microsoft Windows [Version 10.0.19045.2311]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\system32>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
C:\Windows\system32>docker images ps
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
C:\Windows\system32>docker run -d -p 6379:6379 redis
17c17e74666d2a6a9ac4f3f27bf4bab63d88380ba57a84d3da0af846895d11c2
C:\Windows\system32>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED      STATUS      PORTS          NAMES
17c17e74666d   redis     "docker-entrypoint.s..."   7 seconds ago   Up 5 seconds   0.0.0.0:6379->6379/tcp   naughty_haibt
C:\Windows\system32>docker images ps
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
C:\Windows\system32>
```

Осуществить ввод необходимых команд для запуска Redis с помощью Docker.

6.2 Далее необходимо запустить сервер Django. Делается это в терминале командой `python manage.py runserver` находясь в директории с этим файлом. После запуска в терминале будет отображено следующее сообщение о успешном запуске Django сервера:



```
System check identified no issues (0 silenced).
February 04, 2023 - 15:51:42
Django version 3.0.4, using settings 'ZabbixProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

6.3 Подключаем Redis. Для этого вводит в терминале команду: `celery -A ZabbixProject worker -l info`. После подключения в терминале будет отображено следующее сообщение об успешном подключении Redis:

```
[2023-02-04 15:56:12,260: INFO/MainProcess] celery@KND-WN-000036 ready.
```

6.4. Переходим по ссылке <http://127.0.0.1:8000/instructionStore/login/>.
Результат тестирования надежности работы микросервисов успешен.

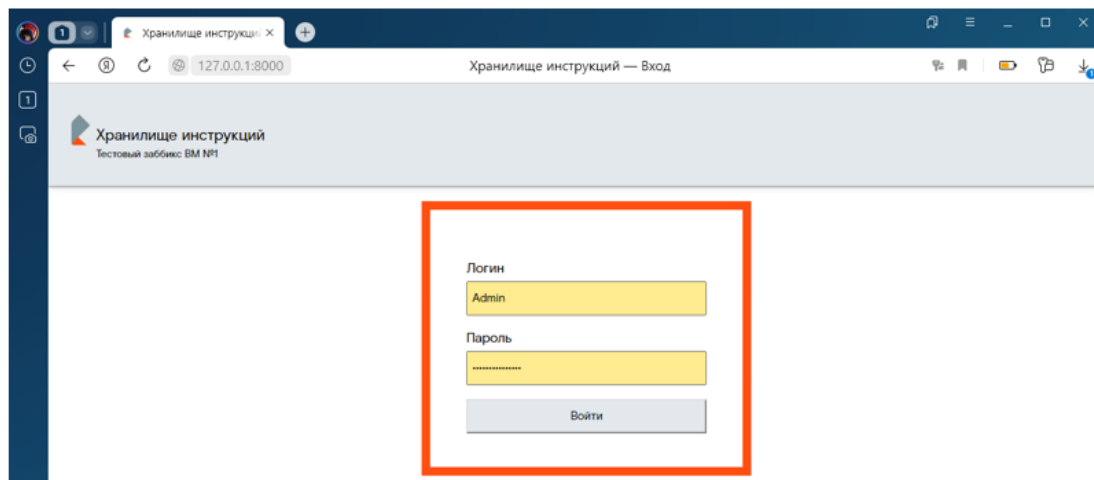
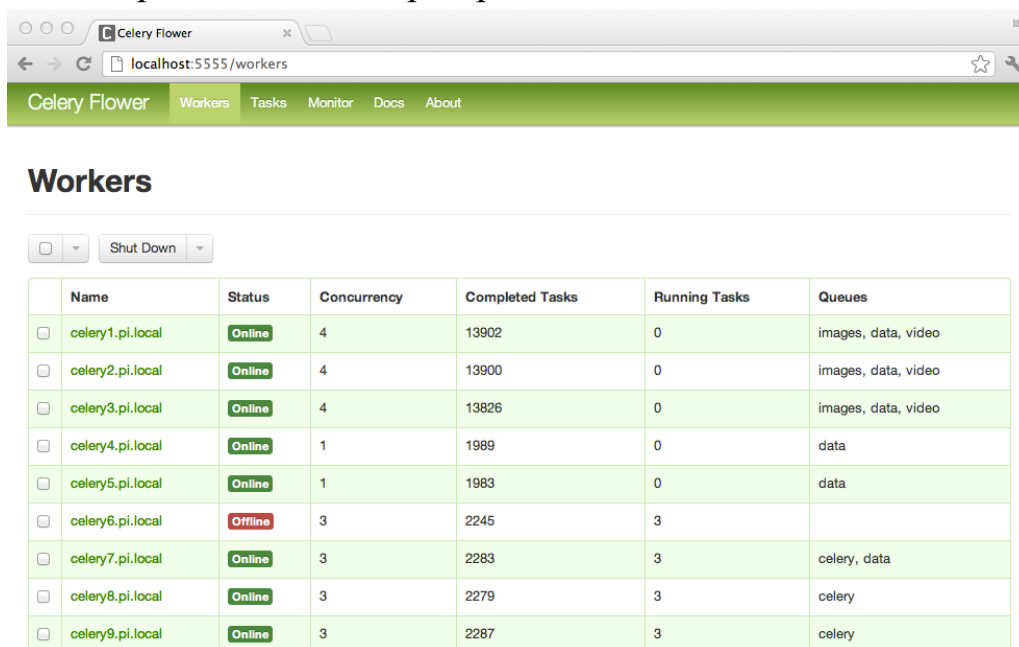


Рисунок 11.4 – Результат тестирования надежности работы микросервисов

7. Реализовать программно мониторинг работы микросервисов

Мониторинг работы микросервисов будет осуществляться с помощью другого микросервиса – Celery Flower. Чтоб его запустить необходимо установить библиотеку flower с помощью команды `pip install flower`, после запустить, пробросив порты командой `flower -A ZabbixProject --port=5555`. Далее переходим по ссылке 127.0.0.1:5555 и наблюдаем работающий Celery Flower, мониторящий наши микросервисы.



	Name	Status	Concurrency	Completed Tasks	Running Tasks	Queues
<input type="checkbox"/>	celery1.pi.local	Online	4	13902	0	images, data, video
<input type="checkbox"/>	celery2.pi.local	Online	4	13900	0	images, data, video
<input type="checkbox"/>	celery3.pi.local	Online	4	13826	0	images, data, video
<input type="checkbox"/>	celery4.pi.local	Online	1	1989	0	data
<input type="checkbox"/>	celery5.pi.local	Online	1	1983	0	data
<input type="checkbox"/>	celery6.pi.local	Offline	3	2245	3	
<input type="checkbox"/>	celery7.pi.local	Online	3	2283	3	celery, data
<input type="checkbox"/>	celery8.pi.local	Online	3	2279	3	celery
<input type="checkbox"/>	celery9.pi.local	Online	3	2287	3	celery

Рисунок 11.6 – Мониторинг работы микросервисов

8. Составить отчет по проведенному исследованию.

Лабораторная работа №12. Практическое освоение понятия ETL. Системы и инструменты интеграции корпоративных данных.

1 Цель работы

Практически реализовать:

- систему управления мастер-данными на основе CDI (интеграция клиентских данных);
- контроллер мастер – данных по сбору информации о предприятиях профильной (аналогичной) деятельности.

2 Краткая теория

Дословный перевод (англ.) понятия ETL (Extract, Transform, Load) означает извлечение, преобразование и загрузка. Информационные системы современных предприятий обеспечивают обработку различных типов и видов данных из множества источников (систем менеджмента). Существующие системы ETL реализуют общий алгоритм:

- 1) извлечение данных из одного (или нескольких) источников;
- 2) подготовка данных к интеграции;
- 3) загрузка (извлечение) данных;
- 4) передача данных в общую корпоративную БД.

Практическое применение ETL-систем обеспечивает в результате широкие возможности по повышению конкурентоспособности бизнеса предприятия, в частности:

- анализ исторических записей для оптимизации процесса продаж;
- корректировка цен и запасов в реальном времени;
- использование машинного обучения и искусственного интеллекта для создания прогнозных моделей развития бизнеса;
- планирование финансовых потоков, прибыли и т.д.

К инструментарию ETL-систем, реализующих конкурентные возможности можно отнести облачную миграцию, хранилище данных, машинное обучение, интеграцию маркетинговых данных, интеграцию данных IoT, репликацию базы данных, программы бизнес-аналитики.

Большинство инструментов ETL с открытым исходным кодом помогают в управлении пакетной обработкой данных и автоматизации потоковой передачи информации из одной системы данных в другую. Эти рабочие процессы важны при создании хранилища данных для машинного обучения.

Некоторые из бесплатных и открытых инструментов ETL принадлежат поставщикам, которые в итоге хотят продать корпоративный продукт, другие

обслуживаются и управляются сообществом разработчиков, стремящихся демократизировать процесс.

В числе, наиболее популярных Open source ETL-инструментов интеграции данных, находятся Apache Airflow, Apache Kafka, Apache NiFi, CloverETL (CloverDX), Jaspersoft ETL, Apatar.

С появлением облачных технологий, SaaS и больших данных значительно выросло число источников информации, что вызвало рост спроса на более мощную и сложную интеграцию корпоративных данных. Возникло понятие MDM, как важнейшее, перспективное направление развития и совершенствования ETL-системы на предприятии.

Управление основными данными (управление мастер-данными, англ. Master Data Management, MDM) – совокупность процессов и инструментов для постоянного определения и управления основными данными компании (в том числе справочными). Можно встретить и другое название – управление справочными данными (англ. Reference Data Management, RDM). К этому варианту примыкает используемое на постсоветском пространстве фактически как синоним MDM понятие «управление нормативно-справочной информацией» (НСИ). Изначально в его рамках подразумевались только фиксированные, исходно наполняемые и изменяемые только в редких случаях справочники, что ближе по первоначальному смыслу к конфигурационным данным.

Таким образом, МДМ-система, рассматривается как развитие базового представления о ETL-системе предприятия и выражает целостный взгляд на все составляющие его бизнеса, в том числе на источники данных, авторство, качество, полноту, на потенциально эффективное использование данных.

Уточним понятие «мастер-данные».

Мастер-данные – это данные с важнейшей для ведения бизнеса информацией, например, о клиентах, продуктах, услугах, персонале, технологиях, материалах и т.д. Заметим, что такие данные сравнительно не часто изменяются и не являются транзакционными.

Цель управления основными данными – удостовериться в отсутствии повторяющихся, неполных, противоречивых данных в различных областях деятельности предприятия. Пример некачественного управления основными данными: работа банка с клиентом, который уже использует кредитный продукт, однако по-прежнему получает предложения взять такой кредит. Причина неправильного поведения банка в этом случае: отсутствие актуальных данных о клиенте в отделе по работе с клиентами.

Подходом к управлению основными данными предусматриваются такие процессы как сбор, накопление, очистка данных, их сопоставление, консолидация, проверка качества и распространение данных внутри самого предприятия, обеспечение их последующей согласованности и контроль использования в различных операционных и аналитических приложениях.

Широко известными вариантами систем управления мастер – данных (MDM) являются:

- RDM (Reference Data Management), в общем и целом, это – управление НСИ. На первый взгляд, это – не мастер-данные, это – другой домен, но отличие НСИ от мастер-данных заключается лишь в централизации. Тот же инструментарий, что используется для управления мастер-данными, может быть использован и для НСИ с единственным отличием: все системы по отношению к RDM работают в режиме только чтения, а все изменения обязательно проходят через соответствующие бизнес-процессы.

- CDI (Customer Data Integration) – интеграция клиентских данных. Под этим термином рассматриваются системы, узко «заточенные» на работу с мастер-профилем клиента. Подразумевается, что эти системы агрегируют максимальный объем информации о клиенте и предоставляют всем другим системам сервисы поиска, создания и обновления клиентских записей, а также двунаправленной синхронизации.

- PIM (Product Information Management) – система управления информацией о продуктах. Это – специализированный вид MDM, заточенный на информацию о продуктах, их свойствах и атрибутах. Например, такая система может быть «авторитетным источником» информации о продуктах для интернет-витрины, печатного каталога, ERP-системы и колл-центра одновременно, обеспечивая целостность представления и единство информации.

В данной лабораторной работе предполагается практическое знакомство с системой CDI.

Для реализации варианта CDI агрегацией данных на уровне OLAP–системы воспользуемся сервером Zabbix.

Zabbix – это система с веб-интерфейсом, которая позволяет собирать различные данные с устройств. С её помощью можно производить мониторинг сети, серверов, виртуальных машин, баз данных, приложений и устройств. Zabbix может получать данные о состоянии устройства: объёме памяти, скорости работы процессора, температуре машины, логах и др. Помимо этого, система способна наблюдать за бизнес-метриками, например, за скоростью продажи продукта.

Zabbix – это open-source система мониторинга корпоративного уровня.

На текущий момент Zabbix одна из самых популярных и функциональных бесплатных систем мониторинга. Благодаря простой установке и настройке Zabbix можно использовать для мониторинга крупных инфраструктур с сотнями серверов, так и для мелких конфигураций.

Имеется множество информационных ресурсов по Zabbix.

Например, ресурс <https://winitpro.ru/index.php/2020/03/10/zabbix-manual-ustanovka-nastrojka/>

Подробно описывает, как выполнить установку и базовую настройку сервера Zabbix 4.4 с веб-интерфейсом на базе Linux Ubuntu и CentOS, установить агенты Zabbix на сервере Windows и Linux, и добавить новые хосты в систему мониторинга.

Для установки Zabbix Server на Windows можно, например, воспользоваться ссылкой:

<https://lolpc.ru/kak-ustanovit-zabbix-server-na-windows/>

3 Порядок выполнения работы

Часть 1. Реализовать вариант CDI с агрегацией данных на уровне OLAP – системы.

1. Произвести установку Zabbix в зависимости от используемой студентом платформы (Linux Ubuntu или Windows).

Инсталляция Zabbix сервера состоит из:

- бинарника zabbix_server (обычно работает как сервис);
- MySQL (MariaDB)/PostgreSQL базы данных;
- Веб сервера Apache2/Nginx с обработчиком PHP;
- Файлов самого frontend сайта (.php, .js, .css и т.д.).

2. Создать узел сети. Руководствоваться инструкцией (ссылки на Интернет-ресурсы). Описать последовательность установки с приложением рисунков – скриншотов.

3. Создать простейшую БД для рассматриваемого предприятия с использованием СУБД типа MySQL или PostgreSQL. Для понимания, необходимо первоначально построить схему ИС на базе OLAP-технологии извлечения, обработки данных и представления информации (общий ее вид представлен на рис. 12.1). В базе данных необходимо отразить различные (8-10) показатели бизнеса рассматриваемого предприятия.

4. Произвести настройку Zabbix. Создать элементы данных (которые будут получать эти данные).

5. Перейти в пункт Preprocessing и настроить процесс препроцессинга перед тем, как полученные данные отправятся в базу данных (рис. 12.2).



Рисунок 12.1 – Построение ИС на базе OLAP-технологии извлечения, обработки данных и представления информации

Рисунок 12.2 – Настройка препроцессинга

В данных настройках препроцессинга (рис.12.2) настраивается процесс изменения за секунду, который определяется по формуле:

(текущее значение-предыдущее значение)/(текущее время-предыдущее время)

и умножение полученного значения на число 0.01.

6. Реализовать примеры работы CDI с базой данных. Выполнить к ним краткое пояснение.

7. Составить отчет по Части 1 проведенного исследования.

Часть 2. Реализовать контроллер мастер – данных по сбору информации о предприятиях профильной (аналогичной) деятельности, функционирующих на территории Краснодарского края.

Контроллер мастер-данных рекомендуется реализовать на языке C# с использованием Windows Forms. Перед началом создания проекта необходимо установить Visual Studio и необходимые для него расширения. Студенту предоставляется возможность свободного выбора методов, библиотек,

инструментария информационно-поисковых систем, платформы, языка программирования, web-технологии и т.д. Как вариант, возможно выполнение этой части лабораторной работы на основе системы RDM.

Последовательность выполнения работы можно представить в следующем, общем виде.

1. Создать и настроить проект контроллера мастер-данных.
2. Сформировать список предприятий-аналогов. Обязательным условием является наличие сайта у предприятия.
3. Сформировать базу данных по 8-10 предприятиям-аналогам.
4. Разработать главную форму Windows Forms для реализации статических и динамических запросов.
5. Разработать итоговую форму таблицы с информацией информации о предприятиях. Примерная форма приведена в табл.12.1.

Таблица 12.1 – Примерная форма итоговой таблицы по сбору данных о предприятиях аналогичной деятельности: _____

(указать вид деятельности)

Наименование предприятия	Место расположения (город)	Значения показателей бизнеса			Адрес сайта
		Численность работников	Доходы, тыс. руб.	Прибыль, тыс. руб.	

По желанию студента можно выбрать другие показатели предприятий, функционирующих на территории Краснодарского края, например, официальные реквизиты (ИНН, коды статистики, юридический адрес, ФИО руководителя и т.д.)

6. Произвести автоматизированное формирование данных в итоговой таблице.
7. Выполнить аналитику полученных данных. Построить диаграммы, графики.
8. Составить отчет по Части 2 проведенного исследования.

Список рекомендуемой литературы

1. Администрирование в информационных системах: Учебное пособие / М.Н. Беленькая, С.Т. Малиновский, Н.В. Яковенко. – М.: Гор. линия-Телеком, 2011. – 400 с.; [Электронный ресурс] Режим доступа: <http://znanium.com/catalog/product/308914>.
2. Администрирование баз данных. СУБД MS SQL Server [Электронный ресурс]: учеб. пособие / О. П. Култыгин. – М.: МФПА, 2012. – 232 с. Режим доступа: <http://znanium.com/catalog/product/451114>.
3. Гультияев А. Виртуальные машины. Несколько компьютеров в одном /А. Гультияев [Электронный ресурс]: Режим доступа: https://4italka.su/komputeryi_i_internet/programmyi/267031/fulltext.htm.
4. Исаев, Г. Н. Управление качеством информационных систем : учебное пособие / Г.Н. Исаев. – Москва : ИНФРА-М, 2022. — 248 с. — (Высшее образование: Бакалавриат). – DOI 10.12737/19428. - ISBN 978-5-16-011794-2. – Текст : электронный. – URL: <https://znanium.com/catalog/product/1860098>.
5. Когаловский, М. Р. Перспективные технологии информационных систем / Когаловский М.Р., – 2-е изд., (эл.) - Москва :ДМК Пресс, 2018. – 287 с.: ISBN 978-5-93700-042-2. – Текст : электронный. – URL: <https://znanium.com/catalog/product/982544>.
6. Краткое руководство. Создание виртуальной машины под управлением Windows на портале Azure [Электронный ресурс]: Режим доступа: <https://learn.microsoft.com/ru-ru/azure/virtual-machines/windows/quick-create-portal>.
7. Ленд, М. Python: Непрерывная интеграция и доставка : практическое руководство / М. Ленд ; пер. с англ. А. Е. Мамонова, Д. А. Беликова. – Москва : ДМК Пресс, 2020. – 168 с. – ISBN 978-5-97060-797-8. – Текст : электронный. – URL: <https://znanium.com/catalog/product/1210637>.
8. Морозова О.А. Интеграция корпоративных информационных систем: учебное пособие / О.А. Морозова. – М.: Финансовый университет, 2014. – 140 с.
9. Назаров, С. В. Архитектура и проектирование программных систем : монография / С.В. Назаров. – 2-е изд., перераб. и доп. – Москва : ИНФРА-М, 2023. – 374 с. – (Научная мысль). – DOI 10.12737/18292. – ISBN 978-5-16-011753-9. – Текст : электронный. – URL: <https://znanium.com/catalog/product/1895672>.

10. Основы операционных систем. Часть I. Виртуальные машины и серверные системы: учеб. пособие / А. А. Лавров, А. В. Лупин, И. А. Малышев. – СПб., 2022. – 100 с.; [Электронный ресурс]: Режим доступа: <https://elib.spbstu.ru/dl/5/tr/2022/tr22-70.pdf/download/tr22-70.pdf>.

11. Организация сетевого администрирования : учебник / А.И. Баранчиков, П.А. Баранчиков, А.Ю. Громов, О.А. Ломтева. – Москва : КУРС : ИНФРА-М, 2024. – 384 с. – ISBN 978-5-906818-34-8. – Текст : электронный. – URL: <https://znanium.com/catalog/product/2096066>.

12. Руководство пользователя Oracle® VM VirtualBox для версии 6.1 [Электронный ресурс]: Режим доступа: <https://docs.oracle.com/en/virtualization/virtualbox/6.1/user/Introduction.html#Глава-1-Первые-шаги>.

13. Рындин, А. А. Современные стандарты информационного взаимодействия систем: учеб. пособие / А. А. Рындин, Э. Р. Саргсян; ФГБОУ ВО «Воронежский государственный технический университет». – Воронеж: Изд-во ВГТУ, 2021. – 144 с.

14. Системное администрирование персонального компьютера [Электронный ресурс]: курс лекций / Н.А. Иванов. – М.: МИСИ-Московский государственный строительный университет, 2017. – 170 с. Режим доступа: <http://znanium.com/catalog/product/1018544>.

15. Солодушкин, С. И. Разработка программных комплексов на языке JavaScript : учебное пособие / С. И. Солодушкин, И. Ф. Юманова ; под общ. ред. В. Г. Пименова ; Министерство науки и высшего образования Российской Федерации, Уральский федеральный университет. – Екатеринбург : Изд-во Уральского ун-та, 2020. – 132 с. – ISBN 978-5-7996-3034-8. – Текст : электронный. – URL: <https://znanium.com/catalog/product/1936353>.

16. Трусов, А. В. Технология проектирования информационных систем : учебное пособие / А. В. Трусов, В. А. Трусов. – Москва ; Вологда : Инфра-Инженерия, 2023. – 244 с. – ISBN 978-5-9729-1340-4. – Текст : электронный. – URL: <https://znanium.com/catalog/product/2100456>.

17. Хамбл, Джек, Фарли, Дейвид. Непрерывное развертывание ПО: автоматизация процессов сборки, тестирования и внедрения новых версий программ. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2011. – 432 с. : ил.; [Электронный ресурс]: Режим доступа: https://k0d.cc/storage/books/Разное/nepreryvnoe_razvertyvanie_po_avtomatizatsiya_protseessov_sborki.pdf.

18. Черушева, Т. В. Проектирование программного обеспечения : учеб. пособие / Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 172 с.