**Оглавление**

[1. Архитектура вычислительной среды, компоненты, реляционное ядро СУБД SQL Server 2000. 3](#_Toc529450420)

[2.Серверные компоненты СУБД SQL Server 2000. 10](#_Toc529450421)

[3.Клиентские средства администрирования СУБД SQL Server 2000. 20](#_Toc529450422)

[4.Протоколирование транзакций. 28](#_Toc529450423)

[5. Восстановление пользовательских баз данных. 31](#_Toc529450424)

[6. Восстановление системных баз данных. 34](#_Toc529450425)

[7. Управление учетными записями пользователей. 36](#_Toc529450426)

[8. Роли пользователей. Администрирование сервером SQL Server 2000. 38](#_Toc529450427)

[9. Настройка дополнительных компонентов (служб электронной почты, настройка связанных серверов). 46](#_Toc529450428)

[10. Мониторинг работы SQL Server 2000. 50](#_Toc529450429)

[11. Администрирование систем типа Oracle. 53](#_Toc529450430)

[12. Распределенные приложения в сети Интернет. 81](#_Toc529450431)

[13. Сетевые протоколы прикладного уровня: ftp, smtp, http, telnet, gopher, nntp, pop3, imap. 90](#_Toc529450432)

[14. Служба WWW. 92](#_Toc529450433)

[15. Использование серверных расширений для управления содержимым (контентом) сайта. 94](#_Toc529450434)

[16. Механизмы ADO.NET. 112](#_Toc529450435)

[17. Безопасность Интернет-сервисов. 114](#_Toc529450436)

[18. Аутентификация пользователей. Веб-сервер Apache. 117](#_Toc529450437)

[19. CGI-программирование. 124](#_Toc529450438)

[20. Язык серверных сценариев PHP. 126](#_Toc529450439)

[21. Электронная почта. 127](#_Toc529450440)

[22. Почтовые клиенты. 129](#_Toc529450441)

[23. Управление протоколами обмена почтовыми сообщениями. 131](#_Toc529450442)

[24. Управлением клиентами почтового сервера. 137](#_Toc529450443)

[25. Администрирование пользователей и контактов. 139](#_Toc529450444)

[26. Безопасность почтовых сообщений. 142](#_Toc529450445)

[27. Аутентификация пользователей, интеграция с Active Directory. 143](#_Toc529450446)

[28. Борьба с вирусами. 150](#_Toc529450447)

[29. Основные задачи сетевого администрирования. 156](#_Toc529450448)

[30. Операционная система MS Windows NT/2000/XP/2003. 159](#_Toc529450449)

[31. Сетевые протоколы NetBIOS, TCP/IP. 163](#_Toc529450450)

[32. Совместный доступ к файлам и принтерам в Windows. 167](#_Toc529450451)

[33. Команды управления сетевыми службами в Windows. 168](#_Toc529450452)

[34. Набор команд управления сетевыми службами Net. 169](#_Toc529450453)

[35. Мониторинг работы системы. 170](#_Toc529450454)

[36. Веб-интерфейс для организации удаленного управления компьютером под управлением Windows 2000/2003. 172](#_Toc529450455)

[37.Маршрутизаторы в сетях TCP/IP. 175](#_Toc529450456)

# 1. Архитектура вычислительной среды, компоненты, реляционное ядро СУБД SQL Server 2000.

Microsoft SQL Server 2000 — это законченное решение для управления и анализа данных, позволяющее оперативно развертывать масштабируемые Web-приложения нового поколения. SQL Server 2000 — ключевой компонент поддержки электронной коммерции, интерактивных деловых приложений и хранилищ данных, обеспечивающий масштабируемость, необходимую для поддержки растущих, динамических сред. В SQL Server 2000 предусмотрена широчайшая поддержка XML (Extensible Markup Language) и других форматов, используемых в Интернете, функций производительности и доступности, гарантирующих своевременное решение поставленных задач, а также развитой функциональности управления и настройки, позволяющей автоматизировать выполнение рутинных задач и снизить совокупную стоимость владения. Кроме того, SQL Server 2000 в полном объеме использует преимущества Windows 2000, обеспечивая интеграцию с Active Directory Services и поддерживая до 32 процессоров и до 64 гигабайт (Гб) оперативной памяти.

SQL Server 2000 — это реляционная СУБД, которая использует язык Transact SQL для пересылки сообщений между компьютером клиента и компьютером, на котором работает SQL Server 2000. Реляционная СУБД состоит из механизма баз данных, собственно баз данных и приложений, необходимых для управления данными и компонентами реляционной СУБД. Реляционная СУБД организует данные в виде связанных строк и столбцов, составляющих базу данных. Реляционная СУБД отвечает за поддержку структуры базы данных и решает следующие задачи:

поддерживает связи между данными в базе;

гарантирует корректное хранение данных и выполнение правил, регламентирующих связи между ними;

восстанавливает данные после аварии системы, переводя их в согласованное состояние, зафиксированное до сбоя.

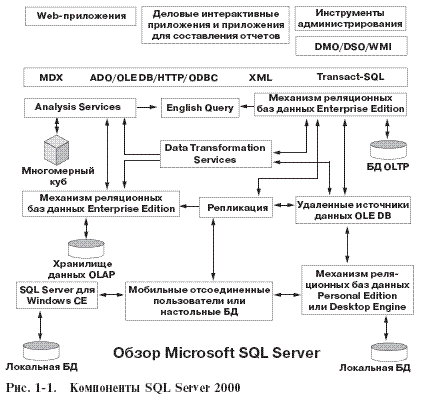
База данных (БД) SQL Server 2000 представляет собой реляционную базу данных, совместимую с SQL (Structured Query Language) с интегрированной поддержкой XML для Интернет приложений. SQL Server 2000 создан на основе современного расширяемого сервера SQL Server 7.0. В следующих разделах вы познакомитесь с базами данных, реляционными базами данных, SQL и XML.

Обзор компонентов SQL Server 2000

SQL Server 2000 состоит из ряда компонентов, таких, как механизм реляционных баз данных, Analysis Services и English Query. Все эти компоненты, каждый из которых играет определенную роль, работая совместно, формируют полнофункциональную реляционную СУБД

Механизм реляционных баз данных

Механизм реляционных баз данных SQL Server 2000 — это современное ядро с высокой степенью масштабируемости, предназначенное для хранения данных. Механизм баз данных сохраняет данные в таблицах. Каждая таблица представляет определенный класс объектов, в зависимости от интересов конкретной организации (например, сведения о транспортных средствах, работниках или клиентах). Таблица состоит из столбцов, каждый из которых представляет атрибут объекта (например, вес, имя или стоимость), который она моделирует, и строк. яКаждая строка представляет один экземпляр объекта, моделируемого таблицей (например, автомобиль с номером ABC-123 или работник с идентификационным номером ID 123456). Приложение передает механизму баз данных оператор SQL, механизм возвращает результат в виде набора данных в табличной форме. Интернет-приложение передает механизму баз данных оператор SQL или запрос XPath, а тот возвращает результат в виде документа XML. Механизм реляционных баз данных обеспечивает поддержку стандартных интерфейсов доступа к данным, таких, как ADO (ActiveX Data Objects), OLE DB и ODBC (Open Database Connectivity).  
Механизм реляционных баз данных обладает высокой масштабируемостью. SQL Server 2000 Enterprise Edition поддерживает группы серверов баз данных, формирующих базы данных терабайтного объема, к которым могут обращаться тысячи пользователей одновременно. Механизм баз данных также способен динамически настраиваться путем выделения дополнительных ресурсов по мере роста числа пользователей, подключенных к базе данных, и освобождения их после отключения пользователей. Другими словами, отдельные пользователи или небольшие рабочие группы, у которых нет администраторов баз данных, могут использовать более простые редакции SQL Server. С помощью административных утилит с графическим интерфейсом из комплекта поставки продукта легко администрировать даже крупные серверы баз данных под управлением Enterprise Edition, работающие в эксплуатационном режиме.



Механизм реляционных баз данных также обладает высокой степенью защиты. Аутентификацию при регистрации допустимо интегрировать с проверкой подлинности Windows, поэтому SQL Server не хранит никаких паролей и не пересылает их по сети (где их могут прочесть, используя анализаторы пакетов). На узлах разрешается задавать аудит всех пользователей, обращающихся к базе данных, соответствующий требованиям безопасности уровня С2, и применять протокол SSL (Secure Sockets Layer) для шифрования всех данных, передаваемых между приложением и базой данных.

Репликация

Репликация SQL Server 2000 позволяет поддерживать несколько копий данных на различных компьютерах с целью повышения общей производительности системы, а также обеспечивает поддержку синхронизации всех копий. Например, какой-либо отдел размещает данные о продажах на собственном сервере и посредством репликации обновляет эти сведения на корпоративном компьютере. Или другой пример: несколько мобильных пользователей работают весь день, отключившись от сети, а в конце рабочего дня они с помощью репликации сведением выполняют слияние своих записей с главной базой данных.  
Репликация — важная и мощная технология распределения данных и некоторых типов объектов баз данных (хранимых процедур, представлений и пользовательских функций) по всему предприятию. В репликации SQL Server используется принцип «публикации и подписки». Издатель (владелец) данных, подлежащих репликации, определяет статьи (аналогичные таблицам базы данных), которые надо сделать доступными для подписчиков (или для адресов, получающих копии оригинальной публикации).  
Модель репликации сведением позволяет многим или даже всем подписчикам, а также издателю обновлять реплицируемые данные. Репликация сведением является идеальным решением для любого окружения, где подписчики часто отсоединены от сети (например, для удаленных продавцов). Подписчики получают возможность модифицировать или добавлять данные в дороге, а по возвращении в офис или при подключении к локальной сети через модем — выполнять слияние модифицированных данных с оригинальной копией (опубликованной издателем).

Analysis Services

Analysis Services предоставляет инструменты для анализа данных, которые находятся в хранилищах и киосках данных. В хранилищах и киосках данных итоговая информация содержится в таблицах фактов. Таблица фактов — центральная таблица в схеме хранилища данных, в ней хранятся численные меры и ключи, связывающие факты с таблицами измерений. Как правило, базовая таблица фактов содержит сведения, описывающие некоторые события в бизнесе, например банковские транзакции или факты продажи продукции. Приложения работают с данными Analysis Services с помощью многомерных расширений ADO и OLE DB. Обработка запросов OLAP посредством многомерных кубов Analysis Services выполняется существенно быстрее, чем с использованием подробной информации из баз данных OLTP.

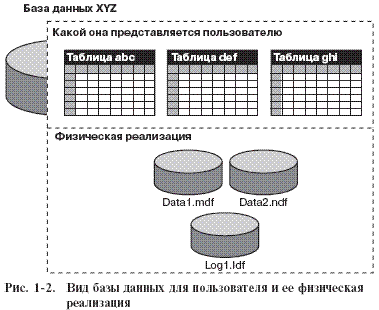
В систему Analysis Services входит сервер, управляющий многомерными кубами, предназначенными для анализа. Он обеспечивает клиенту быстрый доступ к данным куба. Чтобы быстро выдавать ответы на сложные аналитические запросы, Analysis Services организует данные из хранилища в кубические массивы с помощью предварительно вычисленных агрегированных данных. Analysis Services также облегчает создание моделей извлечения информации для данных как из многомерных, так и из реляционных источников. Можно применять модели извлечения информации к обоим типам данных. Посредством службы PivotTable — компонента доступа, совместимого с OLE DB, Microsoft Excel и приложения других производителей могут получать данные с сервера и представлять их пользователю или создавать локальные кубические массивы для автономного анализа.

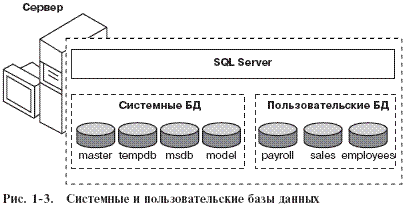
English Query

SQL Server 2000 English Query помогает создавать приложения, способные автоматически настраиваться в соответствии со специальными вопросами, которые задают пользователи. Администратор English Query определяет для обработчика English Query все логические связи между таблицами и столбцами базы данных или между кубическими массивами в хранилище или киоске данных. Затем пользовательское приложение может вывести специальное окно, в котором пользователю достаточно набрать символьную строку с вопросом (записанным по-английски), касающимся данных в базе или хранилище. Приложение передает эту строку обработчику English Query, который анализирует ее с учетом связей, определенных между таблицами и кубами. После этого English Query возвращает приложению оператор SQL или запрос MDX (Multi-Dimensional Expression), при исполнении которого будет получен ответ на заданный пользователем вопрос. Посредством English Query разработчики могут преобразовывать реляционные базы данных в приложения English Query, которые позволяют конечным пользователям вместо формирования запроса с помощью оператора SQL задавать вопросы по-английски. Редактор моделей English Query находится в среде разработки Microsoft Visual Studio 6.0. При применении Visual Studio информация о базе данных задается таким образом, что English Query без труда обрабатывает вопросы относительно конкретных таблиц, полей и данных, введенные по-английски. В English Query входят мастера, автоматически создающие проект (.eqp-файл) или модель (.eqm-файл) для приложения. После проверки проект компилируют в приложение English Query (.eqd-файл), после чего можно осуществлять его развертывание. Мастер SQL Project ускоряет процесс создания проектов English Query путем автоматического создания объектов и связей для выбранных баз данных. Перед развертыванием приложения English Query следует выполнить его полное тестирование, чтобы убедиться, что все существующие объекты и связи возвращают точные результаты. Окно Model Test поддерживает встроенные отладочные средства, позволяющие вносить дальнейшие изменения и улучшения в проект EQ.

Архитектура базы данных

В SQL Server 2000 информация хранится в базах данных. Она организована в доступные пользователю логические компоненты, а сама база данных физически реализована в виде двух или более файлов на диске. Обращаясь к базе данных, вы главным образом имеете дело с логическими компонентами (таблицами, представлениями, процедурами и учетными именами). Физическая реализация файлов во многом прозрачна. Как правило, лишь администратор базы данных работает с ее физической реализацией..  
У каждого экземпляра SQL Server есть четыре системных базы данных (master, tempdb, msdb и model) и одна или несколько пользовательских. В одних организациях все данные содержатся в единственной пользовательской базе данных, а в других для каждой группы создана собственная база данных. Также у каждой базы данных может быть свое приложение, использующее ее. Например, в организации иногда отдельная база данных предназначена для учета продаж, другая — для платежных ведомостей, третья — для работы приложения управления делопроизводством и т. д. Некоторые приложения используют только одну базу данных, а другие могут обращаться к нескольким. На рис. показаны системные и несколько пользовательских баз данных SQL Server.





Нет необходимости запускать несколько копий механизма баз данных SQL Server, чтобы предоставить доступ к базе данных на сервере нескольким пользователям. Единственный экземпляр SQL Server Standard Edition или Enterprise Edition способен обрабатывать запросы тысяч пользователей, одновременно работающих с разными базами данных. Каждый экземпляр SQL Server делает все свои базы данных доступными всем, кто подключается к нему (в зависимости от определенных для них прав доступа).

При подключении к экземпляру SQL Server соединение ассоциируется с определенной базой данных на сервере. Эта БД называется текущей. Обычно соединение устанавливается с базой данных по умолчанию, которую определяет системный администратор. Но, настроив параметры соединения из API баз данных, можно задать и другую БД. Можно переключаться с одной базы данных на другую с помощью оператора Transact SQL USE <имя\_БД> или функции API, которая меняет текущий контекст базы данных.

SQL Server 2000 позволяет отключить базу данных от одного экземпляра SQL Server, а затем подключить ее к другому экземпляру или вернуть обратно. При наличии файла с базой данных можно дать SQL Server указание подключать этот файл при установлении соединения под определенным именем.

# 2.Серверные компоненты СУБД SQL Server 2000.

Ниже представлен перечень серверных компонентов СУБД SQL Server 2000:

DTS

Многим организациям для более эффективного принятия решений требуется централизация данных. Однако данные можно хранить в самых разнообразных форматах и в нескольких различных местах. DTS (Data Transformation Services) в SQL Server позволяет создавать хранилища и киоски данных путем интерактивного или автоматического импорта и передачи данных из нескольких гетерогенных источников по расписанию.

DTS SQL Server 2000 существенно повышает эффективность процесса создания хранилищ данных для оперативной аналитической обработки (Online Analytical Processing, OLAP). Кроме того, он предоставляет средства для тонкой настройки обширных баз данных для оперативной обработки транзакций (Online Transaction Processing, OLTP), в результате чего можно увеличить число одновременно работающих пользователей, активно добавляющих и модифицирующих данные. Структура баз данных OLTP такова, что они регистрируют подробности каждой транзакции. Попытка выполнить сложный анализ для определения трендов продаж за несколько месяцев или лет потребует просмотра огромного числа записей, а большая загруженность обработкой информации при этом снижает производительность баз данных OLTP.

Хранилища и киоски данных создаются в системе OLTP на основе данных, извлеченных и преобразованных в форму, которая лучше подходит для OLAP-обработки. Периодически осуществляется сбор строк с подробными данными OLTP в промежуточную базу данных, где они обобщаются, а итоговые данные помещаются в хранилище или киоск. DTS поддерживает извлечение данных из одного источника и выполнение сложных преобразований с последующим сохранением итоговых преобразованных данных в другом источнике данных. Этот компонент в значительной степени упрощает процесс извлечения данных из нескольких систем OLTP и создания на основе извлеченных данных хранилища или киоска данных для OLAP.

Analysis Services

Analysis Services предоставляет инструменты для анализа данных, которые находятся в хранилищах и киосках данных. В хранилищах и киосках данных итоговая информация содержится в таблицах фактов. Таблица фактов — центральная таблица в схеме хранилища данных, в ней хранятся численные меры и ключи, связывающие факты с таблицами измерений. Как правило, базовая таблица фактов содержит сведения, описывающие некоторые события в бизнесе, например банковские транзакции или факты продажи продукции. Приложения работают с данными Analysis Services с помощью многомерных расширений ADO и OLE DB. Обработка запросов OLAP посредством многомерных кубов Analysis Services выполняется существенно быстрее, чем с использованием подробной информации из баз данных OLTP.  
В систему Analysis Services входит сервер, управляющий многомерными кубами, предназначенными для анализа. Он обеспечивает клиенту быстрый доступ к данным куба. Чтобы быстро выдавать ответы на сложные аналитические запросы, Analysis Services организует данные из хранилища в кубические массивы с помощью предварительно вычисленных агрегированных данных. Analysis Services также облегчает создание моделей извлечения информации для данных как из многомерных, так и из реляционных источников. Можно применять модели извлечения информации к обоим типам данных. Посредством службы PivotTable — компонента доступа, совместимого с OLE DB, Microsoft Excel и приложения других производителей могут получать данные с сервера и представлять их пользователю или создавать локальные кубические массивы для автономного анализа.

English Query

SQL Server 2000 English Query помогает создавать приложения, способные автоматически настраиваться в соответствии со специальными вопросами, которые задают пользователи. Администратор English Query определяет для обработчика English Query все логические связи между таблицами и столбцами базы данных или между кубическими массивами в хранилище или киоске данных. Затем пользовательское приложение может вывести специальное окно, в котором пользователю достаточно набрать символьную строку с вопросом (записанным по-английски), касающимся данных в базе или хранилище. Приложение передает эту строку обработчику English Query, который анализирует ее с учетом связей, определенных между таблицами и кубами. После этого English Query возвращает приложению оператор SQL или запрос MDX (Multi-Dimensional Expression), при исполнении которого будет получен ответ на заданный пользователем вопрос. Посредством English Query разработчики могут преобразовывать реляционные базы данных в приложения English Query, которые позволяют конечным пользователям вместо формирования запроса с помощью оператора SQL задавать вопросы по-английски. Редактор моделей English Query находится в среде разработки Microsoft Visual Studio 6.0. При применении Visual Studio информация о базе данных задается таким образом, что English Query без труда обрабатывает вопросы относительно конкретных таблиц, полей и данных, введенные по-английски. В English Query входят мастера, автоматически создающие проект (.eqp-файл) или модель (.eqm-файл) для приложения. После проверки проект компилируют в приложение English Query (.eqd-файл), после чего можно осуществлять его развертывание. Мастер SQL Project ускоряет процесс создания проектов English Query путем автоматического создания объектов и связей для выбранных баз данных. Перед развертыванием приложения English Query следует выполнить его полное тестирование, чтобы убедиться, что все существующие объекты и связи возвращают точные результаты. Окно Model Test поддерживает встроенные отладочные средства, позволяющие вносить дальнейшие изменения и улучшения в проект EQ.

MetaData Services

Службы Meta Data Services из SQL Server обеспечивают хранение и управление метаданными информационных систем и приложений. Эта технология выполняет функции концентратора определений данных и компонентов, моделей разработки и развертывания, программных компонентов, предназначенных для повторного использования, и описаний хранилищ данных. Этот продукт состоит из следующих компонентов: ядра репозитория, инструментов, API, стандартных информационных моделей, браузера и SDK (Software DevelopmentKit).  
Meta Data Services представляет собой набор служб для управления метаданными. Для работы с Microsoft Meta Data Services необходимо разбираться в характеристиках метаданных. Метаданные описывают структуру и значение данных, а также структуру и значение приложений и процессов. Следует помнить, что метаданные — это абстракция, что у них есть контекст и они способны выполнять различные функции в среде разработки.  
При разработке ПО структуры приложений и баз данных, представляющих данные или хранящих их, можно абстрагировать в схемы классификации метаданных, имеющие смысл для разработчиков и программистов. Таблица или форма являются производными некоторого объекта, который, в свою очередь, может быть производным класса.  
Для метаданных существует несколько уровней абстрагирования. Можно описать экземпляр данных, затем — это описание и так далее — последовательные описания описаний, пока не будет достигнут некий практический предел. Как правило, для описания метаданных, используемого в разработке ПО, имеется три уровня абстрагирования. Например, экземпляр данных «таблица займов» можно представить в виде таблицы базы данных, ту, в свою очередь — как объект «таблица», принадлежащий базе данных, и наконец, объект базы данных «таблица» — как абстрактный класс, формализующий фиксированный набор характеристик, которым должны соответствовать все производные объекты.  
Отличие между данными и метаданными часто приравнивается к отличию между типом и экземпляром. Разработчик модели определяет тип (например, класс или связь), а разработчик ПО определяет экземпляр (например, класс «таблица» или связь «таблица — столбцы»).  
Различие между экземпляром и типом зависит от контекста. То, что в одном сценарии является метаданными, в другом сценарии может стать данными. Например, в типичной реляционной СУБД системный каталог описывает таблицы и столбцы, содержащие данные.  
Можно рассматривать данные системного каталога как метаданные, поскольку они представляют собой описания определений данных. Однако, используя соответствующий программный инструмент, удается манипулировать ими, как и любыми другими данными. Так, при просмотре связей между данными, сведений о версии таблицы или идентификации всех таблиц с финансовой информацией путем поиска столбцов с типом данных «валюта» типичные метаданные (представителем которых является системный каталог) играют роль данных, которыми можно манипулировать.

Books Online

SQL Server Books Online — это электронная документация, которая поставляется с SQL Server 2000. Чтобы найти нужную информацию в SQL Server Books Online, достаточно:

просмотреть оглавление;

выполнить поиск в указателе по ключевому слову;

выполнить поиск по слову или целой фразе.

В состав SQL Server Books Online также входят справочные файлы, которые открываются при щелчке кнопки Help в диалоговом окне или при нажатии клавиши F1.  
Документацию SQL Server Books Online нельзя открыть в полном объеме с компакт-диска SQL Server 2000 — прежде необходимо установить ее на локальный жесткий диск. Раздел вводного руководства по Microsoft SQL Server 2000, посвященный установке, не включает сведения о некоторых изменениях, произошедших после того, как руководство уже было отпечатано. Они внесены в SQL Server Books Online в справочном файле программы установки SQL Server.  
Чтобы установить на жесткий диск только документацию SQL Server Books Online, надо выполнить выборочную установку компонентов SQL Server 2000 и выделить только компонент Books Online. Он представляет собой набор справочных файлов в формате HTML, для просмотра которых необходим Microsoft Internet Explorer версии 5.0 или более поздней.

Инструменты SQL Server 2000

В состав SQL Server 2000 входит множество утилит как с графическим интерфейсом, так и работающих в командной строке. Они предназначены для пользователей, программистов и администраторов и позволяют решать широкий круг задач, в том числе:

администрировать и конфигурировать SQL Server;

определять информацию каталога для копии SQL Server;

конструировать и тестировать запросы;

копировать, импортировать, экспортировать и преобразовывать данные;

выводить диагностическую информацию;

запускать и останавливать SQL Server.

Инструменты командной строки

Утилиты командной строки автоматически устанавливаются во время установки SQL Server 2000 на компьютер под управлением Windows 2000, Windows NT, Windows 95 или Windows 98. В следующей таблице перечислены утилиты командной строки SQL Server и указаны папки, в которые они устанавливаются по умолчанию. Обратите внимание, что копия SQL Server 2000 по умолчанию устанавливается в подкаталог с именем MSSQL. Каждый именованный экземпляр SQL Server 2000 будет установлен в соответствующую папку с именем MSSQL$<имя\_экземпляра>.

|  |  |
| --- | --- |
| Утилиты | Папка |
| console |  |
| sqlagent |  |
| sqldiag |  |
| sqlmaint |  |
| sqlservr |  |
| vswitch | x:\Program Files\Microsoft SQL Server\MSSQL\Binn |
| bcp |  |
| dtsrun |  |
| dtswiz |  |
| isql |  |
| isqlw |  |
| itwiz |  |
| odbccmpt |  |
| osql |  |
| rebuildm |  |
| sqlftwiz | x:\Program Files\Microsoft SQL Server\80\Tools\Binn |
| distrib |  |
| logread |  |
| replmerg |  |
| snapshot | x:\Program Files\Microsoft SQL Server\80\Com |

Во время установки путь к папке x:\Program Files\Microsoft SQL Server\80\Tools\Binn добавляется к системному пути. Утилиты, расположенные в этой папке, можно запускать из командной строки, находясь в любом месте дерева каталогов. Утилиту, расположенную вне этой папки, необходимо запускать либо из того каталога, в котором она установлена, либо добавить путь к этому каталогу к пути поиска ОС.  
Три указанные ниже утилиты больше не устанавливаются программой установки SQL Server 2000:

makepipe;

odbcping;

readpipe.

При необходимости их можно запустить из папки \x86\Binn на установочном компакт-диске SQL Server 2000 или вручную скопировать на компьютер. Описание любой утилиты командной строки из SQL Server 2000 вы найдете в SQL Server Books Online.

Инструменты с графическим интерфейсом

В состав SQL Server 2000 входит богатый набор инструментов с графическим интерфейсом, предназначенных для работы с SQL Server и его администрирования. В этом разделе кратко описаны многие из них. Кроме того, информация обо всех инструментах с графическим интерфейсом есть в Books Online.

Enterprise Manager

SQL Server Enterprise Manager — основной инструмент администрирования SQL Server 2000, поддерживающий пользовательский интерфейс, совместимый с MMC (Microsoft Management Console) и позволяющий решать ряд административных задач:

определять группы серверов, работающих под управлением SQL Server;

регистрировать отдельные серверы в группе;

настраивать любые параметры SQL Server для всех зарегистрированных серверов;

оздавать и администрировать любые базы данных, объекты, идентификаторы пользователей, учетные имена и права доступа к SQL Server на каждом из зарегистрированных серверов;

определять и исполнять все административные задачи SQL Server на каждом зарегистрированном сервере;

интерактивно конструировать и тестировать операторы SQL, пакеты и сценарии, вызывая SQL Query Analyzer;

вызывать различные мастера SQL Server.

MMC поддерживает общий интерфейс для управления различными серверными приложениями в сети Microsoft Windows. В состав серверных приложений входит такой компонент, как оснастка, который предоставляет пользователям MMC интерфейс для управления серверным приложением. SQL Server Enterprise Manager является оснасткой MMC для Microsoft SQL Server 2000.

SQL Server Agent

SQL Server Agent работает на сервере, который функционирует под управлением экземпляра SQL Server 2000 или более ранних версий SQL Server. SQL Server Agent отвечает за решение следующих задач:

запуск заданий SQL Server, запланированных для исполнения в определенное время или по истечении определенного периода времени;

определение особых условий, при наступлении которых необходимо выполнить заданное администратором действие, например предупредить кого-нибудь, отправив сообщение на пейджер или по электронной почте, или запустить задачу, соответствующую этим условиям;

запуск определенных администраторами задач, выполняющих репликацию.

SQL Profiler

SQL Profiler — это инструмент для записи событий SQL Server 2000. События сохраняются в файле трассировки, который впоследствииможно проанализировать или использовать для повтора некоторой последовательности действий при диагностировании возникшей проблемы. SQL Profiler применяется для:

пошагового исполнения проблемных запросов и определения источника проблемы;

поиска и диагностики медленных запросов;

записи последовательностей SQL-операторов, приводящих к возникновению проблем;

мониторинга производительности SQL Server и регулирования его загруженности.

SQL Profiler также поддерживает аудит действий, выполненных с экземплярами SQL Server. Информация о действиях, имеющих отношение к безопасности, сохраняется для последующего просмотра администратором, отвечающим за безопасность.

Утилита Client Network

Client Network используется для управления клиентскими библиотеками Net-Libraries и определения псевдонимов серверов. Кроме того, эта утилита позволяет устанавливать параметры по умолчанию, которые используются приложениями DB-Library.  
Большинству пользователей утилита Client Network никогда не понадобится. Для подключения к SQL Server 2000 им достаточно указать сетевое имя сервера, на котором работает SQL Server, и (что не обязательно) имя экземпляра SQL Server.

Утилита Server Network

Утилита Server Network применяется для управления серверными библиотеками Net-Libraries, а также позволяет задавать:

стеки сетевых протоколов, используемые экземпляром SQL Server 2000 для прослушивания клиентских запросов;

последовательность, в которой серверные библиотеки Net-Libraries определяют, не устанавливает ли приложение соединение;

новые сетевые адреса для прослушивания запросов экземпляром SQL Server 2000.

Большинству администраторов утилита Server Network также никогда не понадобится. Они могут задать серверные библиотеки Net-Libraries во время установки сервера.

Service Manager

SQL Server Service Manager предназначен для запуска, останова и приостановки серверных компонентов SQL Server 2000. Эти компоненты работают как службы в Microsoft Windows NT или Windows 2000, а в Windows 95 и Windows 98 — как отдельные исполняемые программы.

SQL Server. Реализует механизм баз данных SQL Server. Для каждого экземпляра SQL Server, работающего на компьютере, существует по одной службе SQL Server.

SQL Server Agent. Реализует агент, который запускает запланированные административные задачи SQL Server. Для каждого экземпляра SQL Server, работающего на компьютере, имеется по одной службе SQL Server Agent.

Microsoft Search (только для Windows NT и Windows 2000). Реализует механизм полнотекстового поиска. Существует в единственном экземпляре, независимо от числа экземпляров SQL Server на компьютере.

MSDTC (только для Windows NT и Windows 2000). Управляет распределенными транзакциями. Существует в единственном экземпляре, независимо от числа экземпляров SQL Server на компьютере.

MSSQLServerOLAPService (только для Windows NT и Windows 2000). Реализует Analysis Services. Существует в единственном экземпляре, независимо от числа экземпляров SQL Server на компьютере.

Окно Service Manager может быть скрыто и представлено значком в системной области панели задач. Чтобы вывести меню со списком задач, которые поддерживает Service Manager, щелкните правой кнопкой значок на панели задач.

SQL Query Analyzer

SQL Query Analyzer — это инструмент с графическим интерфейсом, предназначенный для решения множества различных задач:

создания запросов и сценариев SQL, а также исполнения их с базами данных SQL Server;

создания часто используемых объектов баз данных в стандартных сценариях;

копирования существующих объектов баз данных;

исполнения хранимых процедур без задания их параметров;

отладки хранимых процедур;

отладки запросов, имеющих проблемы с производительностью;

поиска объектов в базах данных, а также просмотра и работы с объектами;

добавления, обновления и удаления строк в таблице;

определения комбинаций клавиш для запуска часто используемых запросов;

добавления часто используемых команд в меню Tools.

SQL Query Analyzer запускают непосредственно из меню Start или в SQL Server Enterprise Manager. Его также можно запустить, введя в командной строке команду isqlw.

Встроенные мастера SQL Server 2000

В состав SQL Server 2000 входит несколько мастеров, помогающих администраторам и программистам решать сложные административные задачи, а также всем пользователям просматривать и модифицировать информацию в базах данных SQL Server. Подробное описание этих мастеров хранится в SQL Server Books Online.

# 3.Клиентские средства администрирования СУБД SQL Server 2000.

Обычно инструменты администрирования устанавливаются при инсталляции самого SQL Server 2000. Тем не менее они могут быть добавлены и отдельно. Таким образом, на одном компьютере могут иметься только инструменты администрирования, тогда как на другом – только собственно SQL Server 2000 (так называемый Engine). Инструменты SQL Server 2000 спроектированы таким образом, что они могут применяться для работы с любым сервером SQL Server 2000 локальной сети. Более того, инструменты администрирования SQL Server 2000 могут быть использованы для управления и серверами SQL Server 7.0. Администрирование же серверов SQL Server 6.х должно выполняться инструментами администрирования, поставляемыми в составе этих версий.

Большая часть административных задач SQL Server 2000 может быть выполнена следующими методами:

· использованием средств Transact-SQL;

· с помощью графического интерфейса Enterprise Manager;

· утилиты командной строки;

· с помощью мастеров (wizards).

Самым сложным является выполнение задачи средствами Transact-SQL, так как это требует знания синтаксиса команд и хранимых процедур, а также умения обращаться с инструментом Query Analyzer (или любым другим ему подобным). Однако использование средств Transact-SQL открывает пользователю прямой доступ к системным данным.

Enterprise Manager реализован в виде MMC-модуля, на основе которого генерируется консоль администрирования SQL Server. Microsoft Management Console (MMC) является новой разработкой фирмы Microsoft и реализует единый пользовательский интерфейс, обеспечивая интегрированную среду для администрирования сетевых ресурсов. Управление практически всеми новыми продуктами осуществляется с использованием единого интерфейса MMC. Каждое приложение имеет специальный модуль, содержащий всю необходимую информацию о параметрах настройки и администрирования. MMC использует этот модуль для построения интерфейса администрирования, привязанного к конкретному приложению.

Консоль MMC включает, как минимум, одно окно, состоящее из двух панелей. Левая панель, называемая обзорной, содержит пространство имен. Пространство имен отображается в виде дерева узлов, являющихся либо объектами, либо контейнерами. Когда выбирается узел в пространстве имен, то в правой панели, называемой результирующей, отражается содержимое узла. Если выбран объект, то будут выведены характеристики этого объекта. Если же выбран контейнер, то в результирующей панели будут отображены все объекты и контейнеры, которые он содержит. Список всех возможных действий над объектом представлен в его контекстном меню, вызываемом нажатием правой клавиши мыши.

Важной особенностью MMC является возможность сохранить настроенную консоль в файле с расширением .msc. Этот файл имеет небольшие размеры и может быть распространен по сети или электронной почте.

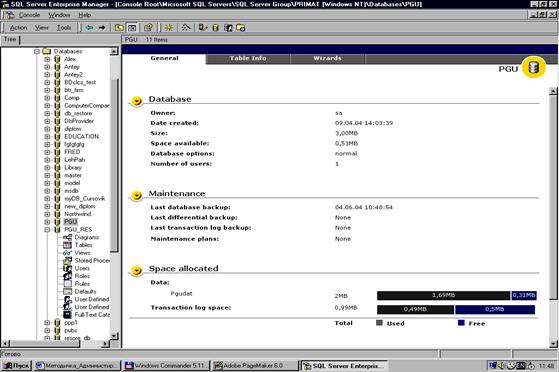


Рисунок. Окно Enterprise Manager.

MMC облегчает администрирование корпоративных сетей. MMC позволяет администратору создавать задачно-ориентированные консоли, которые содержат только утилиты и инструменты, необходимые для решения конкретной задачи. Главный системный администратор предприятия может создавать консоли для административных групп структурных подразделений, выполняющих узкоспециализированные задачи, например, резервное копирование баз данных или управление почтовым сервером.

Enterprise Manager является базовым инструментом при выполнении самых разнообразных задач:

· управления системой безопасностью;

· создания баз данных и их объектов;

· создания и восстановление резервных копий;

· конфигурирования подсистемы репликации;

· управления параметрами работы служб SQL Server 2000;

· управления подсистемой автоматизации;

· запуска, останова и приостанова служб;

· конфигурирования связанных и удаленных серверов;

· создания, управления и выполнения пакетов DTS

Приведенный список не исчерпывает всех областей применения Enterprise Manager и легко может быть расширен.

Enterprise Manager является достаточно простым в использовании инструментом, в то же время он охватывает практически все административные задачи, с которыми столкнется администратор. Конечно, различные нестандартные ситуации с помощью этого инструмента не решить и придется обратиться к средствам Transact-SQL. Не стоит воспринимать Enterprise Manager как инструмент для неопытных пользователей, не умеющих работать со средствами Transact-SQL, и стремиться осилить синтаксис команд и хранимых процедур. Решение некоторых задач средствами Transact-SQL настолько сложно, что это поможет сэкономить достаточно много времени и использовать его более эффективно.

Панель инструментов Enterprise Manager содержит меню Action (Действие), View (вид) и Tools (сервис). Конфигурация меню и перечень доступных команд зависят от того, какой объект выбран в настоящий момент. Меню Action (Действие) содержит тот же набор команд, что и контекстное меню объекта. Меню View (вид) позволяет изменить способ представления информации в результирующей панели Enterprise Manager. С помощью меню Tools (сервис) становится доступным дополнительный инструментарий. Здесь можно вызвать как утилиты SQL Server (такие как Profiler и Query Analyzer), так и внешние по отношению к SQL Server приложения. Пример окна Enterprise Manager приведен на рисунке 7.

SQL Server Service Manager. Единственной задачей утилиты SQL Server Service Manager является предоставление пользователю удобного механизма запуска, останова и приостановки служб SQL Server 2000. Кроме этого, она позволяет лишь запретить или разрешить автоматический запуск той или иной службы при загрузке операционной системы.



Рисунок. Пример окна утилиты SQL Server Service Manager.

Утилита Service Manager устанавливается при инсталляции SQL Server 2000 и по умолчанию автоматически запускается при загрузке операционной системы. В нормальном состоянии утилита Service Manager представлена значком в правой части панели задач (taskbar). Двойной щелчок на пиктограмме приведет к открытию окна программы, с помощью которого можно запускать, останавливать и приостанавливать службы SQL Server 2000, а также разрешать или запрещатьихавтоматический запуск при загрузке операционной системы. Пример окна утилиты SQL Server Service Manager приведен на рисунке 8.

Помимо уже рассмотренных утилит, имеющих графический интерфейс, в SQL Server 2000 существует набор утилит командной строки., с помощью которых также можно выполнять различные задачи. Некоторые из этих утилит используются сервером автоматически и являются, скорее, частью ядра SQL Server 2000, чем собственно утилитами. Указанные утилиты автоматически копируются мастером установки в каталог Binn установочного каталога SQL Server 2000, но могут быть запущены из любого другого каталога, так как мастер соответствующим образом конфигурирует переменную окружения PATH. В таблице 5 приведен список утилит командной строки, имеющихся в распоряжении пользователя при работе с SQL Server 2000. Необходимо отметить, что параметры утилит командной строки чувствительны к регистру вводимых значений. В некоторых случаях параметры, набранные в разных регистрах, могут привести к выполнению различных действий.

Таблица. Утилиты командной строки

|  |  |
| --- | --- |
| Утилита | Описание |
| bcp.exe | Программа копирования данных из таблиц и представлений базы в текстовые файлы и обратно (Bulk Copy Program API) |
| console.exe | Программа для просмотра сообщений при выполнении операций резервного копирования |
| Dtsrun.exe | Программа для запуска, удаления, просмотра и перезаписи пакетов DTS, хранящихся в структурированных COM-файлах, в сервере SQL 2000 и в структурах службы метаданных |
| dtwiz.exe | Программа запуска мастера создания пакета DTS(DTS Import Export Wizard) |
| isql.exe | Программа выполнения команд SQL, системных хранимых процедур или файлов команд, использующая интерфейс SQL Server 6.5 и библиотеку DB-Library |
| isqlw.exe | Программа запуска Query Analyzer |
| itwiz.exe | Программа запуска мастера индекса Index Tuning Wizard |
| makepipe.exe | Программа тестирования именованных каналов, используется совместно с утилитой readpipe.exe |
| odbccmpt.exe | Программа подключения средств совместимости с SQL Server 6.5для ODBC к конкретному приложению |
| odbcping.exe | Программа проверки возможности установления соединения с сервером SQL 2000посредством ODBC |
| osql.exe | Программа для выполнения команд SQL, системных хранимых процедур и файлов команд в режиме командной строки с использованием ODBC |
| rebuild.exe | Программа для перестройки системной базы данных master |
| readpipe.exe | Программа тестирования именованных каналов, используется совместно с утилитой makepipe.exe |
| distrib.exe | Программа конфигурирования агента Distributor Agent при работе с репликациями |
| logread.exe | Программа конфигурирования агента Logreader Agent при работе с репликациями |
| replmerg.exe | Программа конфигурирования агента Mergel Agent при работе с репликациями |
| snapshot.exe | Программа конфигурирования агента Snapshot Agent при работе с репликациями |
| scm.exe | Программа управления работой служб сервера в режиме командной строки |
| sqlagent.exe | Программа запуска службы SQL Server Agent из командной строки в качестве обычного приложения |
| sqldialog.exe | Программа сохранения информации из журнала сервера в текстовом файле \LOG\ SQLdiag.txt |
| sqlmaint.exe | Программа сопровождения базы данных для создания резервных копий усечения журнала транзакций и т.д. а также подготовки отчетов в текстовом файле или HTML –страниц для электронной почты |
| sqlserver.exe | Программа запуска службы MS SQL Server как приложения |
| sqlftwiz.exe | Программа для работы с мастером Full-Text Indexing Wizard |
| Vswitch.exe | Программа переключения версий сервера, установленных на одном компьютере |

Многие задачи администрирования MS SQL Server 2000 могут быть выполнены с помощью программ-мастеров. Это наиболее простой способ выполнения административных задач. Недостатком мастеров являются достаточно ограниченные возможности.

Таблица. Мастера SQL Server

|  |  |
| --- | --- |
| Название | Описание |
| Backup Wizard | Резервное копирование базы данных |
| Failover Setup Wizard | Организация кластеров на основе SQL Server |
| Configure Publishing аnd Distribution Wizard | Конфигурирование издателя и дистрибьютера при репликации |
| Create Alert Wizard | Создание оповещения |
| Create Database Wizard | Создание базы данных |
| Create Diagram Wizard | Создание диаграммы базы данных |
| Create Index Wizard | Создание индекса |
| Create Job Wizard | Создание задания |
| Create New Data Source Wizard | Установка ODBC–драйвера и ODBC–источника данных |
| Create Login Wizard | Создание учетной записи сервера для пользователя |
| Create Publication Wizard | Создание публикации для последующей репликации |
| Create Stored Procedure Wizard | Создание хранимой процедуры |
| Create Trace Wizard | Создание трассировки для Profiler |
| Create View Wizard | Создание представления |
| Create Maintenance Plan Wizard | Создание файла поддержки |
| Disable Publishing аnd Distribution Wizard | Удаление издателя и дистрибьютера для репликаций |
| DTS Export Wizard | Создание пакета DTS для экспорта данных из SQL Server |
| DTS Import Wizard | Создание пакета DTS для импорта данных в SQL Server |
| Full-text Indexing Wizard | Определение полнотекстовых индексов |
| Index Tuning Wizard | Оптимизация индексов |
| Make Master Server Wizard | Установка мастер-сервера |
| Make Target Server Wizard | Установка сервера-приемника |
| Register Server Wizard | Регистрация серверов в Enterprise Manager |
| Pull Subscription Wizard | Конфигурирование подписчика для вытягивания данных |
| Push Subscription Wizard | Конфигурирование подписчика с выталкивающим издателем |
| SQL Server Upgrade Wizard | Обновление базы данных SQL Server |
| Web Assistant Wizard | Создание Web-задачи |

Однако некоторых мастеров сказанное не касается. К ним можно отнести мастера конфигурирования подсистемы репликации, что является довольно сложным процессом. Например, чтобы создать публикацию средствами Enterprise Manager, нужно воспользоваться соответствующим мастером. Конечно, всегда можно применить средства Transact-SQL. Но иногда это настолько сложно и трудоемко, что лучшим решением будет использование мастера.

Программы-мастера запускаются кнопкой Run a wizard в окне Enterprise Manager или выполнением соответствующей утилиты в командной строке. В таблице 6 приведен список программ- мастеров, имеющихся в SQL Server 2000.

# 4.Протоколирование транзакций.

Транза́кция (англ. transaction, от лат. transactio — соглашение, договор) — минимальная логически осмысленная операция, которая имеет смысл и может быть совершена только полностью. Написание «трансакция» или «транзакция» зависит от значения слова.

В системе должны быть определены транзакции – неделимые работы, которые не могут быть выполнены частично. Они либо выполняются полностью, либо вообще не выполняются.

Модель неделимой транзакции пришла из бизнеса. Пусть, например, идет переговорный процесс двух фирм о покупке-продаже некоторого товара. В процессе переговоров условия договора могут многократно меняться, уточняться. Пока договор еще не подписан обеими сторонами, каждая из них может от него отказаться. Но после подписания контракта сделка должна быть выполнена от начала и до конца. Если контракт не подписан, то любые действия, которые были уже проделаны, отменяются или объявляются недействительными.

Незавершенная операция с диском несет угрозу целостности файловой системы. Необходимо протоколировать все изменения, происходящие в рамках транзакции, чтобы на основе этой информации в случае прерывания транзакции можно было отменить все уже выполненные подоперации, то есть сделать так называемый откаттранзакции.

Перед изменением какого-либо блока данных на диске или в дисковом КЭШе производится запись в специальный системный журнал – журнал транзакций, где отмечается, какая транзакция делает изменения, какой файл и какой блок изменяются и каковы старое и новое значения изменяемого блока.

ФС NTFS являются восстанавливаемой файловой системой, но восстанавливаемость обеспечивается только для системной информации файловой системы, то есть каталогов, атрибутов безопасности, битовой карты занятости кластеров и других системных файлов. Сохранность данных пользовательских файлов, работа с которыми выполнялась в момент сбоя, в общем случае не гарантируется.

Для повышения производительности ФС NTFS использует дисковый кэш, то есть все изменения файлов, каталогов и управляющей информации выполняются сначала над копиями соответствующих блоков в буферах оперативной памяти и только спустя некоторое время переносятся на диск. Кэширование повышает риск разрушения файловой системы. В таких условиях NTFS обеспечивает отказоустойчивость с помощью технологии протоколирования транзакций и восстановления системных данных.

Журнал транзакций в NTFS делиться на две части: область рестарта и область протоколирования.

Область рестарта содержит информацию о том, с какого места необходимо будет начать читать журнал транзакций для проведения процедуры восстановления системы после сбоя или краха ОС. Эта информация представляет собой указатель на определенную запись в области протоколирования. Для надежности в файле журнала регистрации хранятся две копии области рестарта.

· Область протоколирования содержит записи обо всех изменениях в системных данных файловой системы, произошедших в результате выполнения транзакций в течение некоторого периода. Записи о подоперациях, принадлежащих одной транзакции, образуют связанный список: каждая последующая запись содержит номер предыдущей записи. Заполнение области протоколирования идет циклически: после исчерпания всей памяти, отведенной под область протоколирования, новые записи помещаются на место самых старых.

Процесс восстановления файловой системы включает следующие шаги:

1)Чтение области рестарта из файла журнала транзакций и определение номера самой последней по времени записи о контрольной точке.

2)Чтение записи контрольной точки и определение номеров записей таблицы незавершенных транзакций и таблицы модифицированных страниц.

3)Чтение и корректировка таблиц таблицы незавершенных транзакций и таблицы модифицированных страниц на основании записей, сделанных в журнале транзакций уже после сохранения таблиц в журнале, но еще до записи журнала на диск.

4)Анализ таблицы модифицированных страниц, определение номера самой ранней записи модификации страницы.

5)Чтение журнала транзакций в прямом направлении, начиная с самой ранней записи модификации, найденной при анализе таблицы модифицированных страниц. При этом система выполняет повторение завершенных транзакций, в результате которого устраняются все несоответствия файловой системы, вызванные потерями модифицированных страниц в КЭШе во время сбоя или краха ОС.

6)Анализ таблицы незавершенных транзакций, определение номера самой поздней подоперации, выполненной в рамках незавершенных транзакций.

7)Чтение журнала транзакций в обратном направлении. Учитывая, что все подоперации каждой транзакции связаны в список, система легко переходит от одной записи модификации к другой, извлекает из них информацию, необходимую для отмены, и выполняет откат незавершенных транзакций.

# 5. Восстановление пользовательских баз данных.

Используя среду SQL Server Management Studio восстановление базы данных на момент времени сводится к выполнению следующих этапов:

1. В обозревателе объектов подключитесь к соответствующему экземпляру Компонент SQL Server Database Engine и разверните дерево сервера.
2. Разверните узел **Базы данных**. В зависимости от типа восстанавливаемой базы данных выберите пользовательскую базу данных или раскройте узел **Системные базы данных**и выберите системную базу данных.
3. Щелкните правой кнопкой мыши базу данных, выберите пункт **Задачи**, затем пункт **Восстановить**и пункт **База данных**.
4. Чтобы указать источник и расположение восстанавливаемых резервных наборов данных, используйте страницу **Общие** , раздел **Источник** . Выберите один из следующих вариантов.
   * **База данных**

Выберите из раскрывающегося списка базу данных для восстановления. Данный список содержит только базы данных, резервное копирование которых было выполнено в соответствии с журналом резервного копирования **msdb** .

**Примечание**

Если резервная копия была получена с другого сервера, на целевом сервере не будет журнала резервного копирования для указанной базы данных. В этом случае щелкните пункт **Устройство** , чтобы вручную указать файл или устройство для восстановления.

* + **Устройство**

Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования** . В окне **Тип носителя резервной копии** выберите один из перечисленных типов устройств. Чтобы выбрать одно или несколько устройств в окне **Носитель резервной копии** , нажмите кнопку **Добавить**.

После добавления нужных устройств в списке **Носитель резервной копии** нажмите кнопку **ОК** для возвращения на страницу **Общие** .

В списке **Источник: Устройство: База данных** выберите имя базы данных, которую нужно восстановить.

**Примечание.** Этот список доступен, только если выбрано **Устройство** . Будут выбраны только те базы данных, резервные копии которых доступны на выбранном устройстве.

1. В разделе **Назначение** , в поле **База данных** автоматически появится имя базы данных для восстановления. Для изменения имени базы данных введите новое имя в окно **База данных** .
2. Щелкните **Временная шкала** для получения доступа в диалоговое окно **Временная шкала резервного копирования** .
3. В разделе **Восстановить на** щелкните **Конкретные дата и время**.
4. Вы можете использовать окна **Данные** и **Время** или пользоваться ползунком, чтобы указать конкретные дату и время, при которых процесс восстановления должен закончиться. Нажмите кнопку **ОК**.

**Примечание**

Используйте поле **Интервал временной шкалы** для изменения количества времени, отображаемого на временной шкале.

1. После указания определенного момента времени помощник по восстановлению базы данных сделает так, что в столбце **Восстановление** сетки **Резервные наборы данных для восстановления** будут выбраны только необходимые для восстановления к указанному моменту времени резервные копии. Эти выбранные резервные копии составляют рекомендованный план восстановления для данного восстановления на момент времени. Следует использовать только выбранные резервные копии для операции восстановления на момент времени.
2. На странице **Параметры** используйте панель **Параметры восстановления** для выбора любого из следующих вариантов, если он подходит к данной ситуации.
   * **Перезаписать существующую базу данных (WITH REPLACE)**
   * **Сохранить параметры репликации (WITH KEEP\_REPLICATION)**
   * **Ограничить доступ к восстановленной базе данных (WITH RESTRICTED\_USER)**
3. Выберите параметр в поле **Состояние восстановления** . В данном окне определяется состояние базы данных после операции восстановления.
   * По умолчанию установлена схема**RESTORE WITH RECOVERY** , при этом база данных находится в готовом состоянии для использования путем отката незафиксированных транзакций. Невозможно восстановить дополнительные журналы транзакций. Выберите данный параметр, если выполняется восстановление всех необходимых резервных копий.
   * Схема **RESTORE WITH NORECOVERY** оставляет базу данных в нерабочем состоянии и не выполняет откат незафиксированных транзакций. Можно восстановить дополнительные журналы транзакций.База данных не может быть использована, пока не будет восстановлена.
   * Схема **RESTORE WITH STANDBY** оставляет базу данных в режиме только для чтения. С помощью данного параметра можно отменить незафиксированные транзакции и сохранить отмененные действия в резервном файле, чтобы результаты восстановления можно было отменить.
4. Настройка **Создать резервную копию заключительного фрагмента журнала перед восстановлением**будет выбрана, если это необходимо для указанного вами момента времени. Нет необходимости изменять данную настройку, однако вы можете выбрать резервное копирование заключительного фрагмента журнала, даже если выполнение этого не требуется.
5. Если имеются активные соединения с базой данных, то операция восстановления может завершиться ошибкой. Проверьте окно **Закрыть существующие соединения** и убедитесь, что все активные соединения между Среда Management Studio и базой данных закрыты. Этот параметр переводит базу данных в однопользовательский режим перед началом выполнения процедуры восстановления, а затем возвращает в многопользовательский режим после ее завершения.
6. Установите флажок **Выдавать запрос перед восстановлением каждой резервной копии** , если хотите отследить каждую операцию восстановления. Обычно это не требуется, за исключением случаев, если необходимо наблюдать за состоянием операции восстановления базы данных большого объема.

# 6. Восстановление системных баз данных.

Системные базы данных, на примере Microsoft SQL Server, являются важными объектами восстановления. База данных master содержит ключи базы данных, а также настройки системы безопасности; база данных msdb — графики и задания SQL Server, а также историю создания резервных копий. Полноценный план восстановления должен включать в себя воссоздание системных баз данных.

Для хранения данных используются таблицы, размещаемые в базах данных. В Microsoft SQL Server базы данных делят на два типа — системные и пользовательские. В системных базах данных размещаются метаданные, используемые для управления системой. При инсталляции Microsoft SQL Server создаются следующие системные базы данных: master, model, tempdb и msdb.

Краткое описание системных баз данных

|  |  |
| --- | --- |
| **Системная база данных** | **Описание** |
| **master** | В этой базе данных хранятся все данные системного уровня для экземпляра SQL Server. |
| **tempdb** | Рабочее пространство для временных объектов или взаимодействия результирующих наборов. |
| **msdb** | Используется агентом SQL Server для планирования предупреждений и задач. |
| **model** | Используется в качестве шаблона для всех баз данных, создаваемых в экземпляре SQL Server. Изменение размера, параметров сортировки, модели восстановления и других параметров базы данных **model** приводит к изменению соответствующих параметров всех баз данных, создаваемых после изменения. |
| **resource** | Системные объекты физически хранятся в базе данных **resource**, но логически отображаются в схеме **sys** любой базы данных. |

Подробное описание системных баз данных

Рассмотрим каждую из системных баз данных более подробно:

* **База данных master** — наиболее важная системная база данных Databse Engine. Она содержит в себе всю системную информацию, в том числе общие для всего экземпляра метаданные, такие как сведения об учетных записях входа, конечных точках и связанных серверах, а также параметры конфигурации системы. Кроме того, база данных **master**содержит информацию обо всех других базах данных. Здесь же SQL Server хранит сведения об инициализации. Таким образом, если база данных **master** станет недоступна, запустить SQL Server будет невозможно.

*Резервное копирование:* По причине важности этой системной базы данных необходимо всякий раз осуществлять ее резервное копирование, после выполнения любых системных операций, таких как добавление новых баз данных, добавления пользователей в SQL Server, изменений параметров экземпляра SQL Server и т. д.

* **База данных tempdb**— служит рабочим пространством для временных объектов или взаимодействия результирующих наборов. Она доступна всех пользователям подключенных к данному экземпляру SQL Server и используется всеми базами данных, принадлежащих экземпляру SQL Server. База данных **tempdb** пересоздается при каждом запуске SQL Server. Поэтому в базе данных **tempdb** ничего не сохраняется от одного сеанса SQL Server до следующего. В данной базе данных содержатся следующие объекты:
  + временные объекты, созданные явно, такие как глобальные или локальные временные таблицы, временные хранимые процедуры, табличные переменные и курсоры;
  + внутренние объекты, создаваемые компонентой Database Engine, например рабочие таблицы, хранящие промежуточные результаты буферов или сортировки, промежуточные результаты вычислений сложных выражений;
  + изменения данных в базе данных, в которой используются транзакции изоляции моментальных снимков с зафиксированным чтением и транзакции изоляции моментальных снимков;
  + изменения данных для таких функциональных возможностей, как операции с индексами в сети, множественные активные результирующие наборы (режим MARS) и триггеры AFTER.

*Резервное копирование:* Операции резервного копирования и восстановления базы данных **tempdb** запрещены.

*Замечание:* База данных **tempdb** используется системой довольно активно. По этой причине необходимо убедиться, что объем памяти для этой базы данных достаточно большой. Также можно получить значительный прирост производительности SQL Server путем выноса базы **tempdb** на отдельный дисковый накопитель, или более быстрый SSD-диск.

* **База данных model**— используется в качестве шаблона для всех баз данных, которые будут создаваться в данном экземпляре SQL Server. Поскольку база данных **tempdb** создается каждый раз при запуске SQL Server, база данных **model** всегда должна существовать в системе SQL Server. Все содержимое базы данных **model**, включая параметры базы данных, копируется в создаваемую базу данных. Соответственно, возможно изменять данную базу данных, добавляя в нее объекты или полномочия. После этого все новые базы данных будут наследовать и эти новые свойства.

*Резервное копирование:* Следует делать резервную копию базы данных **model**, после каждого изменения этой базы данных.

* База данных **msdb**— используется агентом SQL Server для создания расписания предупреждений и заданий, а также другими компонентами, такими как среда «Среда SQL Server Management Studio», компонента «Service Broker«» и «Database Mail» и др. Например, SQL Server хранит полный журнал резервного копирования и восстановления в базе данных **msdb**. В эти сведения включено имя стороны, выполнившей резервное копирование, время резервного копирования и устройства или файлы, в которых храниться резервная копия. Среда «SQL Server Management Studio» использует эти сведения для создания плана восстановления базы данных и применения существующих резервных копий журнала транзакций. События резервного копирования для всех баз данных записываются, даже если они создаются средствами пользовательских приложений или сторонних разработчиков.

*Резервное копирование:* Следует делать резервную копию базы данных msdb, после каждого изменения параметров агента SQL Server.

* **База данных resource** — это доступная только для чтения база данных, которая содержит все системные объекты, включенные в SQL Server. Такие системные объекты SQL Server, как **sys.objects**, физически хранятся в базе данных **resource**, но логически присутствуют в схеме **sys** каждой базы данных. База данных **resource** не содержит пользовательских данных или метаданных. Содержимое этой базы данных обычно используется для системных обновлений.

*Резервное копирование:* Резервное копирование базы данных **resource** не предусмотрено.

Резервное копирование базы данных master

Резервное копирование базы данных master ничем не отличается от резервирования пользовательских баз.

Обязательно резервируйте эту базу данных при выполнении операций, перечисленных ниже.

* Создание и удаление баз данных.
* Модификация системы безопасности за счет дополнения регистрационных записей и изменения ролей.
* Изменение каких-либо параметров конфигурации базы данных.

Поскольку база данных msdb хранит записи всех операций резервного копирования, после резервирования базы данных master заархивируйте и MSDB.

Восстановление базы данных master

Если база данных master была повреждена, то сервер не может быть запущен. Попытка запустить SQL Server в диспетчере служб операционной системы ни к чему не приведет. Попытка подключиться к экземпляру сервера в Management Studio приведет к открытию окна предупреждения о том, что сервер не существует или доступ к нему запрещен.

Единственным решением является запуск SQL Server в монопольном режиме (т.е. с параметром -ш) и восстановление базы master так, будто она является обычной пользовательской базой данных.

Системная база данных msdb

Подобно базе данных master, база MSDB по умолчанию использует простую модель восстановления.

Так как эта база данных содержит информацию, относящуюся к заданиям и графикам агента SQL Server Agent, а также к истории создания резервных копий, она должна резервироваться в следующих ситуациях.

* Выполнение любого типа резервирования.
* Сохранение пакетов DTS.
* Создание новых заданий агента SQL Server Agent.
* Конфигурирование почты и операторов SQL Server Agent.
* Конфигурирование репликаций.
* Создание и изменение графиков заданий.

Резервирование базы данных msdb ничем не отличается от архивирования обычных пользовательских баз данных.

Для восстановления базы данных msdb следует перевести сервер баз данных в монопольный режим, как и в случае с базой данных master. В то же время данное восстановление отличается от обычного, поскольку текущая база данных msdb, в которой хранится история создания резервных копий, повреждена. Таким образом, архив msdb не может быть выбран как резервная копия базы данных — он должен быть выбран как устройство резервирования.

Для проверки дискового устройства на предмет наличия конкретных резервных копий используется кнопка Contents. Если на устройстве содержится несколько экземпляров архивов, то для выбора нужного используется диалоговое окно Contents. После выбора нужной резервной копии ее номер подставляется в форму воссоздания базы данных.

Полное воссоздание сервера баз данных

Если сервер баз данных вышел из строя, резервные копии всех баз данных должны быть воссозданы на новом сервере. Порядок действий при этом следующий:

1. В операционной системе Windows восстановите все учетные записи домена для поддержки аутентификации Windows.
2. Установите SQL Server и все доступные пакеты обновлений.
3. Переведите SQL Server в монопольный режим и восстановите базу данных master.
4. Воссоздайте базу данных msdb.
5. Если база данных model изменялась, восстановите и ее.
6. Восстановите все пользовательские базы данных.

Гладкое выполнение операций восстановления является одним из самых главных навыков администратора баз данных.

Заключение

Цикл восстановления начинается с резервного копирования базы данных. Способность выживать после технических сбоев и фатальных ошибок пользователей является одним из самых критичных свойств ACID базы данных. Если база данных не обеспечивает живучесть транзакций, то ей нельзя доверять в полной мере. По этой причине планирование восстановления и журнал транзакций призваны обеспечить живучесть подтвержденных транзакций.

# 7. Управление учетными записями пользователей.

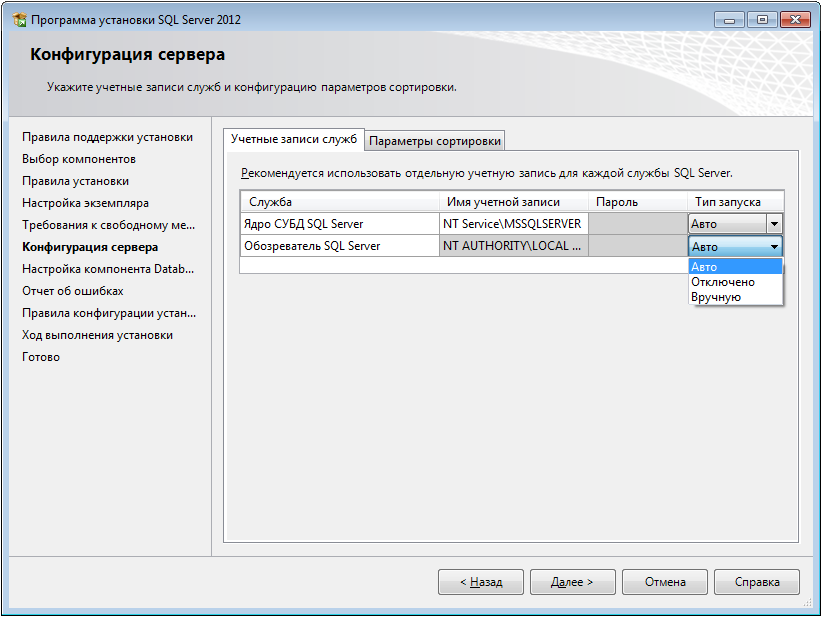
В справке Access 2002 сказано, что в проекте Access можно выполнять задачи администрирования системы безопасности с помощью команды Access **Безопасность базы данных**(Database Security) в меню **Сервис**(Tools). Эта команда доступна только в том случае, если SQL Server установлен на том же самом компьютере, где хранится проект Access. С помощью этого средства можно добавлять, удалять и изменять учетные записи для регистрации на SQL Server, учетные записи пользователей базы данных и роли.

**Замечание**

Для новой версии ядра баз данных MSDE2000, меню Безопасность базы данных (Database Security) а также меню Репликация (Replication) недоступны. Так говорится в справке Access 2002. Однако нам не удалось получить названные меню даже в проекте Access 2002, подключенном к SQL Server 2000, установленом на том же компьютере. Используйте инструкции Transact-SQL или Enterprise Manager для защиты и репликации данных на SQL Server. (Подробнее см. справку SQL Server.)

Существует две системы безопасности, применяемые для регистрации на SQL Server:

* *Встроенная система защиты SQL Server.*Для регистрации на сервере требуется указать имя пользователя сервера и пароль.



* *Интегрированная система защиты с Windows NT,*использующая учетные записи пользователей NT. В этом случае аутентификация пользователя производится в момент регистрации в сети на базе Windows NT. SQL Server принимает учетную запись Windows NT и не требует дополнительной регистрации пользователя.

Чтобы использовать встроенную систему защиты SQL Server:

1. Так как по умолчанию эта возможность выключена, сначала выберите команду меню **Вид, Свойства сервера**(View, Server Properties) и в появившемся диалоговом окне установите флажок **Учетная запись системного администратора**(SA) (Enable system administrator (SA) user name). Нажмите кнопку **ОК.**
2. Откройте окно свойств соединения с SQL Server **Свойства подключения**(Data Link Properties) с помощью команды меню **Файл, Подключение**(File, Connection).
3. На вкладке **Соединение**(Connection) установите флажок **Использовать имя пользователя и пароль**(Use a specific username and password).
4. Если при открытии проекта Access 2002 нужно каждый раз требовать ввод пароля регистрации пользователя, сбросьте флажок **Сохранять пароль**(Allow saving of password). Если ввод пароля при повторной регистрации на сервере не требуется, установите этот флажок.

С такой системой защиты вы сможете поменять пароль учетной записи SQL Server, использующейся для регистрации в проекте Access 2002, с помощью команды меню Access**Сервис, Защита, Задать пароль для входа**(Tools, Security, Set Login Password). He забудьте установленный пароль, чтобы не потерять доступ к проекту!

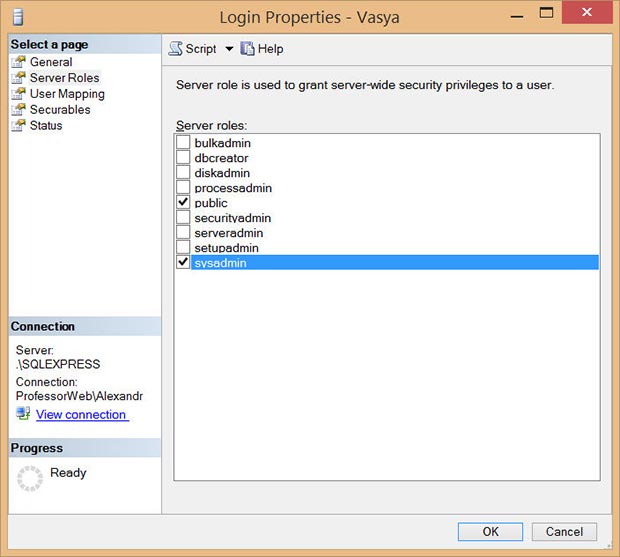
Чтобы использовать интегрированную с Windows NT систему защиты SQL Server:

1. Откройте окно свойств соединения с SQL Server **Свойства подключения**(Data Link Properties).
2. На вкладке **Соединение**(Connection) выберите переключатель **Использовать интегрированную систему защиты с Windows**NT (Use Windows NT Integrated Security).

В этом случае ввод пароля при регистрации на SQL Server требоваться не будет, поскольку будет действовать идентификация пользователя в сети Windows NT или Windows 2000.

# 8. Роли пользователей. Администрирование сервером SQL Server 2000.

Определение роли пользователя средствами графического интерфейса SQL Server Management Studio.



На уровне сервера служба безопасности оперирует понятием «**учетная запись**» (*login*). При подключении к SQL Server требуется ввести *login* и пароль. *Login* внутри SQL Server-а может совпадать с *login*-ом операционной системы (называется такая процедура подключения – аутентификация Windows. Так сейчас сделано в а-303).

Каждой учетной записи можно предоставить права на манипуляции со всеми (!) базами сервера. Эти права называются **ролями сервера**(*server role*).

Имеется 9 стандартных ролей сервера (*fixed server role*):

*Sysadmin* – любые типы администрирования SQL Server-а.

*Serveradmin* – конфигурирование и выключение сервера.

*Setupadmin* – управление связанными серверами и процедурами, автоматически запускаемыми при старте SQL Server.

*Securityadmin* – регистрация и удаление учетных записей, выдача пользователям прав на доступ к базам данных.

*Processadmin*– управление процессами, запущенными в SQL Server.

*Dbcreator* – может создавать и изменять базы данных.

*Diskadmin* – управляет файлами SQL Server.

*Bulkadmin* – управление массивным копированием (при архивировании и восстановлении баз из резервных копий).

Новые учетные записи можно создавать с помощью Enterprise Manager, а можно с помощью команд SQL. Для создания новой учетной записи вызывается системная хранимая процедура:

**sp\_addlogin** **@loginame = ‘**новый\_login**’**

[**,@passwd = ‘**пароль**’**]//если нужен пароль

[**,@defdb = ‘**имя\_базы\_данных\_по\_умолчанию**’**]// по умолчанию системная база ‘master’

[**,@deflanguage = ‘**язык\_по\_умолчанию**’**]

[**,@sid = ‘**SID**’**]

[**,@encryptopt = ‘**опция\_шифрования\_пароля**’**]

SID(*Security ID*) – двоичное число, которое является идентификатором безопасности учетной записи.

"Опция\_шифрования\_пароля" может принимать одно из значений:

* NULL – пароль данной учетной записи будет шифроваться (по умолчанию);
* Skip\_encryption – пароль не будет шифроваться;
* Skip\_encryption\_old – используется старый способ шифрования поролей, принтый в старых версиях SQL Server (используется для совместимости).

Для создания учетных записей SQL Server, совпадающих с login-ами отдельных пользователей Windows NT или группы пользователей используется хранимая процедура:

**sp\_grantlogin @loginname = ‘**Login\_в\_WinNT**’**

Login\_в\_WinNT включает полное имя пользователя с указанием домена. Например, в домене *VT* определена учетная запись *student303*. Регистрация производится командой:

EXEC sp\_grantlogin ‘VT\student303’

Создание учетных записей через Enterprise Manager: выделить значок *logins* в папке *security* нужного сервера → в правой части панели откроется список учетных записей.

Внутри базы данных система безопасности оперирует понятием **пользователь** (*user*). Пользователям выдаются различные права (изменение, добавление, удаление) на объекты той базы данных, в которой они зарегистрированы.

Одна учетная запись (*login*) ассоциируется в каждой базе данных, к которой она имеет доступ, с одним пользователем (*user*). При этом один *user* может использоваться для предоставления доступа нескольким людям, подключенным к серверу под разными *login*-ами.

Например, учетная запись, имеющая доступ к базам данных A, B, C, может проходить в базе A под именем пользователя ‘vasya’, в базе B под именем ‘grant’, а в базе C еще под каким-то именем.

Для связывания учетной записи (*login*) с определенным именем пользователя (*user*) может использоваться хранимая процедура:

**sp\_adduser @loginname = ‘***login***’** // имя учетной записи

[**,@name\_in\_db = ‘***user***’**] // имя пользователя базы данных

[**,@grpname = ‘***role***’**]// имя роли

Если используется учетная запись Windows NT, то ее отображение в имя пользователя производится с помощью хранимой процедуры:

**sp\_grantdbaccess @loginame = ‘***login*’ // Login в Windows NT

[**,@name\_in\_db = ‘***user***’**] // имя пользователя базы данных

Если необязательный параметр *name\_in\_db* не указывать, создается *user* с именем, совпадающим с *login*.

Внутри базы данных права доступа к объектам этой базы распределяются на уровне пользователей. Например, один пользователь имеет право добавлять и удалять строки таблицы A, таблицу B только читать, а в таблице С изменять существующие строки.

Администратор БД имеет полный доступ ко всем объектам базы.

Пользователь, который создал объект, считается **владельцем объекта**и имеет на этот объект все права (изменение, добавление, удаление). Владелец объекта может назначать другим пользователям права доступа на свой объект.

При создании базы данных определяются два стандартных пользователя: *dbo* и *guest (public)*.

**guest (public)** – имеет минимальные права доступа. Любой новый пользователь автоматически получает уровень прав *guest*.

**dbo** – (*Database owner* – владелец базы данных) – пользователь с максимальными правами на базу.

Если несколько пользователей имеют одинаковые права доступа к объектам БД, этих пользователей удобно объединить в группу и давать права доступа общие для всех членов группы. Такие группы пользователей в SQL Server называются **ролями базы данных**(*database role*). Можно назначить права доступа к объектам базы данных для конкретной роли и работать с ней как с обычным пользователем. Например, если сотрудник переходит в другой отдел, достаточно удалить его из одной роли и добавить в другую, а не устанавливать новые права для каждого пользователя.

Роли базы данных бывают фиксированными, или стандартными (*fixed database role*), и роли пользователя (их ‘вручную’ определяет администратор БД).

Фиксированные роли базы данных:

*db\_owner* – владелец всех объектов базы, имеет максимальные права.

*db\_accessadmin* – может добавлять и удалять пользователей.

*db\_securityadmin* – управлять всеми разрешениями, объектами, ролями и членами ролей.

*db\_ddladmin* – может выполнять любые команды DDL, кроме GRANT, DENY и REVOKE.

*db\_backupoperator*– может выполнять команды DBCC, CHEKPOINT и BACKUP.

*db\_datareader* – может просматривать любые данные в любой таблице БД.

*db\_denydatareader* – запрещается просмотр данных в любой таблице БД.

*db\_denydatawriter* – запрещается модифицирование данных в любой таблице БД.

*Public* – роль с минимальными правами, которая присваивается каждому пользователю.

В роль базы данных можно добавить пользователей, другие роли SQL Server, отдельных пользователей и группы пользователей Windows NT (которым предварительно разрешен доступ к соответствующей базе данных *login*-ом). Это делает системная хранимая процедура

**sp\_addrolemember @rolemember = ‘**имя роли в текущей БД**’**

**@membername = ‘**имя объекта**’**//*login*-ыWinNT,*user*-ы и имена ролей

Другие хранимые процедуры:

**sp\_addrole** – создать роль;

**sp\_droprole** – удалить роль;

**sp\_droprolemember** – удалить пользователя из роли;

**sp\_dropuser** – удалить пользователя из базы данных.

Когда пользователи подключаются к SQL Server, действия, которые они могут выполнять, определяются **правами** (или разрешениями – *permissions*), выданными их учетной записи, роли или *user*-у.

Права в SQL Server делятся на 3 категории:

1. Права на доступ к объектам баз данных

Под объектами базы данных понимаются: таблицы (можно регулировать доступ как к таблицам целиком, так и к отдельным столбцам таблиц), представления (view), хранимые процедуры и функции.

|  |  |  |  |
| --- | --- | --- | --- |
| Название права | Применимость | | |
| Таблицы и представления | Столбцы  таблиц и  представлений | Хранимые процедуры и функции |
| **SELECT**– выборка (чтение) данных | + | + | – |
| **INSERT**– вставка новых строк в представление или таблицу | + | - | – |
| **UPDATE**– изменение данных | + | + | – |
| **DELETE**– удаление строк из таблицы или представления | + | – | – |
| **REFERENCES**– возможность ссылаться на указанный объект. Применительно к таблицам разрешает пользователю создавать внешние ключи, ссылающиеся наPRIMARYKEYилиUNIQUEстолбец этой таблицы.  Применительно к представлениям (view) позволяет связывать представление со схемами таблиц, на основе которых строится представление. Это позволяет отслеживать изменение структуры исходных таблиц, которые могут повлиять на работу представления | + | – | – |
| **EXECUTE**– выполнение хранимых процедур и функций | – | – | + |

Предоставить права доступа к объектам бд позволяет команда

**GRANT**

**{ ALL PRIVILEGES** | *permission* [, ... *n*] **}**

**{**

**[(***имена\_столбцов\_через\_запятую***)] ON***имя\_таблицы\_или\_представления*

**| ON** *имя\_процедуры*

**| ON***имя\_таблицы\_или\_представления* **[(***имена\_столбцов\_через\_запятую***)]**

**}**

**TO** *имена\_объектов\_системы\_безопасности* //через запятую*login*-ы,*user*-ы и имена ролей

**[WITH GRANT OPTION]** // для передачи своих прав другим пользователям командойGRANT

**[AS {***имя\_роли* | *имя\_группы\_WinNT***}]**// имя роли, обладающей правомGRANT, от имени которой

**//**происходит передача прав.

Параметр*permission* – через запятую список доступных разрешений SELECT, INSERT, UPDATE и т.д. (из таблицы выше).

Например:

GRANT SELECT, INSERT

ON Materials

TO Engineer

WITH GRANT OPTION

Члены группы *Engineer* получают права на просмотр и добавление записей в таблицу *Materials*. Любой из пользователей группы *Engineer* впоследствии может передавать все или часть своих прав другим пользователям (в нашем примере пользователю с именем *Liss*):

GRANT SELECT, INSERT

ON Materials

TO Liss

AS Engineer

**2. Права на выполнение команд Transact-SQL** даются на следующие команды

CREATE DATABASE

CREATE TABLE

CREATE DEFAULT – создание умолчаний.

CREATE RULE – создание правил.

CREATE PROCEDURE – создание процедур.

BACKUP DATABASE – резервное копирование базы данных.

BACKUP LOG – резервное копирование журнала транзакций.

ALL – все перечисленные выше права.

Права на выполнение команд Transact-SQL даются командой:

**GRANT {ALL |***перечисленные\_выше\_команды\_через\_запятую***}**

**TO** *имена\_объектов\_системы\_безопасности*

Запрещение доступа к объектам – синтаксис аналогично GRANT:

DENY

**{ ALL PRIVILEGES** | *permission* [, ... *n*] **}**

**{**

**[(***имена\_столбцов\_через\_запятую***)] ON***имя\_таблицы\_или\_представления*

**| ON** *имя\_процедуры*

**| ON***имя\_таблицы\_или\_представления* **[(***имена\_столбцов\_через\_запятую***)]**

**}**

**TO** *имена\_объектов\_системы\_безопасности* //через запятую*login*-ы,*user*-ы и имена ролей

**[CASCADE]**// отзывать права не только у данного пользователя, но и у всех тех пользователей, которым он предоставил данные права командойGRANT.

Запрещение выполнения команд SQL:

**DENY {ALL** | *список*\_*команд*}

**TO** *имена*\_*объектов*\_*безопасности*

Неявное отклонение доступа (*revoke access*) действует только на том уровне, на котором определено. Если пользователю на определенном уровне неявно отклонен доступ, он может получить его на другом уровне иерархии через членство в роли, имеющей право просмотра.

REVOKE [GRANT OPTION FOR]

**{ALL PRIVILEGES** | *список\_разрешений*}

**{**

**[(***имена\_столбцов\_через\_запятую***)] ON***имя\_таблицы\_или\_представления*

**| ON** *имя\_процедуры*

**| ON***имя\_таблицы\_или\_представления* **[(***имена\_столбцов\_через\_запятую***)]**

**}**

**{TO | FROM}** *имена\_объектов\_безопасности*

**[CASCADE]**

**[AS** {*группа* | *имя\_роли***}]**

GRANT OPTION FOR используется, когда необходимо отозвать право, предоставляемое параметром WITH GRANT OPTION команды GRANT. Пользователь оставляет за собой право доступа, но больше не может передавать его другим.

Неявное отклонение команд SQL выполняется с помощью оператора

**REVOKE {ALL**| *список\_команд***}**

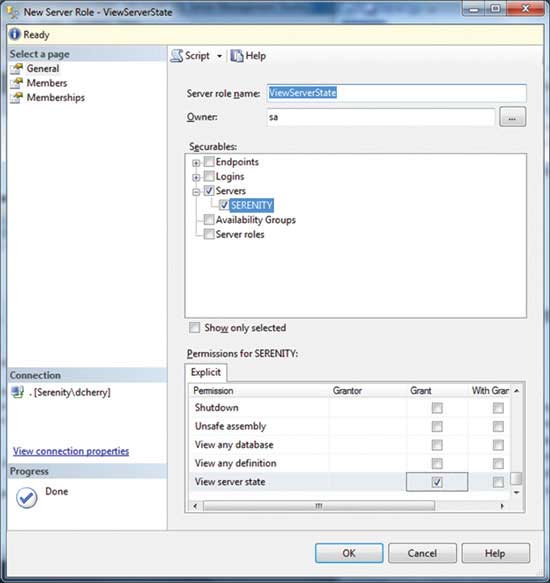
**FROM** *имена*\_*объектов*

Конфликт доступа возникает, если одному *user*-у некое право доступа разрешено через членство в первой роли, но запрещено через членство в другой роли или лично.

При разрешении конфликтов предпочтение отдается запрещению.

**Консоль SQL Server Management Studio**

Задачу можно легко выполнить средствами графического интерфейса **SQL Server Management Studio**. Чтобы создать пользовательскую серверную роль, подключитесь к экземпляру средствами Object Explorer. В обозревателе объектов перейдите к узлу <имя\_экземпляра>/Security/Server Roles. Щелкните правой кнопкой Server Roles и выберите New Server Role. В открывшемся окне New Server Role в поле Server role name укажите имя роли, в поле Owner укажите владельца роли, после чего укажите объекты и разрешения, которые должны предоставляться членам роли (рис. 1).



*Рис. 1. Укажите права, которые должна предоставлять роль*

По завершении работы на странице General перейдите на страницу Members (рис. 2) и укажите имена входа **SQL Server**, которые должны входить в эту пользовательскую серверную роль.

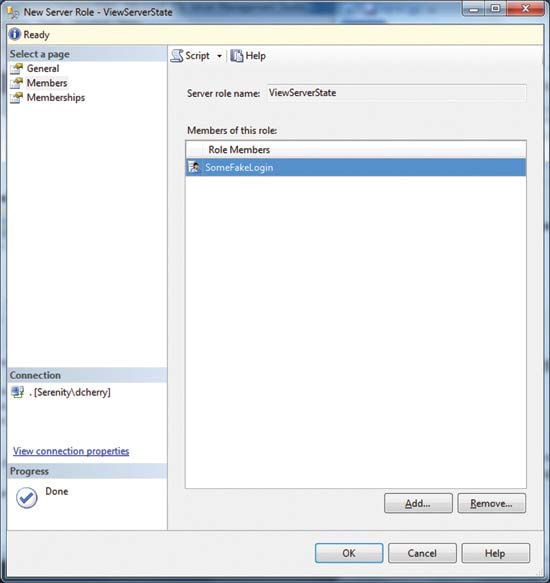


Рис. 2. Определение членов роли

После определения членов роли на странице Members перейдите на страницу Memberships. Здесь задаются серверные роли, членом которых должна быть пользовательская серверная роль. Если на этой странице выбрать какую-то роль, пользователи, члены создаваемой пользовательской серверной роли, будут также обладать правами выбранной роли.

Если создать пользовательскую роль и сделать ее членом фиксированной серверной роли serveradmin (рис. 3), все члены вновь созданной пользовательской роли будут также обладать правами роли serveradmin. Как и при вложении доменных групп или ролей баз данных, при вложении ролей надо соблюдать большую осторожность, чтобы не предоставить пользователям прав, которых у них быть не должно

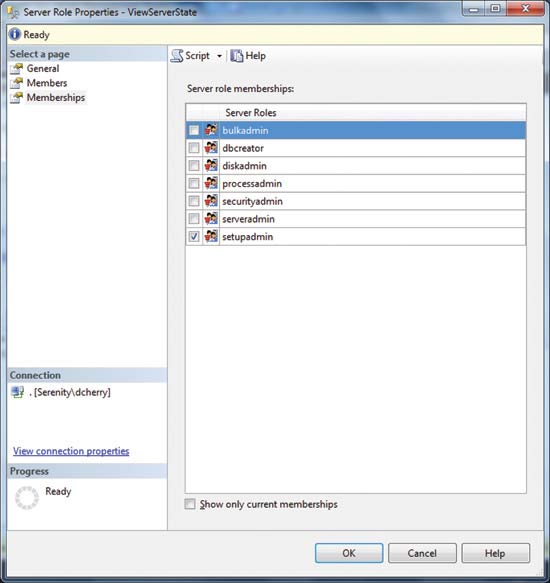


Рис. 3. Пользовательские роли можно включать в другие серверные роли

# 9. Настройка дополнительных компонентов (служб электронной почты, настройка связанных серверов).

Компоненты сервера

Службы Reporting Services - это серверный продукт, следовательно, большая часть компонентов этой службы размещается на сервере.

* **Базы данных служб Reporting Services**.Базы данных служб Reporting Services хранят метаданные решений, сохраненные отчеты, информацию, имеющую отношению к сеансам и все, что необходимо для обслуживания состояния сервера. Службы Reporting Services управляют двумя базами данных: ReportServer, которая включает всю постоянную информацию, и ReportServerTempDB, которая содержит всю временную информацию. Эти базы данных можно найти в каталоге баз данных SQL Server, как любые обычные базы данных.
* **Сервер отчетов**. Этот виртуальный компонент отвечает за обработку отчетов, используя поддержку всех блоков расширений. Он реализован в виде двух больших блоков:
  + **Служба Windows сервера отчетов**. Служба Windows сервера отчетов отвечает за инициализацию и обслуживание серверов, а также за вывод процессов планирования и доставки, с помощью доступных расширений доставки.
  + **Веб-служба сервера отчетов**. Службы Reporting Services взаимодействуют с внешним миром, главным образом, через интерфейс веб-службы. Действительно, именно этот интерфейс использует Диспетчер отчетов для выполнения своих обязанностей. Доступны две веб-службы: ReportingServices2005 (для управления) и ReportExecutionService (для выполнения отчетов).

Эти два блока зависят от некоторых функциональных модулей, в том числе, от различных обработчиков и расширений.

* **Обработчик сервера отчетов**. Этот модуль делает всю основную работу по синтаксическому анализу определения отчета, запросу данных от расширения обработки данных, выполнения вычислений и отправке вывода на расширение интерпретации для форматирования вывода при необходимости.
* **Обработчик планирования и доставки**. Отвечает за запуск и обслуживание всех запланированных и завершившихся процессов, а также за передачу вывода соответствующему расширению доставки. Модуль взаимодействует с обработчиком сервера отчетов и службой Агент SQL Server.
* **Расширение проверки подлинности**. Этот модуль отвечает за проверку подлинности и авторизацию пользователей, обращающихся к службам Reporting Services. Для использования пользовательских или неподдерживаемых механизмов проверки подлинности необходимо создать и зарегистрировать пользовательские сборки.
* **Расширение обработки данных**. Данный модуль отвечает за обработку взаимодействия с источниками данных. Источники данных, которые не требуют дополнительной настройки - это SQL Server, службы Analysis Services, Oracle, OLE DB, ODBC, поставщики данных ADO.NET и XML. Для доступа к неподдерживаемым базам данных необходимо создать и зарегистрировать пользовательские сборки.
* **Расширение интерпретации**. Службы Reporting Services используют это расширение для формирования окончательной интерпретации отчета в выбранном формате. В Reporting Services имеется несколько расширений подготовки. HTML, PDF, Excel, текст с разделителями "запятая" (CSV), XML, и Image. Чтобы перевести отчеты в дополнительные форматы, следует создать и зарегистрировать пользовательские сборки.
* **Служба Агент SQL Server**. Службы Reporting Services используют Агент SQL Server для обработки всех задач, связанных с планированием. Службы Reporting Services создают и удаляют задания Агента SQL Server в определенной им последовательности. Имена заданий, которые генерируются службами Reporting Services, начинаются с GUID, и могут быть использованы в списке заданий Агента SQL Server.

Эти задания выполняют очень простую работу: они вставляют в базу данных службы Reporting Services строку "schedule fired". Вставка этой строки инициирует на Сервере отчетов действия, связанные с расписанием.

* **Конфигурационные файлы**. Хотя в большинстве случаев эту работу выполняют службы Reporting Services и инструменты настройки конфигурации SQL Server, вам следует знать о существовании нескольких файлов конфигурации (.NET-файлов \*.config) а также о том, что изменить некоторые параметры можно только прямым редактированием таких конфигурационных файлов.
* **Дополнительные компоненты**. Некоторые задачи, связанные с созданием отчетов, например, проверка подлинности или доставка, при выполнении своих обязанностей зависят от приложений сторонних разработчиков или серверов. Например, доставка по электронной почте потребует участия SMTP-сервера, а проверка подлинности - доступности определенных видов полномочий проверки подлинности. Следовательно, установка системы отчетов может быть нарушена, если нарушена связь с этими элементами.

Клиентские компоненты

Службы Reporting Services предоставляют интерфейсы для того, чтобы внешние приложения могли осуществлять обмен данными с сервером. Среди доступных интерфейсов - прямой доступ через URL и API веб-служб. Не существует ограничения на количество возможных клиентских компонентов. В установке SQL Server и служб Reporting Services без дополнительной настройки доступны следующие клиентские инструменты.

* **Диспетчер отчетов**. Хотя этот модуль в действительности выполняется на сервере, потому что является приложением ASP.NET, он включен в этот раздел, потому что действует в качестве клиента. Диспетчер отчетов позволяет конечным пользователям обращаться к отчетам и управлять ими в соответствии с набором полномочий, которыми они обладают. Диспетчер отчетов управляет веб-приложением для администраторов служб Reporting Services.
* **Диспетчер конфигурации служб Reporting Services**. Как уже говорилось, конфигурация представляет собой критически важный аспект работы служб Reporting Services. Помимо непосредственного редактирования конфигурационных файлов, для централизованного редактирования большей части параметров конфигурации можно использовать Reporting Services Configuration Manager (Диспетчер конфигурации служб Reporting Services).
* **Диспетчер конфигурации SQL Server**. С помощью этого инструмента вы сможете запустить и остановить службу Windows Reporting Service и осуществить доступ к Диспетчеру конфигурации служб Reporting Service.
* **SQL Server Management Studio**. SQL Server Management Studio позволяет работать с некоторыми аспектами служб Reporting Services, например, хранением информации о пользователях, которым разрешен доступ к службам Reporting Services, ролях, совместно используемых расписаниях, папках и объектах служб Reporting Services, а также генерацией моделей отчетов *SMDL*(Semantic Model Definition Language, языка определения семантических моделей). SQL Server Management Studio следует рассматривать как альтернативу Диспетчеру отчетов для администраторов.

**Дополнительная информация** Дополнительную информацию о каждом из этих компонентов можно найти в теме "Обзор компонентов Reporting Services" Электронной документации по SQL Server 2005.

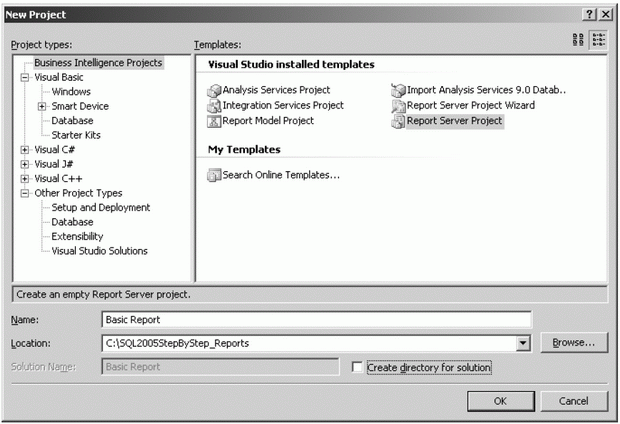
Масштабируемость

Хотя подробное описание того, как масштабировать установку Reporting Services с одного сервера до полномасштабной установки, в данной книге не приводится, стоит рассказать здесь о возможностях масштабирования.

Службы Reporting Services встроены в такие стандартные технологии Microsoft, как базы данных SQL Server, ASP.NET и Internet Information Server (IIS), об этом мы рассказывали в предыдущих разделах. Таким образом, службы Reporting Services могут использовать возможности масштабирования, предоставляемые этими технологиями.

Не вдаваясь в детали, можно считать, что службы Reporting Services можно расширить до веб-фермы, так же, как любую установку IIS, или оставить кластер SQL Server, как и в любой корпоративной установке.

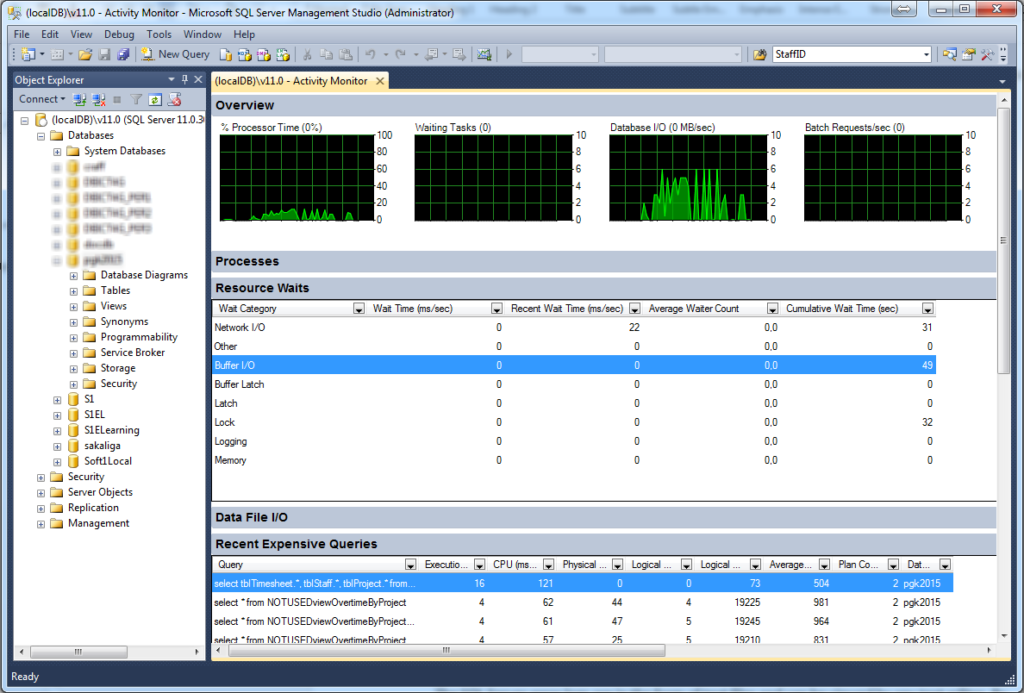
Окно создания проекта отчета изображено на рис.



# 10. Мониторинг работы SQL Server 2000.

Для анализа производительности SQL Server 2005 существует программ: системный монитор (System Monitor), logman.exe, relog.exe, typeperf.exe, HostMonitor, AppManager.

Рабочее окно одной и из подобных программ (Activity Monitor) изображено на рис.



На производительность приложения, работающего с SQL Server, влияют самые разные факторы. Для анализа каждого из таких факторов вам потребуются свои средства.

Наиболее универсальное средство мониторинга и анализа производительности — это Системный монитор (*System Monitor*). В Windows NT 4.0 он назывался Монитором производительности (*Performance Monitor*). Это средство предназначено для работы со счетчиками производительности как для операционной системы, так и для SQL Server. Про работу с Системным монитором подробно рассказывается в следующих разделах. Здесь отметим только некоторые моменты:

в Windows Server 2003 появилась версия Системного монитора, которая предназначена для работы из командной строки. Это утилита logman. Она позволяет выполнять протоколирование показаний счетчиков в файлы на диске или источники данных ODBC точно так же, как и графическая версия Системного монитора. Если вам постоянно приходится выполнять замеры производительности, собирая показатели одного и того же набора счетчиков, то будет удобнее написать пакетный файл с командной строкой на запуск этой утилиты с необходимыми параметрами;

q      утилита командной строки relog, которая также появилась только в Windows Server 2003, позволяет производить конвертацию файлов с собранными показаниями счетчиков из одного формата Системного монитора в другой (например, из двоичного формата BLG в текстовый CSV);

q      еще одна новая утилита Windows Server 2003, которая называется typeperf, предоставляет возможность для просмотра показаний счетчиков Системного монитора в командной строке;

q      существуют утилиты третьих фирм, которые позволяют работать со счетчиками Системного монитора, например, HostMonitor и AppManager. Эти утилиты предоставляют более удобный графический интерфейс для просмотра и анализа данных по производительности. Но, с точки зрения автора, те дополнительные возможности, которые дают эти утилиты, не принципиальны. Обычно они предлагают более удобные возможности просмотра тех же счетчиков Системного монитора. Однако вы всегда можете загрузить данные, собранные Системным монитором, в Excel и использовать для просмотра и анализа мощные средства этого приложения Microsoft Office.

При оптимизации производительности всегда имеет смысл ориентироваться на максимальную нагрузку сервера. Обычно проблемы с производительностью возникают именно в такие пиковые часы, когда с системой работает максимальное количество пользователей. Однако часто возникают ситуации, когда администратору приходится оценивать производительность приложения и принимать решение о целесообразности его приобретения до того, как оно развернуто на предприятии. Другая возможная ситуация — нужно принять решение о покупке нового сервера, но не совсем понятно, насколько мощное оборудование потребуется приложению при максимальной нагрузке.

В таких ситуациях может быть очень полезной искусственная имитация нагрузки со стороны пользователей — так называемое *нагрузочное тестирование* (*stress testing*).

Еще одно важное средство для анализа производительности работы приложения — профилировщик (SQL Server Profiler). Это средство позволяет найти и запротоколировать команды, которые передаются на сервер приложением, а также найти запросы, на выполнение которых требуется много времени.

Если вы подозреваете, что работу вашего приложения тормозят проблемы с сетью, то для анализа таких проблем вам потребуется Сетевой монитор*.*

Для анализа и оптимизации системы индексов удобнее всего использовать программное средство Database Tuning Advisor (Index Tuning Wizard в SQL Server 2000).

Для оптимизации запросов к SQL Server очень удобно использовать средства SQL Server Management Studio.

# 11. Администрирование систем типа Oracle.

СУБД Oracle всегда отличалась богатейшими функциональными возможностями и средствами, состав и количество которых увеличиваются в каждом новом выпуске СУБД. Это помогает пользователям сделать систему базы данных более надежной, мощной и эффективной, но этот рост функциональных возможностей СУБД также оказывает непосредственное влияние и на методы администрирования СУБД. Для того чтобы воспользоваться преимуществами новых функциональных возможностей, АБД должны пересматривать используемые методы администрирования и приводить их в соответствие с каждой новой версией программного обеспечения. В этой работе в общих чертах представлен ряд практических методов, которые могут помочь АБД воспользоваться новыми функциональными возможностями СУБД Oracle9*i* для повышения надежности, доступности, сопровождаемости и производительности систем баз данных. Мы отдаем себе отчет, что каждый заказчик имеет уникальную систему базы данных, поэтому предлагаемые практические методы должны учитывать это. В данной работе мы делаем акцент на обобщенных методах администрирования, независимых от платформ и среды, но в то же время важных для хорошего сопровождения СУБД. Рассматриваемые методы можно разделить на четыре категории:

1. Конфигурация систем и баз данных.
2. Сопровождение пространства и объектов.
3. Оптимизация производительности.
4. Резервирование и восстановление.

Подробное обсуждение каждой категории приведено ниже.

Конфигурация систем и баз данных

Перед созданием базы данных АБД должен принять ряд решений о конфигурировании системы. Правильно принятые решения могут в дальнейшем помочь избежать существенных проблем. В данную категорию попадают определение оптимальной конфигурации подсистемы хранения и выбор подходящего размера блоков базы данных. В данном разделе рассматриваются возможные проблемы и лучшие методы конфигурирования и создания баз данных.

Конфигурация внешней памяти

Оптимальное конфигурирование подсистемы хранения базы данных – очень важная задача в большинстве систем. Плохо сконфигурированная подсистема хранения может стать причиной возникновения узких мест ввода-вывода, которые могут существенно снизить скорость работы самой быстрой машины. Поэтому системные администраторы отдают должное конфигурированию внешней памяти и тратят на это очень много времени. Исторически конфигурирование систем хранения базировалось на глубоком понимании приложений базы данных, что приводило к необоснованному усложнению процесса конфигурирования, к уменьшению его управляемости. Альтернативой этого подхода является модель конфигурации внешней памяти SAME (**S**tripe **a**nd **M**irror **E**verything – расщепление и зеркалирование всех данных), в которой предлагается лучший и более управляемый подход. Эта модель была разработана экспертами корпорации Oracle, проделавших значительную исследовательскую работу по определению оптимальной конфигурации внешней памяти систем баз данных Oracle. Модель базируется на четырех простых предложениях:

1. Расщепляйте все файлы по всем дискам с шириной полосы расщепления, равной 1 Мб.
2. Зеркалируйте данные для обеспечения высокой доступности.
3. Распределяйте данные по секциям, а не по дискам.
4. Размещайте данные, к которым возможен частый доступ, на внешних половинах пространства дисков.

Размер блоков базы данных

Одно из первых решений, которое должен принять АБД при создании базы данных в СУБД Oracle, заключается в выборе размера блоков базы данных. Это делается во время создания базы данных установкой значения параметра инициализации DB\_BLOCK\_SIZE. Этот параметр может иметь значения 2К, 4К, 8К, 16К или 32К байтов. Как правило, размер блоков базы данных должен быть кратным размеру блоков операционной системы. Большой размер блоков базы данных повышает эффективность физического и логического ввода-вывода. Однако, если размер блоков слишком велик, то в каждом блоке можно будет размещать много логических записей, что в OLTP-среде может привести к возникновению конкуренции за блоки. Другой отрицательной стороной использования очень большого размера блоков является то, что при индексном доступе, когда извлекается только одна строка данных, без всякой необходимости читается много лишних данных. Таким образом, достоинства использования большого размера блоков следует балансировать с недостатками их использования. Размеры блоков 4Кб и 8Кб подходят для большинства сред (4Кб более подходит для OLTP-систем, а 8Кб – для DSS-систем).

Если во время создания базы данных выбран неподходящий размер блоков, в СУБД Oracle9i ситуацию можно исправить. Для этого нужно сначала создать новые табличные пространства с надлежащим размером блоков (в операторе создания используется предложение BLOCKSIZE). Затем, с помощью средств СУБД Oracle для оперативной реорганизации можно перенести объекты в эти новые табличные пространства (без остановки экземпляра СУБД). Для таких реорганизаций можно использовать графические средства инструментария Oracle Enterprise Manager. Следует заметить, табличное пространство SYSTEM нельзя реорганизовать оперативно, поэтому размер блоков этого табличного пространства можно изменить только путем повторного создания базы данных. Хорошая практическая рекомендация: во время создания базы данных постараться принимать верные решения и выбрать надлежащий размер блоков базы данных.

Метод создания базы данных

Процесс создания баз данных, их уничтожения и изменения структуры можно упростить с помощью графического инструментального средства конфигурирования Oracle Database Configuration Assistant (DBCA). В режиме адаптации к требованиям пользователей (custom) DBCA позволяет подключать различные опции СУБД (такие, как Spatial – обработка пространственных данных, OLAP-сервисы и т.п.), определять размерные параметры, местонахождение архивных и других файлов, атрибуты хранения, а также, в конце процесса создания базы данных, вручную выполнять все штатные SQL-скрипты. DBCA выполняет все требуемые шаги с минимальным вводом информации и устраняет необходимость детального планирования параметров и структуры базы данных. Более того, DBCA также выдает рекомендации о конкретных установках параметров базы данных в зависимости от введенной информации и, таким образом, помогает АБД принять правильные решения по выбору оптимальной конфигурации базы данных. DBCA можно использовать как для создания баз данных в режиме работы с одним экземпляром СУБД, так и для создания или добавления экземпляров в среде Oracle Real Application Cluster.

В СУБД Oracle9i в DBCA введено понятие шаблонов баз данных (database templates), которое существенно упрощает сопровождение баз данных. Шаблоны баз данных – это новый механизм хранения определений баз данных в XML-формате. В определения включаются все характеристики баз данных, такие, как параметры инициализации, журнальных файлов и т.п.; они могут использоваться для создания идентичных баз данных как в локальных машинах, так и в удаленных. Существует два вида шаблонов: для создания только структур баз данных и для создания структур совместно с данными. Эта функциональная возможность позволяет АБД создавать новые базы данных с помощью предопределенных шаблонов или создавать по существующей базе данных новые шаблоны для их последующего использования. В результате, с помощью этого нового мощного средства существующие базы данных можно "клонировать" с данными или без данных. Шаблоны можно сохранять и повторно использовать для развертывания новых баз данных.

Используя предопределенные шаблоны, DBCA можно запускать в пакетном (silent) режиме и автоматически создавать базы данных или связанные с ними приложения. Это означает, что запуск заданий создания баз данных можно осуществлять с помощью планировщика заданий ОС без какого-либо вмешательства АБД. Все эти возможности делают DBCA очень мощным средством, которое мы рекомендуем для создания баз данных и экземпляров СУБД Oracle.

Сопровождение пространства и объектов

В СУБД Oracle9i для облегчения сопровождения различных объектов базы данных и повышения производительности работы с ними сделан ряд усовершенствований. Карл Бутнер (Karl Buttner), президент компании "170 Systems", приветствуя новые средства СУБД Oracle9i, заявляет: "усовершенствованные средства администрирования и управления базами данных позволяют нашим заказчикам снизить их административные и эксплуатационные издержки, повышая при этом доступность и производительность своих систем". Одним из наиболее значительных усовершенствований в сфере сопровождения СУБД Oracle9i является появление опции Automatic Undo Management (автоматическое управление пространством отката транзакций) которая позволяет устранить необходимость вручную сопровождать сегменты отката. Коме того, улучшено управление табличными пространствами и сегментами, которое при соответствующем использовании позволяет существенно снизить объем работы АБД.

Автоматическое управление пространством отката транзакций

Для упрощения сопровождения сегментов отката в СУБД Oracle9i введено автоматическое управление пространством отката транзакций (Automatic Undo Management), которое позволяет серверу базы данных автоматически управлять выделением и сопровождением пространства отката транзакций для различных активных сеансов. Администраторам уже не нужно вручную создавать сегменты отката (rollback segments). Вместо этого, они могут просто создать табличное пространство отката (табличное пространство типа UNDO) и выделить для него достаточный объем дисковой памяти.

Первым преимуществом такого подхода является улучшение сопровождаемости. АБД больше не требуется назначать некоторым транзакциям конкретные сегменты отката или беспокоиться об их размере и количестве.

Другим преимуществом автоматического управления пространством отката транзакций является то, что существенно ограничивается возможность конкуренции за блоки для отката и блоки для согласованного чтения. Используя это средство, СУБД Oracle динамически регулирует количество сегментов отката (undo segments), удовлетворяющее потребностям текущей рабочей нагрузки. При необходимости она создает дополнительные сегменты отката (undo segments). В то же время, по мере необходимости сегменты отката переводятся в автономный или оперативный режимы, позволяя регулировать использование памяти в зависимости от текущей рабочей нагрузки. Все эти операции выполняются без вмешательства администратора. АБД только должен установить в параметре инициализации UNDO\_MANAGEMEN (управление пространством отката транзакций) значение AUTO (автоматическое), а в параметре UNDO\_RETENTION (сохранение информации в пространстве отката) – время выполнения самого продолжительного запроса, а также выделить достаточное пространство для табличного пространства отката. В Enterprise Manager для этого предусмотрено инструментальное средство определения размера табличного пространства отката в зависимости от времени сохранения сегментов отката.

Например, для конфигурирования сервера базы, в котором продолжительность выполнения запросов не превышает 30 минут, АБД может просто установить в параметре UNDO\_RETENTION значение, равное 30 минутам. С помощью управления временем сохранения информации сегментов отката пользователи могут успешно выполнять продолжительные запросы без возникновения неприятной ошибки ORA-1555 (snapshot too old – моментальная копия слишком стара).

Локально управляемые табличные пространства

Локально управляемые табличные пространства (locally managed tablespaces) работают лучше табличных пространств, управляемых с помощью словаря данных (dictionary managed tablespaces), их легче сопровождать и они устраняют проблемы, связанные с фрагментацией пространства. Для выполнения операций, типа выделение и освобождение пространства, в них используются битовые карты, которые хранятся в заголовках файлов данных и, в отличии от табличных пространств, управляемых с помощью словаря данных, и при работе с ними не нужно конкурировать за централизованные ресурсы. Это устраняет необходимость выполнения многих рекурсивных операций, которые иногда требуются для табличных пространств, управляемых с помощью словаря данных, таких, как выделение нового экстента.

В локально управляемых табличных пространствах для управления экстентами предлагается два режима: Auto Allocate (автоматическое выделение) и Uniform (выделение экстентов одинакового размера). В режиме Auto Allocate СУБД Oracle определяет размер каждого выделяемого экстента без вмешательства АБД, тогда как в режиме Uniform АБД указывает размер всех экстентов, выделяемых в данном табличном пространстве. Мы рекомендуем использовать режим Auto Allocate, поскольку, несмотря на то, что в этом режиме у объектов могут появиться множественные экстенты, пользователям не нужно об этом беспокоиться, так как локально управляемые табличные пространства могут поддерживать большое количество экстентов (свыше 1000 для каждого объекта), причем без заметного влияния на производительность. Поэтому мы полагаем, что АБД не будут беспокоиться о размере экстентов и использовать режим Auto Allocate.

В СУБД Oracle9i пользователи могут перемещать табличные пространства, управляемые с помощью словаря данных, в локально управляемые табличные пространства с помощью процедуры PL/SQL DBMS\_SPACE\_ADMINISTRATION.TABLESPACE\_MIGRATE\_TO\_LOCAL. Это позволяет осуществлять переход с более ранних версий СУБД Oracle и использовать преимущества локально управляемых табличных пространств. Начиная с СУБД Oracle9i Release 2 все табличные пространства (исключая табличное пространство SYSTEM) по умолчанию создаются как локально управляемые табличные пространства.

Временные табличные пространства

В СУБД Oracle для выполнения различных операций сортировки создаются временные сегменты. Использование постоянных табличных пространств для таких сегментов ведет к излишним накладным расходам по управлению пространством, аналогичным расходам по управлению пространством для постоянных данных. Хорошей практикой в таких случаях является использование специального типа табличного пространства, которое называется Temporary Tablespace (временное табличное пространство). Во временном табличном пространстве выделение пространства осуществляется посредством выборки экстентов из пула в SGA, снижая, таким образом, конкуренцию множественных операций сортировки, уменьшая накладные расходы и устраняя множественные операции управления пространством. Во временном табличном пространстве все операции сортировки разделяют один и тот же сегмент сортировки данного табличного пространства. Ясно, что это представляет собой наиболее эффективный способ работы с временными сегментами, как с точки зрения использования системных ресурсов, так и производительности системы баз данных.

Временные табличные пространства могут управляться как локально, так и с помощью словаря данных. После обсуждения преимуществ использования локально управляемых табличных пространств понятно, что предпочтительно создавать локально управляемое временное табличное пространство. Локально управляемые временные табличные пространства не требуют резервирования и могут использоваться как в базах данных только для чтения, так и в резервных базах данных. Следует отметить, оператор создания таких табличных пространств отличается от операторов создания других табличных пространств: CREATE TEMPORARY TABLESPACE <имя> TEMPFILE <имя\_файла> SIZE <размер\_файла> EXTENT MANAGEMENT LOCAL.

В СУБД Oracle9i АБД могут для всей базы данных устанавливать Default Temporary Tablespace (временное табличное пространство по умолчанию). Это делается с помощью предложения DEFAULT TEMPORARY TABLESPACE операторов CREATE DATABASE или ALTER DATABASE. Использование опции "временное табличное пространство по умолчанию" гарантирует, что всем пользователям будет назначаться временное табличное пространство. Это обеспечивает выполнение всех дисковых сортировок во временном табличном пространстве, а не в табличном пространстве SYSTEM, которое ошибочно может быть использовано для выполнения сортировок. Кроме того, опция "временное табличное пространство по умолчанию" гарантирует, что постоянные и временные объекты не будут совместно использовать одно и то же табличное пространство – еще одна хорошая практическая рекомендация.

Сопровождение сегментов

Другим вопросом, интересующим АБД, является сопровождение пространства для объектов базы данных. Начиная с СУБД Oracle9*i*, АБД получают возможность сопровождения пространства для объектов автоматически с помощью опции Automatic Segment Space Management (автоматическое управление пространством сегментов). Она упрощает задачи администрирования пространства и устраняет многие задачи оптимизации производительности, связанные с управлением пространством. Она облегчает сопровождение свободного пространства в таблицах и индексах, повышает уровень использования пространства и обеспечивает существенно лучшие "коробочные" производительность и масштабируемость.

До СУБД Oracle9*i* администрирование управления пространством осуществлялось с помощью структур данных, называемых FREELISTS (списки свободных блоков). С помощью FREELISTS СУБД следила за блоками объектов, в которых имелось достаточное свободное пространство для вставки новых строк. АБД при создании объектов определял для объектов количество списков свободных блоков и количество групп списков свободных блоков (FREELIST GROUPS), а на уровне таблиц использовался параметр PCTUSED (процент использованного пространства), с помощью которого осуществлялось управление размещением и удалением блоков из списков свободных блоков.

Новый механизм, введенный в СУБД Oracle9*i*, делает управление пространством в объектах полностью прозрачным, используя для этого битовые карты, позволяющие отслеживать использование пространства в каждом блоке данных, выделенном объекту. Состояние битовой карты показывает, сколько в данном блоке данных имеется свободного пространства (например, >75%, от 50% до 75%, от 25% to 50% или <25%), а также, является ли оно сформатированным или нет. Эта новая реализация освобождает АБД от необходимости ручного управления пространством в объектах. Другим преимуществом механизма автоматического управления пространством сегментов является повышение использования пространства в блоках данных. Причина заключается в том, что битовые карты по сравнению с FREELISTS намного лучше приспособлены для отслеживания и управления свободным пространством на уровне блоков данных. Это делает возможным улучшение повторного использования доступного свободного пространства, особенно для объектов со строками переменного размера. Кроме того, механизм Automatic Segment Space Management существенно повышает производительность конкурентных DML-операций, так как различные части битовой карты могут быть использоваться одновременно, устраняя сериализацию при поиске достаточного свободного пространства.

Улучшение производительности и сопровождаемости, обеспечиваемое механизмом Automatic Segment Space Management особенно заметно в среде Real Application Cluster. Ниже, на рис. 1, показаны результаты внутреннего тестирования, проведенного в СУБД Oracle для сравнения производительности при автоматическом и ручном управлении пространством в сегментах. Этот тест проводился в среде Real Application Cluster в двух узлах (каждый узел имел 6 x 336 Мгц ЦП, 4 Гбайта ОП), при этом в таблицу вставлялось около 3 миллионов строк. Механизм Automatic Segment Space Management обеспечил, по сравнению с оптимально настроенным ручным режимом (8 FREELIST GROUPS, 20 FREELISTS), повышение производительности более чем на 35%.

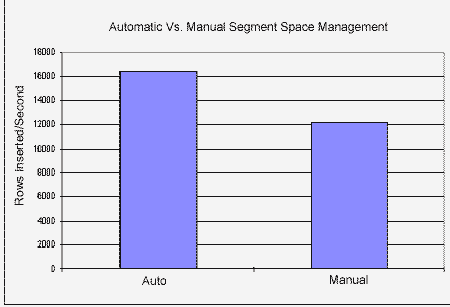


Рис. Механизм Automatic Segment Space Management обеспечивает лучшую производительность.

*Надписи на рисунке:*

* *Automatic Vs. Manual Segment Space Management* – автоматическое управление пространством сегментов по сравнению с ручным;
* *Rows Inserted/Second* – строки, вставленные в секунду;
* *Auto* – автоматическое;
* *Manual* – ручное.

Механизм Automatic Segment Space Management доступен только в локально управляемых табличных пространствах (locally managed tablespaces). В операторе CREATE TABLESPACE (создать табличное пространство) появилось новое предложение SEGMENT SPACE MANAGEMENT (управление пространством сегментов), которое имеет два параметра: AUTO (автоматическое) и MANUAL (ручное). Параметр AUTO позволяет использовать механизм Automatic Segment Space Management, а в режиме MANUAL для управления свободным пространством в объектах по-прежнему будут использоваться списки свободных блоков.

Оптимизация производительности

Плохо настроенная система – это только небольшой шаг, чтобы система стала недоступной. Настройка системы баз данных должна обеспечивать приемлемую производительность, поэтому настройка имеет у АБД высший приоритет. В СУБД Oracle9*i* появилось несколько новых функциональных возможностей, облегчающих настройку. Марк Шейман (Mark Shaiman), старший научный аналитик компании META Group, подтверждает: *“Хотя СУБД Oracle всегда обладала существенными функциональными возможностями, в общем, для достижения оптимальной производительности требовались различные "настройки". В СУБД Oracle9i автоматизированы многие вспомогательные процессы, позволяющие АБД облегчить сопровождение и обеспечить "дружественность по отношению к пользователям" ”.* В этом разделе мы обсудим ряд хороших практических методов в свете новых возможностей СУБД Oracle9*i*.

Оптимизация производительности – это не одноразовая задача, а постоянный процесс. Для эффективной настройки систем баз данных требуются хорошие методы и инструментальные средства. В комплексный процесс оптимизации производительности должны входить следующие шаги:

1. Регулярный сбор полного набора статистических показателей операционной системы, экземпляров СУБД и приложений.

АБД всегда должны иметь набор статистических показателей, собранных в конкретный момент времени, которые они должны использовать как базовый набор для сравнения. Когда они будут пытаться определить причину деградации производительности или пытаться повысить производительность, сравнение текущих статистических показателей с базовыми статистическими показателями поможет АБД быстро определить главную область приложения усилий. Это даже более важно в случае перехода на новую версию СУБД. Статистические показатели всегда следует собирать перед и после перехода на новую версию СУБД, что может позволить АБД в случае неожидаемого поведения системы определить, настолько быстро, насколько возможно, причину проблемы. Для этих целей в СУБД Oracle предусмотрена утилита StatsPack. StatsPack – это инструментальное средство для комплексного диагностирования проблем производительности на уровне СУБД и приложений. Эта утилита может быть настроена для снятия "моментальных копий" данных производительности и сохранения их в постоянных таблицах базы данных. Пользователь может задать интервал снятия моментальных копий, т.е. частоту сбора данных, а также количество собираемых статистических показателей. Более подробно об инсталляции и использовании утилиты StatsPack см. в технических документах "Diagnosing Performance Using StatsPack, Part I and II".

1. Определение проблемных областей.

Главная цель сбора статистических показателей – определить проблемы производительности сервера базы данных. Утилита StatsPack выдает легкочитаемый отчет, который указывает на потенциальные области проблем производительности. Итоговая страница типового отчета утилиты StatsPack показана на рис. 2. Этот отчет наиболее полезен, когда для сравнения с текущими показателями производительности системы имеются базовые показатели, поскольку это позволяет АБД обнаружить, что действительно изменилось в системе. Например, если бы процент использования разделяемого пула существенно увеличился, это указало бы на возможное использование в приложениях литеральных значений, приводящее к дополнительному разбору операторов SQL и, следовательно, к замедлению работы системы.

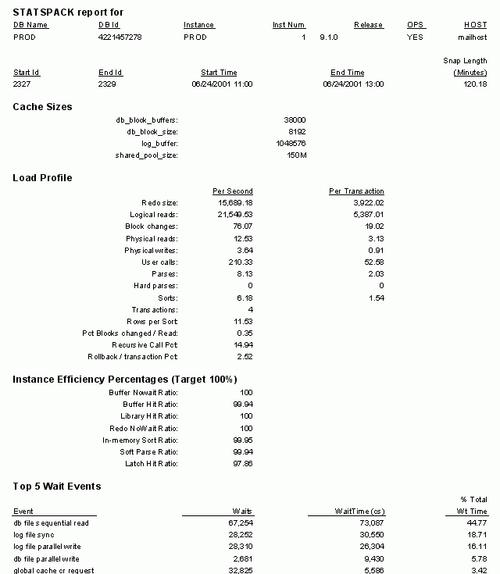


Рис. На первой странице типового отчета утилиты StatsPack на первом плане размещаются ключевые данные о системе

1. Определение потенциальных решений.

Как только проблемные области будут идентифицированы, определение потенциальных решений будет относительно простой задачей. Например предположим, на шаге (ii) мы обнаружили, что причиной появления узких мест в системе является использование в хранимых процедурах литеральных значений. Очевидным решением этой проблемы является замена в хранимых процедурах литеральных значений на переменные связывания. Если приложение очень большое, такое, как большинство ERP-приложений, это решение может оказаться нежизнеспособным. В таком случае лучшим решением было бы включение принудительного разделение курсоров с помощью установки параметра инициализации CURSOR\_SHARING=FORCE. Таким способом следует наметить в общих чертах одно или более решений для каждой идентифицированной проблемной области.

1. Реализация решений.

Золотое правило настройки системы – одновременное внесение одного изменения, а затем измерение возникших различий. Давайте предположим, что на шаге (ii) мы идентифицировали несколько проблемных областей, а на следующем шаге определили потенциальные способы решения проблем. Рекомендуемое правило заключается в реализации каждого способа по отдельности, а затем в измерении его влияния на систему. Однако требования к времени простоя системы могут не позволить воспользоваться таким правилом. В этом случае нужно попытаться реализовать только такой набор способов, которые можно проверить независимо.

1. Повторение процесса.

Как только способ решения проблем реализован, его необходимо проверить. Проверку следует осуществлять на уровне пользовательского восприятия производительности, а также повторением шага (i) и сбора дополнительных статистических показателей, позволяющих оценить, имеет ли данный способ желательное влияние или нет. Кроме того, если нагрузка на систему со временем изменяется или модифицируются приложения, весь процесс настройки следует повторить.

Настройка кеша буферов

Некоторые структуры оперативной памяти СУБД, такие, как кеш буферов, существенно влияют на производительность СУБД. Несмотря на то, что утилита StatsPack раскрывает тайны процесса настройки, важность хорошо настроенного кеша буферов заслуживает более строгого метода. С этой целью в СУБД Oracle9*i* введен механизм консультативной справки по кешу буферов (Buffer Cache Advisory). Эта справка выдается по результатам внутреннего моделирования, основанного на текущей рабочей нагрузке. Справка предсказывает частоту неудачных обращений к кешу для различных размеров кеша буферов в диапазоне от 10% до 200% текущего размера кеша. Эти предсказания публикуются через новое представление V$DB\_CACHE\_ADVICE, которое можно использовать для определения оптимального размера кеша буферов при существующей рабочей нагрузке. В Oracle Enterprise Manager имеется удобное средство, которое интерпретирует данные представления V$DB\_CACHE\_ADVICE (см. рис.). Обеспечивая детерминированный способ определения размера кеша буферов, СУБД Oracle9*i* берет на себя работу по предсказанию конфигурации кеша буферов.

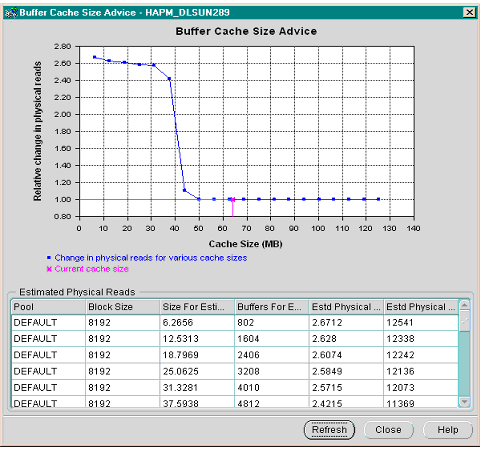


Рис. Консультативная справка по кешу буферов

АБД должны использовать эту консультативную справку для настройки кеша буферов. Все консультативные справки, включая консультативную справку по кешу буферов, и сбор статистических данных в СУБД Oracle9*i* Release 2 управляются с помощью одного параметра STATISTICS\_LEVEL (уровень сбора статистических данных). У него может быть три возможных значения: BASIC (базовый), TYPICAL (типовой) и ALL (все). Значение по умолчанию – TYPICAL. Значения TYPICAL и ALL позволяют включать в сервере механизмы создания консультативных справок: Buffer Cache Advisory (кеш буферов), Shared Pool Advisory (разделяемый пул), PGA Advisory (программная глобальная область) и MTTR Advisory (ожидаемое среднее время восстановления). При этом оказывается незначительное влияние на производительность, связанное со сбором данных и моделированием кеша. Однако выгоды от надлежащим образом настроенной памяти намного перевешивают эти накладные расходы.

Настройка разделяемого пула

Разделяемый пул, так же как и кеш буферов, представляет собой важную структуру оперативной памяти, которую необходимо тщательно настраивать для обеспечения высокой производительности сервера базы данных. Влияние настройки разделяемого пула на время реакции СУБД Oracle идентично настройке кеша буферов – облегчайте настройку, используя консультативные справки, которые основаны на внутреннем моделировании рабочей нагрузки. Статистические данные консультативной справки по разделяемому пулу (Shared Pool Advisory) позволяют отслеживать использование библиотечного кеша в памяти разделяемого пула и предсказывать изменение общего времени разбора на уровне экземпляра для различных размеров разделяемого пула. Два новых представления, V$SHARED\_POOL\_ADVICE и V$LIBRARY\_CACHE\_MEMORY содержат информацию для определения, сколько памяти используется для библиотечного кеша, сколько памяти закреплено в настоящее время, сколько объектов в LRU-списке разделяемого пула, а также сколько времени может быть потеряно или сэкономлено при изменении размера разделяемого пула. Представление V$SHARED\_POOL\_ADVICE содержит информацию о расчетной экономии времени разбора для различных размеров разделяемого пула. Эти размеры варьируются в пределах от 50% до 200% текущего размера разделяемого пула с равными интервалами изменения. В Oracle Enterprise Manager имеется средство графической визуализации этого моделирования, с которым проще работать, чем с представлением V$SHARED\_POOL\_ADVICE через интерфейс командной строки (см. рис.).

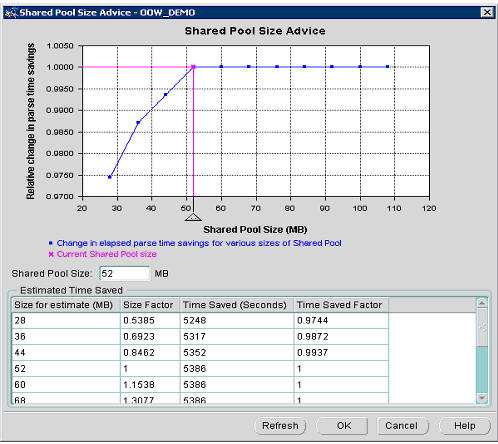


Рис. Консультативная справка по разделяемому пулу.

Консультативная справка по разделяемому пулу также управляется с помощью параметра STATISTICS\_LEVEL. Когда в параметре установлены значения TYPICAL или ALL, механизм создания этой справки включается, для его отключения следует установить значение BASIC. Как отмечено ранее, с помощью параметра STATISTICS\_LEVEL выполняется управление сбором статистических данных для всех консультативных справок. Установка в нем значения BASIC отключит все справки.

Настройка памяти для выполнения операторов SQL

Производительность выполнения сложных долго работающих запросов, типичных в DSS-среде, существенно зависит от размера памяти, доступной в программной глобальной области (PGA, Program Global Area). В СУБД Oracle8*i* администраторы устанавливают размер PGA, тщательно настраивая ряд параметров инициализации, таких, как SORT\_AREA\_SIZE, HASH\_AREA\_SIZE, BITMAP\_MERGE\_AREA\_SIZE, CREATE\_BITMAP\_AREA\_SIZE и т.д. В СУБД Oracle9*i* появилась опция полностью автоматического управления памятью PGA. Администраторам нужно просто указать максимальный объем памяти PGA, доступной экземпляру СУБД. Для этого используется новый параметр инициализации PGA\_AGGREGATE\_TARGET (суммарная память PGA). Сервер базы данных автоматически некоторым интеллектуальным способом распределяет эту память между различными активными запросами так, чтобы обеспечить максимальную производительность и наиболее эффективно использовать память. Кроме того, СУБД Oracle9*i* может сама адаптироваться к изменению рабочей нагрузки, таким образом ресурсы используются эффективно независимо от нагрузки на систему. Объем памяти PGA, доступной экземпляру, можно изменять динамически, изменяя значение параметра PGA\_AGGREGATE\_TARGET. Это позволяет оперативно добавлять или отнимать память PGA у активного экземпляра. АБД должны использовать данную опцию и не пытаться настраивать PGA вручную, поскольку машина базы данных лучше оснащена средствами определения потребностей в памяти для выполнения операторов SQL. Это относится как к повышению пропускной способности систем с большим количеством пользователей, так и к уменьшению времени ответа при выполнении запросов.

С этой самонастраивающейся PGA единственная остающаяся задача для АБД состоит в том, чтобы установить надлежащим образом значение параметра PGA\_AGGREGATE\_TARGET. Это очень просто делать с помощью новой консультативной справки по размеру PGA (PGA Advisory), введенной в СУБД Oracle9*i* Release 2. Как и консультативная справка по кешу буферов эта справка использует внутреннее моделирование для предсказания оптимального размера PGA при текущей рабочей нагрузке. В представлении V$PGA\_TARGET\_ADVICE содержатся результаты этого моделирования, показывающие надлежащее значение параметра PGA\_AGGREGATE\_TARGET. В Oracle Enterprise Manager имеется очень простое средство графической визуализации этого моделирования (см. рис.).

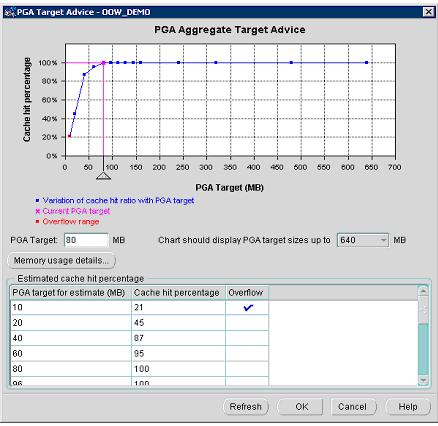


Рис. Консультативная справка по PGA.

Средство автоматического управления памятью для выполнения операторов SQL включается установкой в файле инициализации значения в параметре PGA\_AGGREGATE\_TARGET, в противном случае сервер базы данных вернется в режим ручного управления памятью PGA. Для включения механизма создания консультативной справки по PGA необходимо в параметре инициализации STATISTICS\_LEVEL установить значение TYPICAL или ALL.

Несмотря на то что повышение производительности наиболее значительным будет в DSS-среде с большой рабочей нагрузкой, это средство будет также полезно и в OLTP-среде. В большинстве OLTP-приложений требуется регулярно подготавливать сложные отчеты, производительность их подготовки может быть повышена благодаря автоматической настройке рабочих областей PGA. Следовательно, АБД должны использовать это средство совместно с консультативной справкой по PGA независимо от характера нагрузки на их системы.

Общие причины проблем производительности

Итак, мы обсудили лучшие практические методы оптимизации производительности, давайте сейчас рассмотрим методы, использование которых приводит к возникновению проблем производительности. Наша рекомендация здесь очень простая: "избегайте их".

1. Плохое управление соединениями.

Приложение соединяется и отсоединяется от базы данных при каждом взаимодействии с ней. Это – обычная проблема программного обеспечения промежуточного слоя в серверах приложений. Эта ошибка снижает производительность более чем на два порядка, а также полностью отсутствует масштабируемость. Возможное решение – трехзвенная архитектура, когда пользователи или клиенты соединяются со средним звеном, который имеет постоянные соединения с базой данных. Таким образом, одно постоянное соединение с базой данных может быть использовано разными пользователями.

1. Плохое совместное использование курсоров.

Отсутствие разделения курсоров приводит к многократным разборам. Если не используются переменные связывания, то выполняется полный разбор (hard parsing) всех операторов SQL. Это имеет существенное отрицательное влияние на производительность. В курсорах, которые открываются и повторно используются много раз, следует использовать переменные связывания. Для больших приложений, в которых не используются переменные связывания, разделение курсоров можно включить принудительно, установив параметр инициализации CURSOR\_SHARING=FORCE.

1. Неправильное конфигурирование ввода-вывода.

На многих установках базы данных неудачно размещаются на доступных дисках. На других установках неправильно задается количество дисков, так как диски конфигурируются по дисковому пространству, а не по пропускной способности ввода-вывода. Этой проблемы можно избежать, используя методологию SAME, которая была рассмотрена в начале этой статьи.

1. Проблемы конфигурирования журнала базы данных.

На многих установках слишком мало журнальных файлов и размер их слишком мал. Небольшие журнальные файлы являются причиной частых переключений журнальных файлов, которые могут существенно нагрузить кеш буферов и систему ввода-вывода. Если журнальных файлов слишком мало, процесс архивирования может не справляться со своей работой и работа сервера базы данных будет приостанавливаться. Общее правило: базы данных должны иметь по крайней мере 3 журнальных группы с двумя журнальными файлами в каждой.

1. Нехватка списков свободных блоков, групп списков свободных блоков и сегментов отката.

В кеше буферов может возникать сериализация блоков данных из-за нехватки списков свободных блоков (free lists), групп списков свободных блоков (free list groups), участков транзакций (INITRANS) или сегментов отката (rollback segments). Особенно часто это возникает в приложениях, часто выполняющих операции вставки данных в базу данных с большим размером блоков (от 8Kб до 16Kб). Этой проблемы легко избежать, используя опцию автоматического управления пространством сегментов (Automatic Segment Space Management) вместе с опцией автоматического управления откатом (Automatic Undo Management).

1. Длительные полные просмотры таблиц.

Длительные полные просмотры таблиц в операциях большого объема или в интерактивных операциях, выполняемых в оперативном режиме, могут указывать на неудовлетворительное проектирование транзакций, отсутствие индексов или плохую оптимизацию операторов SQL. Длительные полные просмотры таблиц по своей природе являются операциями, интенсивными по вводу-выводу и немасштабируемыми при увеличении количества пользователей. В пакете настройки (tuning pack) Enterprise Manager предлагается мощный инструмент, SQL-анализатор (SQL Analyze), для обнаружения и оптимизации операторов SQL, интенсивно потребляющих ресурсы. Это – хороший способ решения проблем оптимизации операторов SQL.

1. Дисковые сортировки.

Дисковые сортировки, в противоположность сортировкам в оперативной памяти, могут в операциях, выполняемых в оперативном режиме, указывать на неудовлетворительное проектирование транзакций, отсутствие индексов, неоптимальное конфигурирование PGA или плохую оптимизацию операторов SQL. Дисковые сортировки по своей природе являются операциями, интенсивными по вводу-выводу и немасштабируемыми. Эту проблему можно устранить, выделяя память PGA достаточного размера и используя опцию автоматической настройки PGA (Automatic PGA Tuning), рассмотренную выше.

1. Большие объемы рекурсивных операций SQL.

Большие объемы рекурсивных операций SQL, выполняемых в схеме SYS, могут указывать, что имеют место действия по управлению пространством, такие, как выделение экстентов. Эти действия немасштабируемы и влияют на время реакции системы. Обычно это возникает при использовании табличных пространств, управляемых с помощью словаря данных, (dictionary managed tablespaces). В таких случаях повышению производительности может помочь использование локально управляемых табличных пространств (locally managed tablespaces), существенно уменьшающих объемы рекурсивных операций SQL.

1. Ошибки схем и оптимизатор.

Во многих случаях в приложении используются чрезмерные ресурсы из-за того, что схема, которой принадлежат ресурсы, была неправильно перенесена из среды разработки или из более старой среды промышленной эксплуатации. Примеры: отсутствие индексов или некорректные статистические данные. Эти ошибки могут привести к появлению неоптимальных планов выполнения и низкой производительности интерактивной работы пользователей. При переносе приложений с известной производительностью для поддержи стабильности планов экспортируйте статистические данные схемы с помощью пакета DBMS\_STATS. Более того, набор параметров оптимизатора, установленный в файле параметров инициализации, может привести к изменению испытанных оптимальных планов выполнения. По этим причинам для обеспечения устойчивой производительности статистическими данными схемы и параметрами оптимизатора следует управлять вместе как группой.

1. Использование нестандартных параметров инициализации.

Они могли быть установлены на основании плохого совета, неправильного предположения или предшествующей потребности. В частности, параметры, связанные со счетчиком циклических запросов защелок (spin\_count ) и недокументированными возможностями оптимизатора, могут привести к возникновению большого количества проблем, которые могут потребовать значительного исследования.

Резервирование и восстановление

Просто говоря, здесь могут происходить плохие дела. Может быть полный отказ систем, файлы могут ошибочно удаляться, могут происходить сбои носителя. АБД, у которого нет обоснованной стратегии резервирования и восстановления для разрешения таких ситуаций, должен иметь под рукой свое резюме о поступлении на работу. В этом разделе мы кратко рассмотрим некоторые процедуры и практические методы, которые помогут АБД сохранить свою работу.

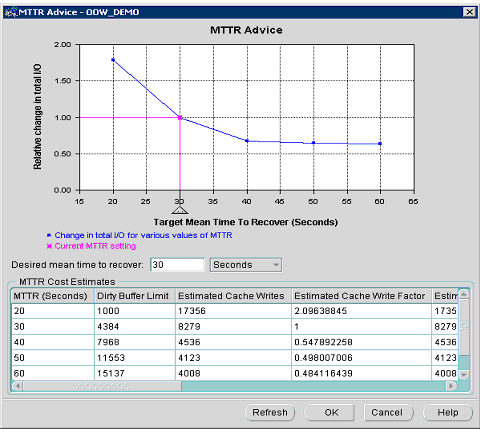
У бизнеса должна быть стратегия восстановления после катастроф, которые приводят к потере данных. В настоящее время только 45% компаний из списка крупнейших компаний мира (Fortune 500) имеют формальные стратегии для своей защиты от бедственной потери данных. Только 12% из этих стратегий рассматриваются на корпоративном уровне ("Calculating the Cost of Downtime" (вычисление стоимости времени простоя), Анжела Карр (Angela Karr), Advanstar Communications). При разработке формальной стратегии АБД должны планировать потерю данных из-за неожиданных отказов, причинами которых могут быть аппаратные или программные сбои, человеческие ошибки, природные явления и т.д. Для каждого из таких возможных событий АБД должны иметь план действий. В некоторых случаях может потребоваться восстановление носителя, в других будет достаточно восстановления экземпляра. Для обеих этих ситуаций АБД должны иметь планы действий. Мы рассмотрим их в этом разделе ниже. В конце концов, администратор базы данных и системный администратор должны документировать среду базы данных и стратегии восстановления, чтобы при возникновении сбоя АБД мог диагностировать проблему и немедленно заняться ее решением.

Планирование восстановления экземпляра

Работа систем может аварийно завершаться по разным причинам, например, из-за прекращения подачи электроэнергии. В этом случае после перезапуска экземпляра СУБД и до открытия базы данных для пользователей будет выполняться восстановление экземпляра. Цель восстановления экземпляра – восстановление физического содержимого базы данных в транзакционно согласованное состояние. Поэтому во время восстановления должны применяться все сделанные зафиксированными транзакциями изменения, которые находились в кеше буферов и не были записаны на диск до сбоя, а затем выполняется откат всех записанных на диск изменений, сделанных незафиксированными транзакциями. Восстановление экземпляра выполняется внутренними механизмами СУБД Oracle и не требует вмешательства администратора. Единственный аспект, о котором должен заботиться АБД – продолжительность выполнения восстановления; это легко контролируется настройкой в файле инициализации параметра FAST\_START\_MTTR\_TARGET (MTTR – mean time to recover, среднее время восстановления). В системах, в которых размер кеша буферов составляет много гигабайтов, восстановление экземпляра может быть очень длительным с совершенно не предсказуемым временем восстановления. Для АБД оба эти свойства мало допустимы. Новая опция быстрого восстановления с предсказуемым временем (Fast-Start Time-Based Recovery) в СУБД Oracle9*i* позволяет АБД задавать среднее время восстановления экземпляра (в секундах) с помощью установки значения в параметре FAST\_START\_MTTR\_TARGET. Эта опция реализуется ограничением количества "грязных" буферов и количества журнальных записей, сгенерированных после последней контрольной точки. Для обеспечения заданного времени восстановления сервер базы данных по существу регулирует скорость выполнения контрольных точек.

Если в параметре FAST\_START\_MTTR\_TARGET установлено слишком низкое значение, сервер базы данных будет писать на диск очень часто, и это может привести к снижению скорости работы системы. Поэтому этот параметр необходимо настраивать так, чтобы сервер не выполнял чрезмерное количество контрольных точек, но при этом время восстановления оставалось бы в приемлемых пределах. С этой целью в СУБД Oracle9*i* Release 2 введена новая консультативная справка по MTTR (MTTR Advisory). Эта справка позволяет экземпляру оценивать (в процентах) изменение количества записей на диск, которое могло бы произойти при других значениях в параметре FAST\_START\_MTTR\_TARGET. Используя оценку, полученную при типичной рабочей нагрузке на экземпляр, вы можете определить влияние повышения или понижения значения в данном параметре. Значения консультативной справки по MTTR содержатся в представлении V$MTTR\_TARGET\_ADVICE. Это представление имеет пять строк, которые показывают предполагаемую дисковую активность при различных значениях в параметре FAST\_START\_MTTR\_TARGET: текущее значение; приблизительно десятая часть текущего значения; приблизительно половина текущего значения; приблизительно полтора текущего значения и приблизительно двойное текущее значение. В Enterprise Manager можно получить графическое представление консультативной справки по MTTR (см. рис. 6).

Теперь АБД имеют инструментарий для управления временем восстановления экземпляра и могут это делать без компрометирования производительности системы. Они должны использовать этот инструментарий, поскольку он позволяет сделать время восстановления экземпляра предсказуемым, а также дает ручки настройки этого времени в соответствии с их потребностями.



Консультативная справка по MTTR

Планирование восстановления носителя

Стратегия резервирования

Восстановление носителя требуется всегда, когда теряются данные. Наиболее общие причины потери данных: сбои носителя и человеческие ошибки. Для того чтобы можно было восстанавливать носитель, сервер базы данных должен функционировать в режиме ARCHIVELOG (архивирование журнала). Его легко включить с помощью оператора ALTER DATABASE. Затем АБД должны выбрать метод резервирования. В прошлом АБД писали свои собственные скрипты для резервирования базы данных. Сейчас делать это уже нецелесообразно, так как утилита Recovery Manager (диспетчер восстановления) лучше оснащена для управления резервированием. Утилита Recovery Manager имеет вариант с интерфейсом командной стоки, а также вариант с графическим интерфейсом, встроенным в Enterprise Manager. Утилита Recovery Manager позволяет пользователям специфицировать все их требования резервирования, например, полное оперативное резервирование базы данных на ленту, автономное резервирование всей базы данных на диск и т.п. Более того, использование средств планирования в Enterprise Manager позволяет запускать задания создания резервных копий в заданное время, как регулярно, так и по потребности. Хорошая практическая рекомендация: вместе с резервными копиями файлов данных базы данных создавать резервную копию управляющего файла. Резервную копию управляющего файла, как минимум, необходимо создавать при изменении структуры базы данных.

Для тех администраторов, которым требуются также файлы систем резервирования, корпорация Oracle инициировала, совместно с поставщиками систем управления носителями, программу решения проблем резервирования. Участники этой программы – сторонние поставщики, инструментальные средства которых обеспечивают комплексное решение вопросов резервирования как системных файлов, так и файлов базы данных Oracle. Это лучше использования для управления резервированием двух отдельных инструментов, Recovery Manager или Enterprise Manager для файлов базы данных Oracle и отдельного инструмента для системных файлов.

Как обсуждалось ранее, журнальные файлы следует мультиплексировать. Это означает, что каждая журнальная группа будет иметь два журнальных файла или элемента. Настоятельно рекомендуется, чтобы задание создания резервных копий резервировало по крайней мере два элемента каждой журнальной группы. Это подчеркивает важность журнальных файлов, поскольку если во время восстановления возникнут проблемы с архивными журнальными файлами из-за их физического повреждения или человеческой ошибки, то полное восстановление базы данных будет невозможным, и это приведет к потере некоторых данных. Вероятность такого происшествия существенно уменьшается, если есть запасная резервная копия. Заметим, АБД не должны создавать резервную копию журнального файла, а затем резервировать ее еще один раз. Вместо этого они должны резервировать два зеркальных журнальных файла по отдельности, так чтобы в случае повреждения одного из них для восстановления был доступен второй журнальный файл. Если обе копии являются копиями одного и того же архивного журнального файла, то мы не защищаем себя от повреждений журнала.

Стратегия восстановления

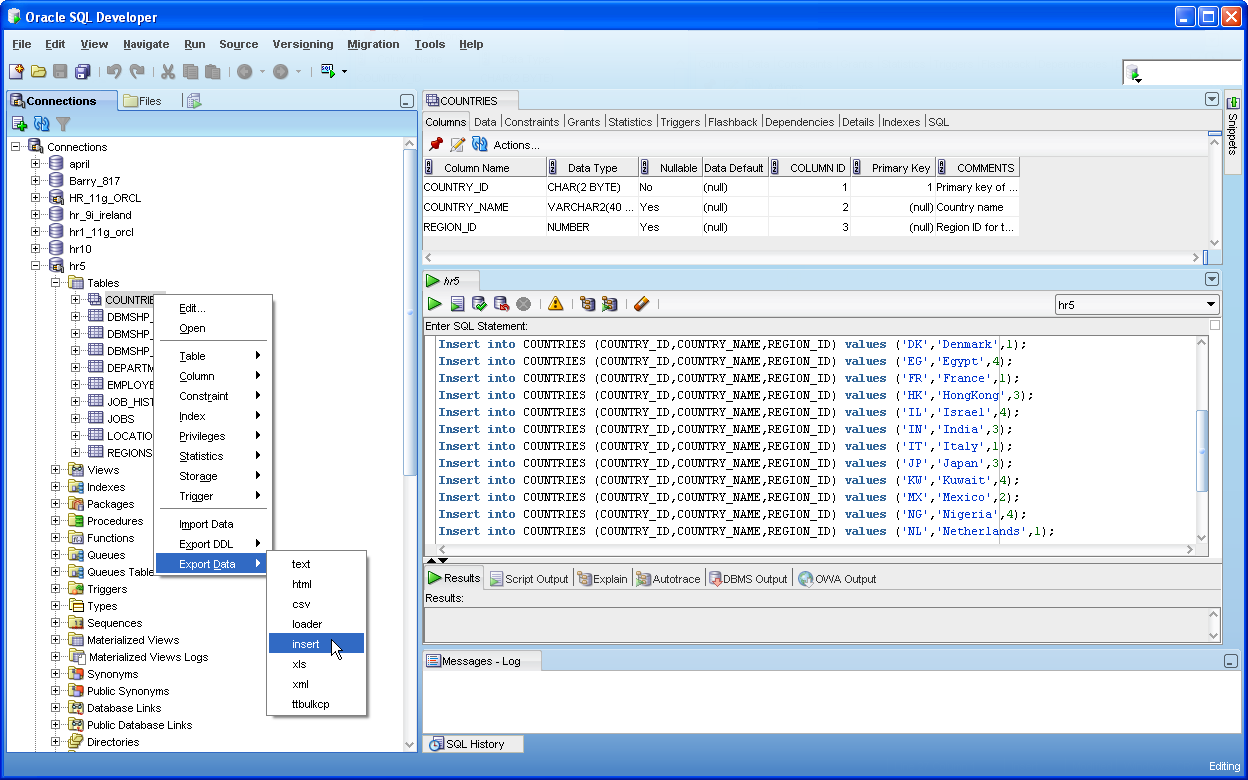
АБД должны моделировать сценарии потери данных и тестировать свои планы восстановления. К сожалению, большинство АБД не тестируют свои стратегии резервирования и восстановления. Проведение испытаний плана восстановления помогает проверить процессы на месте, а также держит администраторов в курсе проблем и ловушек восстановления. Четыре основные причины потери данных: сбои носителя, человеческие ошибки, повреждение блоков данных и стихийные бедствия. План восстановления необходимо разрабатывать с учетом каждого случая.

1. *Сбой носителя.* Обычный пример сбоя носителя – разрушение дисковых головок, которое приводит к потере всех файлов базы данных, размещенных на диске. Для полного отказа диска уязвимы все файлы, связанные с базой данных, включая файлы данных, управляющие файлы, оперативные и архивные журнальные файлы. Восстановление после сбоя носителя включает в себя копирование всех неисправных файлов из резервной копии, а затем их восстановление. В наихудшем случае может потребоваться копирование и восстановление всей базы данных. Сбой носителя – первостепенная задача стратегии резервирования и восстановления, поскольку во время восстановления обычно требуется копировать некоторые или все файлы базы данных, а затем применять для их восстановления журнальные файлы.
2. *Человеческая ошибка.* Пользователи могут непредумышленно удалять данные или уничтожать таблицы. Хороший способ минимизации последствий таких ситуаций – предоставлять пользователям только такие привилегии базы данных, которые им безусловно необходимы. Но человеческие ошибки будут по-прежнему возникать и АБД должны быть готовы к работе над ними. Создание логических резервных копий или экспорт таблиц – хороший способ справляться со случайными уничтожениями таблиц. Ретроспективный запрос (Flashback Query), новое средство, введенное в СУБД Oracle9*i*, очень хорошо подходит для восстановления после случайных удалений или вставок. Последним средством, к которому можно прибегнуть, является копирование из резервной копии и восстановление всего табличного пространства, содержащее объекты, на которые подействовала человеческая ошибка. Существуют различные методы восстановления после человеческих ошибок, и все они должны быть протестированы.
3. *Повреждение блоков данных.* Неисправная аппаратура или ошибка операционной системы могут быть причиной повреждения блоков данных, в результате которого их формат не будет распознаваться СУБД Oracle или их содержимое будет внутренне несогласованным. В СУБД Oracle имеется средство восстановления носителя на уровне блоков (Block Media Recovery), которое копирует и восстанавливает только поврежденные блоки, и не требует восстановления всего файла данных. Это уменьшает среднее время восстановления (MTTR, Mean Time to Recovery), так как копируются и восстанавливаются только те блоки, которые требуется восстановить, а испорченные файлы данных остаются в оперативном режиме.
4. *Стихийные бедствия.* Землетрясения и наводнения могут привести к полному разрушению оборудования информационного центра. В таких случаях требуется копирование из резервных копий и восстановление на альтернативных установках. Это неизбежно влечет за собой необходимость поддержки резервных копий не только на основной установке, но и на альтернативной. "Внешнее" копирование и восстановление следует тщательно тестировать и планировать по времени, чтобы бизнес мог выдержать такие стихийные бедствия без компрометирования своих соглашений об уровнях сервиса.

Тщательно тестируя и документируя свой план восстановления, АБД могут существенно ограничить неожидаемые перерывы в работе и повысить доступность систем баз данных.

Заключение

Бизнес пришел к оценке своих ИТ-инфраструктур как стратегического императива. Учитывая, что системы управления базами данных являются ядром ИТ-инфраструктуры, их надежность и доступность имеют принципиальную важность для бизнеса. В этой работе мы наметили в общих чертах некоторые ключевые практические методы в области конфигурирования систем баз данных, их производительности, управления и восстановления, которые существенно повышают надежность и доступность баз данных, а также уменьшают для бизнеса эксплуатационные и административные расходы. Даже минимальное время простоя системы базы данных становится недопустимым для увеличивающегося числа компаний, кроме того, в связи с ростом мощности систем баз данных и расширением их функциональных возможностей, нагрузка на корпоративных АБД в недавнем прошлом росла экспоненциально. Мы надеемся, что эта работа устранит часть этой нагрузки и в то же самое время повысит эффективность их систем баз данных.



Среда Oracle SQL Developer для работы с базой данныз Oracle Database

# 12. Распределенные приложения в сети Интернет.

Распределённая система - система нескольких автономных вычислительных узлов, взаимодействующих для выполнения общей цели.

Распределённая система - система, чьи компоненты размещены на различных узлах взаимодействующие и управляемые только посредством передачи сообщений.  
Существуют системы с разделяемой памятью (или с разделяемым временем).

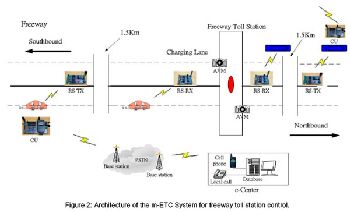
Распределённая система - “это набор независимых узлов (компьютеров), которые представляются пользователю как один компьютер.” [Tanenbaum]

Распределённая система - ”Система, состоящая из набора двух или более независимых узлов которые координируют свою работу посредством синхронного или асинхронного обмена сообщениями.”

Распределённая система - “это собрание независимых компьютеров соединенных сетью с программным обеспечением, обеспечивающим их совместное функционирование.”

Распределённые приложения в сети Интернет базируются на Клиент-серверной архитектуре и Клиент-Клиент.

Клиент-серверная архитектура

Архитектура клиент/сервер (двухуровневая – two-tier) предполагает разделение задачи между сервером (хранит, обрабатывает и поставляет данные) и клиентом (которому необходимы эти данные). В настоящее время эта архитектура вытесняется распределенными приложениями (distributed application).

Распределенное приложение обеспечивает совместное использование программного обеспечения различными пользователями. Бухгалтерское приложение для небольшой компании можно разделить между владельцем и его ассистентом. В больших организациях торговли по почте доступ к центральному приложению оформления заказа распределяется между сотнями операторов, общающихся с клиентами по телефону.

В традиционной архитектуре клиент/сервер обе части системы представляют собой специализированные приложения. Например, можно написать на Visual Basic серверное приложение, отсылающее по факсу заказ на обед в местный ресторан. Затем создать еще одно приложение (другой проект на Visual Basic), позволяющее сотрудникам обращаться к приложению заказа обеда и отсылать соответствующий запрос на сервер.

В этом случае возникает проблема с внедрением клиентской части системы на всех компьютерах компании. Потребуются программа установки приложения и инструкция по сетевой установке клиентской части. После того как коллеги установят клиентскую часть приложения, начнутся телефонные звонки, связанные с неправильной установкой и невозможностью запуска приложения.

Web-среда устраняет головоломки с установкой и настройкой клиентской части. Поскольку все сотрудники компании уже имеют web-браузеры, можно использовать возможности сервера Personal Web Server или Internet Information Server, размещенного на сетевом сервере.

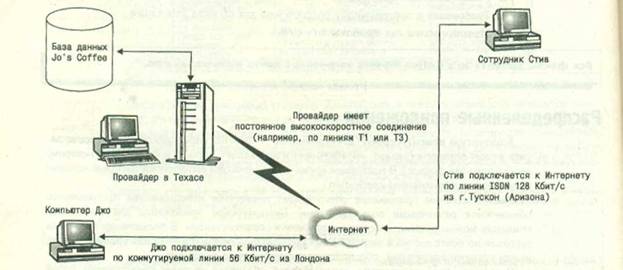
Придется переработать серверную часть приложения в набор бизнес-объектов и создать страницы ASP, которые смогут показать пользователям сети меню ресторана и сформировать заказ по факсу. Возможно, все приложение сконцентрируется на сервере, однако распространить сведения о нем поможет рассылка рекламного сообщения по электронной почте. Реализация доступа к web-странице интрасети гораздо проще установки клиентской части на всех компьютерах сети. Кроме того, устраняются проблемы с внедрением приложения на стороне клиента.

Однако это еще не все. Предположим, что сервер компании обеспечивает доступ по Интернету, поэтому сотрудники смогут заказать меню даже извне, используя обычный web-браузер. Скорее всего, распределенному приложению никогда не понадобится специализированная клиентская часть на Visual Basic. Пока мы не затрагиваем вопросы безопасности доступа к приложению из web-браузера.

Итак, наше приложение будет "распределенным приложением, использующим web-браузер в качестве уровня представления". Размещенное на web-сервере приложение станет прикладным сервером (application server), который пользуется технологией ASP или чем-то похожим. Любой сотрудник, имеющий web-браузер, будет потенциальным пользователем приложения, причем для доступа уже не понадобится специализированная клиентская программа.

Подключение к распределенному приложению

Теперь посмотрим, как посетители joscoffee.com смогут использовать распределенное приложение.



До появления Интернета трудно было представить себе приложение в Техасе, с которым работают пользователи из Лондона и Аризоны. Хотя технологии Интернета известны уже почти два десятка лет, их практическое применение для распределенного приложения продажи кофе было нереальным.

Интернет — гигантская всемирная компьютерная сеть, объединяющая десятки тысяч сетей всего мира. Ее назначение — обеспечить любому желающему постоянный доступ к любой информации. Интернет предлагает практически неограниченные информационные ресурсы, полезные сведения, обучающие программы, развлечения, возможность общения с компетентными людьми, услуги удаленного доступа, передачи файлов, электронной почты и многое другое.

Интернет обеспечивает принципиально новый способ общения людей, не имеющий аналогов в мире.

Благодаря сети Интернет стал доступен (бесплатно или за умеренную плату) огромный объем информации. Так, пользователь в любой стране может связаться с людьми, разделяющими его интересы, или получить ценные сведения в электронных библиотеках, даже если они находятся на другом конце света. Нужная информация окажется в его компьютере за считанные секунды, пройдя путь по длинной цепочке промежуточных компьютеров, по кабелям и радиоканалам.

Сети в Интернете неограниченно коммутируются (т.е. связываются) друг с другом, потому что все компьютеры, участвующие в передаче данных, используют единый протокол коммуникации TCP/IP (ти-си-пи/ай-пи).

Обычно Интернет как пример распред-й системы характеризуют как "гигантская сеть компьютеров", решающих единую распределенную задачу. Потрясающие возможности Интернета связаны с тем, что компьютерные системы могут подключаться и взаимодействовать такими способами, которые ранее были невозможны или экономически невыгодны.

Откроем web-браузер, подключимся к Amazon.com и подумаем, что происходит в действительности. Можно сказать: "Мы просматриваем web-страницу". Но это лишь половина истории. В реальности выполняется расширение распределенной компьютерной системы Amazon.com на персональный компьютер (ПК) пользователя. На определенном уровне компьютерная система этой компании перестает быть автономной системой центральной обработки данных и начинает объединять тысячи компьютеров посетителей сайта, плюс компьютеры поставщиков товаров, плюс компьютеры поставщиков этих поставщиков и т. д.

Интернет реализует быстрое подключение коммерческих компаний, организаций и частных лиц, отключение, переподключение и прочие операции с различными распределенными приложениями на основе web-технологий (такими, как сайты электронной коммерции, аукционы, страницы телевизионных программ).

Джо в Лондоне, так же как и его коллега Стив в Аризоне, становятся частью распределенного приложения, которое размещено и работает на компьютере провайдера в Техасе. Раньше для подключения к приложению пришлось бы пользоваться телефонным звонком, создающим постоянное соединение между сетями British Telecom и AT&T. Однако современная Интернет, как глобальный коммутатор с миллиардами шлюзов и узлов, позволяет установить и обслужить соединение между Лондоном и Техасом за миллисекунды, устраняя все существовавшие ранее барьеры.

Попадая на страницу Amazon.com, помните, что вы подключились к массивному распределенному приложению, призванному продавать книги, CD, видео и другие товары!

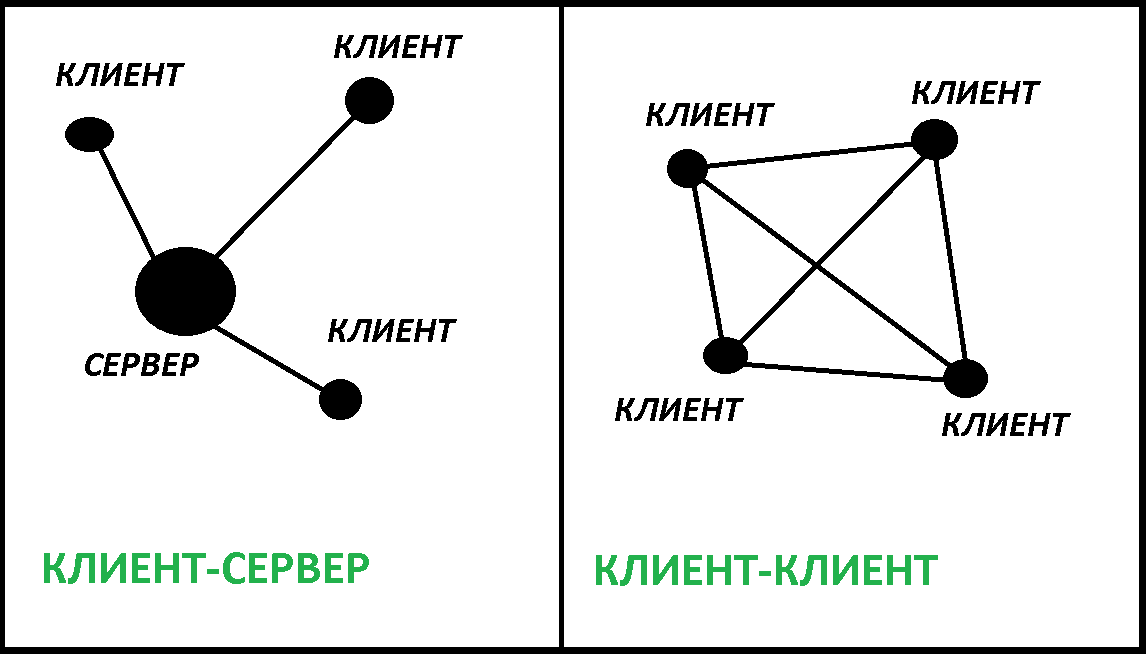
Децентрализованные блокчейн-приложения

Технология блокчейн в действии обладает важным свойством – децентрализацией. Оно активно внедряется в повседневную жизнь: уже существуют бета-версии децентрализованного интернета (Substratum), децентрализованные анонимные мессенджеры (DigitalNote), децентрализованные социальные сети (Golos), блокчейн-кошельки. Естественно, популярность набирают децентрализованные приложения (DApps). Под приложением понимается программа, установленная на компьютере или смартфоне. Рассмотрим, чем же децентрализованные блокчейн-приложения отличаются от традиционных, как они взаимодействуют с умными контрактами, и вспомним историю их развития.

Пиринговая сеть (P2P) как основа

Классические программные приложения используют взаимодействие вида «клиент-сервер». К примеру, такую архитектуру имеет Google, Amazon, голосовой поиск VoiceSearch, приложение для распознавания музыки Shazam и большинство существующих на данный момент web-приложений - ПО, взаимодействующих с интернетом. Архитектура ПО определяет, какие элементы будут присутствовать в приложении и каким образом они будут взаимодействовать друг с другом.

В программах с шаблоном архитектуры “клиент-сервер” клиентов много, а управляющий сервер один. В данном случае клиентами являются ПК, смартфоны и ноутбуки. Сервер представляет собой специализированный компьютер, который хранит огромный объем информации, направляет ее и проводит вычисления, через него проходят все данные. Главным недостатком клиент-серверной архитектуры является централизация.



Архитектура приложения «клиент-клиент» чаще называется пиринговой (от англ. peer-to-peer, или p2p). В таком приложении каждый пользователь сочетает в себе функции клиента и сервера одновременно. Также он может связаться с другим пользователем напрямую. Первое пиринговое приложение появилось в 1999 году. Им стала Napster − всемирная платформа для бесплатного обмена файлами. Известным приложением, использующим пиринговую сеть, является Ace Stream P2P Multimedia. Технология пиринговых сетей используется и в блокчейн-приложениях.

Пиринговая сеть стала основой для разработки полностью децентрализованных распределенных приложений.

1. Распределенные web-приложения проводят вычисления на нескольких узлах. К примеру, в сети Bitcoin необходимо проводить криптографические вычисления по алгоритму SHA-256. Такую операцию могут осуществлять следующие узлы: центральные процессоры компьютеров (CPU), графические процессоры видеокарт (GPU) и специализированные ASIC-майнеры.
2. Децентрализованные приложения (DApps) не имеют узлов, которые управляют другими узлами. Все участники децентрализованной системы равноправны.

Очень важно понять, что распределенное приложение может быть централизованным, а вот децентрализованное приложение всегда является распределенным. К примеру, тот же ВКонтакте использует множество серверов для распределения вычислений и хранения данных, но все эти серверы подконтрольны главному. Следовательно, ВКонтакте является распределенным, но не децентрализованным приложением.

Существует еще одна разновидность децентрализованных приложений – частично децентрализованные (гибридные). Такими являются uTorrent и The Onion Router. Они используют сервера, но надежность сети обеспечивают за счет p2p-соединения.

Преимущества децентрализованных приложений

Стоит отметить, что децентрализованные программы входили в список самых скачиваемых приложений еще задолго до появления Bitcoin. Всем известный клиент uTorrent является примером успешного частично децентрализованного приложения.



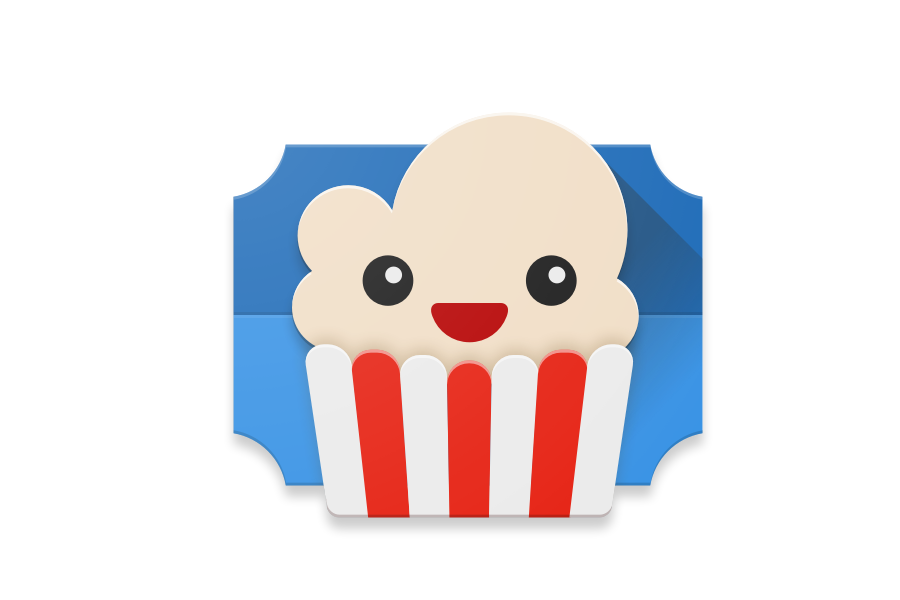
Протокол BitTorrent решил проблему загрузки объемных данных объединением всех желающих скачать большие файлы быстро и бесплатно. Сеть устроена так, что все клиенты, загружающие файлы, превращаются в раздающих, а само скачивание файла идет не из одного источника, а сразу из нескольких. Поощрение раздачи файлов заключается в том, что все активные раздающие получают бонусы в качестве добавления скорости во время собственных скачиваний. Но разработчики uTorrent не открыли исходный код своего приложения для того, чтобы на нем не смогли нажиться другие программисты.

Для успешной разработки децентрализованных блокчейн-приложений необходимо соблюдать следующие правила:

1. Открытый исходный код. Данное свойство ставит приложение в более выгодную позицию, так как пользователи могут проверить его. Открытый код позволяет разработчикам улучшать приложение, создавая новые доработанные версии. Казалось бы, таким образом можно просто создать новое приложение и забрать прибыль у истинных изобретателей. Но на деле выстреливают только те проекты, которые решают проблемы их предшественников.
2. Мотивация разработчиков и пользователей. Традиционные приложения приносят своим разработчикам прибыль за размещение рекламы, покупку платных услуг или комиссий за транзакции. В приложении с открытым кодом данные функции можно убрать, изменив исходный код. Как же в таком случае получить прибыль за разработку блокчейн приложений? В решении этого вопроса помогают дефицитные токены – внутренняя валюта приложения. Разработчики изначально выдают себе или добывают какое-то количество токенов. Пользователи могут добывать монеты за поддержку приложения, различного рода активность, предоставление вычислительных ресурсов. Чем больше участников подключается к использованию приложения, тем сильнее растет спрос на токены. Как следствие, монета становится более ценной. Хорошим примером являются игры для биткойн-добычи CoinBrawl, MyBitMine, BitCasino. Децентрализованный аналог нашумевшей игры Mineсraft реализовали на платформе Ehthereum – Etheria.
3. Децентрализованный консенсус. Допустим, пользователь залез в код майнинг-игры и добавил себе какое-то количество сатошей. Затем он сохранил код, зашел в игру, но количество сатошей осталось прежним. Дело в цепочке блоков, взломать которую можно, обладая только огромными вычислительными мощностями. Внести изменения в игровой баланс можно единственным легитимным способом − выполнить определенные игровые условия. Так как условия не были выполнены, баланс не изменился. Проверка условий осуществляется с помощью смарт-контрактов. Умный контракт прописан в коде приложения и не может быть изменен. Как внедрить смарт-контракт в свое приложение, рассмотрим в следующих материалах.
4. Отсутствие точки отказа. Приложение не прекратит работу, пока будет существовать хотя бы один узел.

Популярные децентрализованные приложения

Протокол BitTorrent был создан в 2004 году. Он был призван улучшить уже существующие на тот момент приложения Napster и Gnutella. Как известно, в результате получились такие гиганты, как uTorrent и NetFlix. На протоколе BitTorrent также функционирует PopcornTime – приложение, которое позволяет передавать потоковое видео в режиме реального времени.



Еще одним ярким представителем DApps является децентрализованная версия Ebay – OpenBazaar.

Приложение позволяет покупать товары без посредников, расплачиваясь криптовалютами.

Политические события 2014 года в Китае подтолкнули сообщество к созданию приложения, позволяющего смартфонам с операционной системой iOS 7 связываться между собой без посредников. Так появился децентрализованный мессенджер FireChat. Аналогами являются EthetTweet, DigitalNote, Gems.

Приложение Lighthouse сочетает в себе платформу для работы со смарт-контрактами и блокчейн-кошелек Bitcoin. Хороший пример некоммерческого продукта. Создавать собственные блокчейн-приложения позволяет СoinbaseToshi для сети биткойн.

Lazooz является отличным примером децентрализованного блокчейн-приложения. Токены Zooz можно майнить, катаясь на машине, а расплачиваться ими - за услуги такси.

В следующих материалах мы разберем, какую платформу выбрать для разработки собственного децентрализованного блокчейн-приложения.

# 13. Сетевые протоколы прикладного уровня: ftp, smtp, http, telnet, gopher, nntp, pop3, imap.

В соответствии с архитектурой клиент-сервер, программа делиться на две части (одна работает на сервера, вторая – на компьютере пользователя), функционирующие как единое целое.

Протоколы прикладного уровня описывают взаимодействие между клиентской и серверной частями программы.

*1. Протокол HTTP* (Hyper Text Transfer Protocol) – протокол передачи гипертекста, работающий на 80 порту. При этом каждая HTML-страница загружается отдельно, причем соединение между загрузками прерывается и никакой информации о соединении не сохраняется. Это сделано для того, чтобы каждый из пользователей Web-страниц получал информацию «в порядке общей очереди». В противном случае могла бы создаться ситуация, например, когда один пользователь начинает качать страницу с большим содержанием рисунков высокого разрешения, а все остальные ждут, пока он это закончит.

2. *Протокол FTP* (File Transfer Protocol) – протокол передачи файлов, работающий на 20 и 21 порту. Он предназначен для копирования файлов между компьютерами. Полностью занимает канал, пока не будет получен файл, а далее сохраняет информацию о соединении. При сбое возможна докачка с того места, где произошел сбой.

3. *Протоколы STMP, IMAP-4, POP3* – почтовые протоколы (электронная почта), работающие, SMTP – на 25 порту, IMAP-4 – на 143 порту, POP3 – на 110 порту. Отличие данных протоколов состоит в том, что протокол STMP предназначен на доставку почты до конкретного получателя, а протоколы IMAP-4 и POP3 – протоколы взаимодействия пользователя со своим почтовым ящиков на сервере..

При использовании STMP предполагается, что почтовый адрес указывает на компьютер конечного получателя, и на этом компьютере запущена специальная программа, которая принимает и обрабатывает почту.

Однако чаще всего бывает, что почта не доставляется на компьютер каждого отдельного пользователя, обрабатывается централизованно, на отдельном почтовом сервере. В таком случае, каждый пользователь имеет на почтовом сервере свой почтовый ящик. Почта доставляется до сервера по протоколу STMP (конечный получатель – сервер) и перемещается в почтовые ящики пользователей. Затем пользователи подключаются к своим почтовым ящикам по протоколу POP3 или IMAP-4 и забирают почту.

Таким образом, наиболее распространенный вариант работы с почтой для обычного пользователя состоит в следующем: отправка почты по протоколу STMP (на почтовый сервер получателя), получение почты – по протоколу POP3 или IMAP-4 (скачивание почты из почтового ящика на своем почтовом сервере).

4. *Протокол TELNET* – используется для подключения и управления удаленным компьютером, работает на 23 порту.

После подключения, каждый символ, введенный на локальной машине, обрабатывается так, как если бы он был введен на удаленной машине. Либо может использоваться командный режим – управление удаленной машиной при помощи специальных команд. Фактически TELNET – это протокол эмуляции терминала: при помощи TELNET можно подключиться, например, на 25 порт и вручную набрать все необходимые поля заголовка письма, изменив адрес отправителя (обычно эти поля заполняются автоматически специальными почтовыми программами) и отправить само письмо. Или, например, подключиться на 80 порт и «поиграть» роль Web-браузера Internet Explorer.

POP3

POP3 — протокол для приема сообщений электронной почты.

При просмотре почты с использованием протокола POP3 все электронные письма скачиваются пользователю на компьютер и автоматически удаляются с сервера (хотя это можно отключить). Все дальнейшие действия с письмами будут производиться именно на компьютере пользователя.

Протокол POP3 предпочтителен, если почтовым ящиком пользуется лишь один человек с одного-единственного компьютера, и доступ к почте необходим даже при отсутствии подключения к интернету.

IMAP

IMAP — протокол, поддерживающий прием и возможность управления электронной почтой прямо на почтовом сервере.

При использовании протокола IMAP электронные письма всегда находятся на сервере и скачиваются оттуда каждый раз при просмотре. Это очень удобно, когда доступ к почтовому ящику должны иметь несколько человек — все операции с письмами, которые произведет один из них (изменение статуса письма на «прочитанное», перемещение писем между папками и т. п.) увидят и все остальные. Недостаток — для работы с почтой, даже со старыми письмами, необходимо активное подключение к интернету.

SMTP

SMTP-сервер — это почтовый сервер, поддерживающий только отправку электронной почты.

При использовании SMTP-сервера используется порт 25. Некоторые интернет-провайдеры блокируют порт 25. В этом случае, вам нужно либо обратиться к администратору сети, чтобы он разблокировал порт, либо использовать альтернативный порт 587.

Gopher

Gopher — один из протоколов интернета, был популярен в начале 1990-х, впоследствии его роль уменьшилась. В основном с повсеместным развитием WWW. Ошибочно считается (а именно так написано в русской и английской википедии), что протоколы Gopher и HTTP были своего рода конкурентами, это и привело к уменьшению доли одного и увеличению другого. Нет, изначально Gopher был разработан для своеобразной замены FTP, протокола, который был разрабтан еще в 1960-х годах.

Gopher состоит из системы связанных гиперссылками меню. Ссылки могут вести на другие сервера. Зайдя на «страницу» мы можем наблюдать меню с директориями и файлами, а также с ссылками на другие «страницы». Это чем-то похоже на то, как, для примера Firefox, показывает список файлов и директорий если мы укажем ему путь на локальном диске.

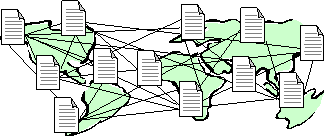
За протоколом был закреплен 70-й порт. Протокол настолько простой, что его можно использовать вообще без клиента (используя telnet). Кстати именно через telnet у меня получилось смотреть utf-8 файлы на русском. Другие клиенты не захотели этого делать.

На данный момент, про Gopher слышали немногие люди, а поколение «после Чернобыля» вообще обошло данный этап развития сети стороной. Я не хочу сказать, что протокол незаслуженно забыт (сегодня количество функционирующих Gopher серверов измеряется сотнями, а если быть более точным, то 133), он совершенно не вписывается с современное понимание сети. WWW/HTTP покрывает вопрос распространения/доступа к распределенной по сети информации, оброс множеством возможностей и сопутствующих технологий, так что попыток вдохнуть жизнь в Gopher даже не предвидится.

# Служба WWW.

Интернет – это система, которая объединяет компьютерные сети по всему миру. Компьютеры подразделяются на две группы – клиенты и серверы. Клиентами называют обычные пользовательские устройства, куда входят и персональные компьютеры, и ноутбуки, и планшеты, и, конечно же, смартфоны. Они отправляют запрос, получают и отображают информацию.

Серверами являются мощные компьютеры, работающие непрерывно. Кроме хранения информации они получают запросы от клиентов и отправляют необходимый ответ. При этом обрабатывают они сотни таких запросов.



**Служба World Wide Web** (сокращенно WWW, англ. *Всемирная паутина*) – это служба поиска и просмотра гипертекстовых документов. Эти документы называются **Web-страницы**, а совокупность близких по смыслу или тематике и хранящихся вместе Web-страниц называется – **Web-сайт**или **Web-узел**.

Web-страницы могут включать в себя текст, рисунки, анимацию, звук, видео, а также активные элементы – небольшие программы, оживляющие страницу, делающие ее интерактивной, то есть изменяющейся в зависимости от действий пользователя.

На самом деле разница между Интернетом и Всемирной паутиной достаточно большая. Если Интернет – это огромная сеть, связующая миллионы компьютеров по всей планете для совместного использовании информации, то Всемирная паутина лишь один из способов обмена этой информацией. Кроме обеспечения работы World Wide Web, Интернет позволяет использовать электронную почту и различные мессенджеры, а также передавать файлы по по FTP-протоколу.

Источник: http://topor.info/internet/internet-i-vsemirnaya-pautina

Главной и первоначальной идеей этой службы является **идея гипертекста**. Эту идею выдвинул Тим Бернерс Ли в 1989  году в качестве новой основы доступа к информации и он не ошибся, именно широкое использование гипертекстовой структуры послужило началом бурного развития данной службы и всего Internet в целом. Благодаря идее гипертекста и службе WWW в Internet стали участвовать не только профессионалы, но и огромное число обычных пользователей.

Идея гипертекста проста: **гипертекст**– это текст, содержащий ссылку на другой документ, который может быть как аналогичной Web-страницей, так и текстом, рисунком, звуком, видеозаписью и вообще файлом любого формата. Гипертекст представлен в виде гиперссылок, выделенных на странице обычно подчеркиванием, цветом, по которым достаточно щелкнуть мышью, и будет осуществлен переход к другой Web-странице или загружен нужный файл. Именно потому, что страницы с помощью гиперссылок переплетены между собой, эту службу называют «паутина».

Технологически Web-страницы представляют собой текстовые файлы, написанные на языке HTML. **HTML** (Hyper Text Markup Language) – это язык разметки гипертекста, он позволяет не только разместить на странице текст как таковой, но и указать его формат, то есть шрифт, размер, выравнивание. Это же самое относится и к рисункам и другим элементам, только рисунки размещаются в отдельных файлах, а внутри страницы указываются имена этих файлов. В настоящее время в связи с развитием данной технологии Web-страницы стали мультимедийными.

Но для того, чтобы Web-страница была выведена пользователю не в виде английских слов языка HTML вперемежку с русским текстом, а так, как она действительно должна выглядеть, используются программы просмотра Web-страниц, которые называются **браузеры** (англ. *browser*– обозреватель) или **обозреватели**.

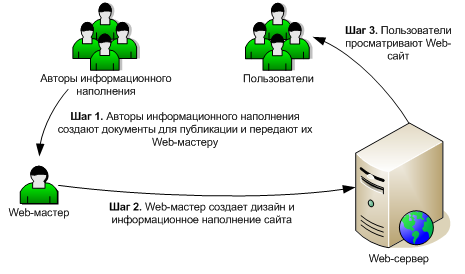
В мире насчитывается множество браузеров: Netscape Navigator, Opera и другие. И все-таки одним из самых распространенных является **Microsoft Internet Explorer**, поставляемый вместе с операционной системой Windows.

Программа браузер работает на компьютере пользователя и является клиентом. Она запрашивает нужную информацию на Web-серверах, с которыми она общается по при помощи специального **протокола НТТР** (Hyper Text Transfer Protocol) – протокола передачи гипертекста.

Приемы работы с программой Microsoft Internet Explorer будут подробно освещены в следующей главе, а пока остается добавить, что служба WWW включает в себя огромный массив данных, например, новости, погода, биржевые котировки, курсы валют, предложения работы, электронные доски объявлений, каталоги, сведения о кинофильмах, литературные обзоры, игры, магазины и многое, многое другое. Эти данные размещаются на Web-страницах не только от организаций, но и от персональных пользователей, а разместить свою информацию на таких страницах можно подчас бесплатно.

# 15. Использование серверных расширений для управления содержимым (контентом) сайта.

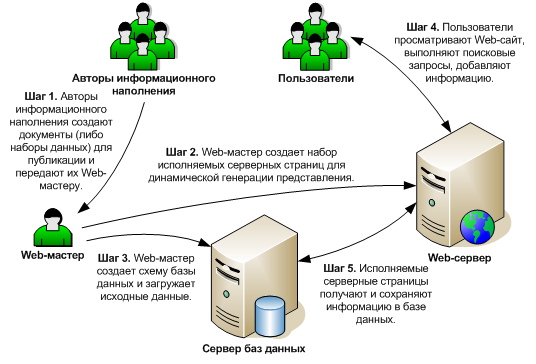
Системы управления контентом являются важнейшим классом систем в World Wide Web. Бум на них прошел приблизительно в 2000-2001 годах, тогда в прессе появилось несколько публикаций, рассказывающих о том, что такое системы управления контентом и как их использовать. С тех пор прошло уже достаточно времени (а по меркам информационных технологий – целая вечность), и рынок систем управления контентом стабилизировался. Появились признанные лидеры, пользователи, поработав с различными системами, высказали свои мнения и пожелания. Таким образом, сегодня появляется возможность всесторонне (как говорится, “с высоты птичьего полета”) рассмотреть этот класс систем: дать точное определение системы управления контентом, составить на основе различных реализаций точный перечень функций, которые должны выполнять подобные системы, и, наконец, взглянуть на лучших представителей этого класса.

  
*Рисунок 1. Информационные системы первого поколения (статические сайты).*

**Введение**

Информационная гипертекстовая система World Wide Web является самым популярным ресурсом сети Internet. В 1989 году Тим Бернерс–Ли, которого принято считать основателем World Wide Web, предложил руководству Центра ядерных исследований в Швейцарии концепцию новой распределенной информационной системы. Эта система, построенная на принципах гипертекста, должна была объединить все множество информационных ресурсов Центра. Вскоре проект был одобрен, и работа началась. Программа Mosaic, разработанная Марком Андресеном и построенная на принципах World Wide Web, обеспечила бурный рост этой системы и внесла огромный вклад в развитие ряда спецификаций World Wide Web. С тех пор развитие World Wide Web продолжалось фантастическими темпами. В январе 1993 года число Web-серверов достигло 50. Приблизительно через год их количество увеличилось в 4 раза. А в 1995 году World Wide Web превратился в столь заметное явление, что стал главной темой, обсуждавшейся на встрече Большой Восьмерки, которая состоялась в феврале этого года в Брюсселе. Сейчас число пользователей сети Internet составляет сотни миллионов, а количество Web-сайтов, согласно исследованию компании Netcraft, превысило 60 миллионов. Всемирная паутина (именно так переводится World Wide Web) настолько популярна, что ее даже иногда путают со всей сетью Internet (в которой, кстати, используются около десятка других протоколов), и часто, когда произносят World Wide Web, подразумевают Internet и наоборот. В связи с высокой популярностью всемирной паутины, многие люди занимаются ее изучением и даже пытаются классифицировать информационные системы, построенные на принципах World Wide Web. На сегодняшний день существуют различные подходы к выделению классов или, точнее, поколений Web-систем, например, в зависимости от богатства графики или анимации. Однако чаще всего информационные системы, построенные на принципах World Wide Web, делят на три поколения в зависимости от способа управления информацией: статические сайты, динамические и сайты, построенные на основе систем управления контентом.

Первое поколение представляет собой статические сайты, созданные в виде набора HTML-страниц, которые управляются и поддерживаются вручную. Специалисты, отвечающие за создание и поддержку системы, создают в формате HTML каждую страницу в отдельности, включая ее оформление и содержание. Очевидно, что такой подход приводит к смешению содержания и представления, а также исключает возможность повторного использования кода. Это усложняет процесс публикации новых сообщений и редактирования уже существующих (кроме непосредственной подготовки сообщений, их приходится еще и оформлять в соответствии с принятым дизайном сайта). При изменении дизайна сайта и общих элементов навигации страниц, таких как меню, заголовок, информация о разработчике, приходится вручную переделывать каждую страницу. Кроме того, реализовать бизнес-процессы, например, для поддержки электронных издательств, на базе статических сайтов практически невозможно. Вся работа по поддержке дизайна и информационного наполнения сайта взваливается на одного человека (или группу людей) – web-мастера. Все остальные, если они хотят добавить новую информацию или изменить представление уже существующей, должны обращаться исключительно к этому человеку, который в дальнейшем, возможно через достаточно длительный промежуток времени, создаст необходимые страницы. Поэтому сегодня, по мере объединения жизни реальных предприятий с World Wide Web, когда размеры гипертекстовых систем достигли громадных размеров, традиционная архитектура статических страниц, в которой все обязанности возлагаются на одного человека, уже не работает. Организациям приходится нанимать сотни людей, что приводит к дополнительным издержкам и создает проблемы несогласованности разработчиков.

  
*Рисунок 2. Информационные системы второго поколения (динамические сайты).*

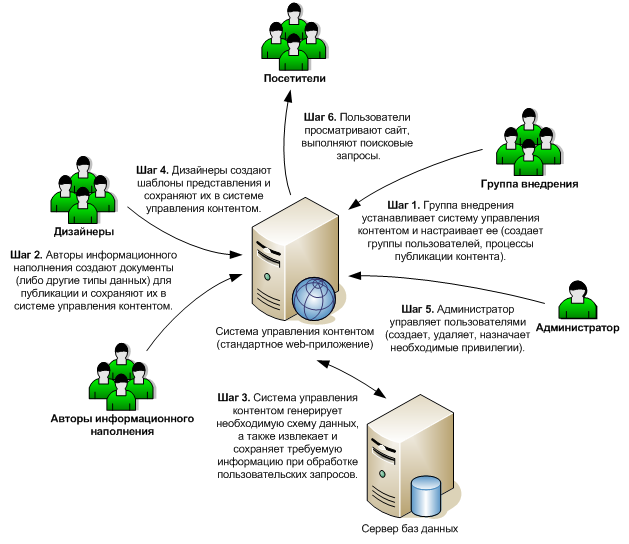
Web-сайты второго поколения или динамические сайты создаются на основе различных серверных технологий, динамически генерирующих HTML, например Active Server Pages от Microsoft или Java Server Pages от Sun . При этом web-приложение представляет собой набор функций, генерирующих различные части web-страниц на основе информации, которая хранится, например, в реляционной базе данных. Такой подход дает возможность реализовать авторизацию и аутентификацию пользователей, а также генерацию персонального представления для каждого посетителя. Кроме того, при таком подходе удается решить проблему повторного использования кода, а также появляется возможность разделить работу между программистами и дизайнерами с помощью различных шаблонных механизмов. Однако при отсутствии развитого механизма управления информацией и шаблонами представления, такое разделение обязанностей очень затруднительно. В итоге представление и содержание чаще всего по-прежнему управляется одним человеком – web-мастером. Администраторы (или разработчики) сайтов должны сами загружать всю информацию в базы данных или иные хранилища информации, а также разрабатывать требуемое представление. Как следствие, динамические сайты дороги в разработке и предъявляют высокие требование к квалификации разработчиков и администраторов. Несмотря на это, при качественной разработке и поддержки динамические сайты наиболее гибки в использовании и являются мощным инструментом поддержки бизнес-процессов.

В информационных системах второго поколения (динамические сайты), также как и первого, присутствует “эффект бутылочного горла”, - чтобы опубликовать информацию, нужно пройти узкое горлышко, которое ассоциируется с web-мастером. В следующем поколении информационных систем разработчики решили облегчить себе работу и автоматизировать процесс публикации информации, предоставив пользователям возможность самим публиковать материалы в WWW и определять их визуальное представление, используя для этого стандартные средства, не требующие знания языка HTML и достаточно сложных для неспециалиста процедур. Были созданы *системы управления контентом*(в английском языке существует устоявшийся термин – *Content Management Systems*или, сокращенно, *CMS*). Системы управления контентом – наследники средств визуальной разработки и поддержки сайтов, таких как Microsoft FrontPage или Macromedia DreamWeaver. С их помощью можно, не будучи профессиональным разработчиком, создавать и модифицировать информационное наполнение сайтов. Но поскольку такие системы работают преимущественно со статическим представлением информации, сегодня они не устраивают пользователей. Основная идея систем управления контентом – разделение визуального дизайна сайта и его информационного наполнения. При создании сайта с помощью такой системы разрабатывается набор шаблонов страниц, в которых впоследствии размещается информация. В этом случае роль разработчиков (фактически это группа внедрения) ограничивается только созданием “начальной” информационной системы на основе системы управления контентом, затем пользователи сами публикуют требуемую информацию и определяют ее представление. Управление сайтом сводится к минимуму, – администратору остается только управлять пользователями.

При использовании систем управления контентом не возникает эффекта бутылочного горла, когда вся работа взваливается на одного человека. Пользователи традиционно делятся на две группы – создатели шаблонов страниц и авторы контента (информационного наполнения). Таким образом, одна группа пользователей создает структуру и оформление страниц, а другая наполняет его содержанием.

**Системы управления контентом**

Прежде чем окончательно понять, что такое системы управления контентом, необходимо рассмотреть понятие контента. Под контентом (дословный перевод английского термина content, означающего содержание, содержимое) понимают информационное наполнение сайта – то есть все типы материалов, которые находятся на сервере: web-страницы, документы, программы, аудио-файлы, фильмы и так далее. Таким образом, управление контентом – это процесс управления подобными материалами. Он включает следующие элементы: размещение материалов на сервере, удаление материалов с сервера, когда в них больше нет необходимости, организацию (реорганизацию) материалов, возможность отслеживать их состояние, например, используются они или нет. Управлять контентом можно различными способами в зависимости от решаемых задач. Например, обычному пользователю, поддерживающему свою домашнюю страницу, нужна только одна возможность – всегда видеть последнюю версию своей страницы. Разработчикам программного обеспечения требуется для поддержки совместной работы хранить версии своих программ на сервере. Это позволяет им вернуться в случае возникновения ошибки к одному из предыдущих состояний. Разработчикам web-сайтов нужно знать, кто над какой страницей в данный момент работает, чтобы не перезаписать по ошибке чью-то работу. Возможность хранения всех версий сайта для них не так уж и важна. Большой корпорации, возможно, потребуется назначить ответственных за различные аспекты работы сайта: например, юридический отдел отвечает за правовую информацию, отдел маркетинга – за описание продуктов, а дизайнеры – за то, как все это выглядит. Более того, очень часто возникает необходимость в дополнительных просмотрах изменений страниц до того, как они станут доступными широкой публике. Большая страховая компания или банк по закону обязаны сохранять копии Web-страниц в течение нескольких лет. Такая разница в требованиях приводит к разнице в технологиях и продуктах, которые обеспечивают управление контентом. В то же время потребности пользователей развиваются в различных направлениях, и они не хотят полностью менять свои системы управления контентом, чтобы удовлетворить новые потребности.

  
*Рисунок 3. Информационные системы третьего поколения, построенные на основе систем управления контентом.*

Согласно классификации Giga Information Group, системы управления контентом делятся на четыре основных категории, которые частично перекрываются. *Системы управления исходными кодами* традиционно поддерживают управление исходными кодами программ, и часто предоставляют некоторый web-интерфейс, который может использоваться внутри корпоративной сети, а также вне ее для параллельной работы с исходными кодами. *Системы управления документами*предназначены для организаций, оперирующих с большим количеством документов, например, офисы больших компаний, редакции и страховые компании. *Системы управления web-контентом*представляют собой новую индустрию программных продуктов. Эти системы предназначены для разработки и управления Web-сайтами различной степени сложности. Обычно такие системы поддерживают и некоторый тип управления потоками работ. *Системы электронной коммерции***–**обеспечивают хранение и управление электронными каталогами товаров. По сути, эти системы незначительно отличаются друг от друга. Самое главное отличие этих систем – это люди, которые их используют.

Каждой из этих категорий систем соответствует свой прикладной интерфейс программирования. Например, системы управления исходными кодами и средства разработки общаются между собой согласно общему интерфейсу, который называется Source Code Control API (SCC API). Применение этого интерфейса ограничено платформами Microsoft Windows, но, тем не менее, это большой шаг к тому, чтобы пользователи различных средств разработки могли хранить свои исходные коды в хранилищах различных поставщиков. Точно так же, в мире управления документами, существует интерфейс – Document Management API (DMA). Разработчики средств обработки документов часто поддерживают этот интерфейс, чтобы интегрировать свои средства разработки с DMA-хранилищами документов. В системах управления Web-контентом используется Web Distributed Authoring and Versioning (WebDAV), который представляет собой набор расширений протокола HTTP/1.1, обеспечивающих публикацию, совместное редактирование и управления файлами на удаленных Web-серверах. Кроме того, в последнее время появляются организации, пытающиеся объединить разработчиков систем управления контентом, создать единую информационную среду для потенциальных пользователей подобных систем, разработать и утвердить единые стандарты. Прежде всего, это ассоциация Open Source Content Management (OSCOM, http://www.oscom.org/). Одним из ее основных проектов является CMSML – язык, основанный на XML и предназначенный для описания систем управления контентом. Элементы этого языка описывают три основных задачи, которые решают системы управления контентом, — создание, управление и доставка контента конечному пользователю. В рамках проекта Java Community Process (JCP) ведутся работы над JSR-170, новой спецификаций API для хранилищ контента. Этот интерфейс содержит описание управления версиями контента, полнотекстовый поиск, разграничение доступа на уровне отдельных атрибутов, управление категориями и отслеживание событий.

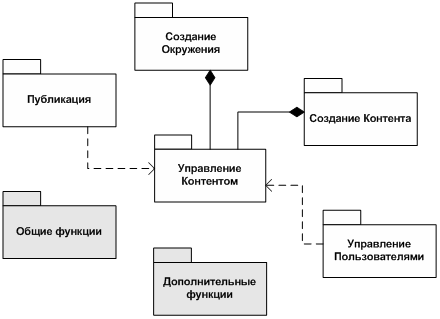
Тем не менее, единых стандартов в этой области, в том числе на терминологию, пока еще нет. Как правило, под системами, обозначаемыми общим термином *системы управления контентом*понимаютименно системы управления Web-контентом, адля систем управления исходными кодами, систем документооборота и систем поддержки электронной коммерции используются собственные названия. Иногда различают понятия системы управления контентом и системы управления сайтом (Иван Косяков, "Создание бизнес-сайта компании"). Система управления контентом — это система управления информацией на сайте. Система управления сайтом — это система, которая позволяет управлять информацией на сайте, вносить изменения в дизайн сайта и осуществлять изменение структуры сайта. Однако сегодня практически все продукты, которые продвигаются на рынок под названием *система управления контентом*, реализуют обе эти функции.

**Функции систем управления контентом**

Требования, предъявляемые к системам управления контентом, у каждой организации разные. В ходе подготовки статьи был проанализирован целый ряд систем управления контентом, ссылки на которые можно найти в следующих источниках. Прежде всего, это CMSWatch (http://www.cmswatch.com) – организация, которая ежегодно выпускает отчет, включающий в себя обзор рынка систем управления контентом и сравнения некоторых из них. Кроме того, обзоры систем управления контентом публикуются на сайтах <http://www.la-grange.net/cms>, http://www.opensourceCMS.com и [http://www.cmsinfo.org](http://www.cmsinfo.org/).

Более того, на сегодняшний день существуют и специальные работы, посвященные анализу функций систем управления контентом. Среди них есть как коммерческие руководства, так и открытые публикации. Наиболее полное из них – это Content Management Requirements Toolkit (<http://www.steptwo.com.au/products/toolkit/>), который распространяется как коммерческий продукт. В нем упоминаются 112 функций систем управления контентом, которые разбиты на несколько разделов. Несмотря на кажущуюся полноту и ясность, этот продукт содержит много лишних или само собой разумеющихся требований. Например, то, что HTML, содержащийся в сгенерированных Web-страницах, должен быть правильным. Также вызывает вопросы и классификация функций систем управления контентом. Кроме указанного руководства, существует и ряд открытых публикаций, в которых также дается интересный перечень функций систем управления контентом, например, James Robertson "How to Evaluate a Content Management System" (опубликовано в [http://www.intranetjournal.com](http://www.intranetjournal.com/)), Paul Browning "CMS Feature List" (находится на сайте университета Бристоля, по адресу <http://www.bris.ac.uk/ISC/cms/features.htm>). Будем придерживаться следующей классификации и перечня функций систем управления контентом, полученного на основе перечисленных выше работ.

Функции систем управления контентом структурированы согласно жизненному циклу системы. Сначала администратор предоставляет доступ к системе различным пользователям, затем создается контент, и, наконец, он публикуется. На рисунке 4 представлены категории функций систем управления в виде UML-диаграммы: пунктирная линия со стрелкой обозначает зависимости между пакетами функций, например, механизм публикации зависит от механизма управления контентом, сплошная линия с ромбом на конце означает композицию, так пакет *Управление Контентом*(в котором, кстати, сосредоточена основная часть требований) содержит два подпакета *Создание Окружения*и *Создание Контента.* Первый – *Создание Окружения* соответствует настройке системы на определенный поток работ (менеджер создает поток работ, а также все типы данных и схемы их метаописаний). На втором этапе (пакет *Создание Контента*) пользователи начинают работать с контентом, согласно установленному потоку работ. Пакет *Управление Пользователями*описывает процесс управления группами, отдельными пользователями, а также назначение пользователям прав на выполнение определенных операций. Рассматривается возможность пользователя делегировать свои права. Пакет *Публикация* определяет требования к процессу публикации на основе механизма шаблонов. Пакет *Представление*определяет требования к полученному представлению информации. В данную спецификацию не включены требования, которые автоматически подразумеваются или являются общими для систем подобного класса (*Общие функции*), например, то, чтобы сгенерированный HTML был "правильным" и требования к производительности и защищенности, а также дополнительные требования (*Дополнительные функции*), которые часто предъявляют к системам управления контентом, например, наличие форумов и конференций (они могут быть реализованы “поверх” систем управления контентом). Пакеты этих требований выделены серым цветом.

  
*Рисунок 4. Классификация функций систем управления контентом.*

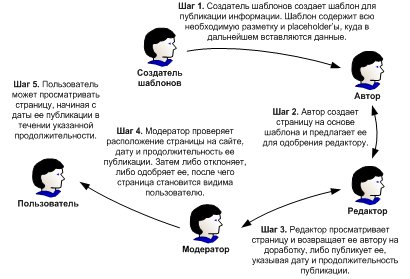
В мире существует множество систем управления контентом. Условно можно выделить три основных класса подобных приложений. Это системы крупных производителей, например, Microsoft Content Management Server, системы с открытым исходным кодом, например, RedHat CMS и OpenCMS, и разработки небольших компаний, которые применяются на нескольких проектах, созданных непосредственно компанией-разработчиком (более 80% всех систем управления контентом). Ниже мы более подробно рассмотрим функции систем управления контентом, входящих в каждый пакет требований, а также реализацию этих функций в двух наиболее известных и старейших системах, написанных на языке Java. Это – система управления контентом Red Hat (бывшая ArsDigita) и система управления контентом OpenCms (существует русскоязычный сайт для поддержки этой системы, [http://www.opencms.ru](http://www.opencms.ru/)). Эти системы были отобраны по следующим критериям. Во-первых, они достаточно технологичны: платформа J2EE довольно продвинута и содержит множество стандартных сервисов, например, поддержки транзакций, доступа к базам данных, глобализации приложений и так далее. Во-вторых, в отличие от продуктов Microsoft они распространяются бесплатно, поэтому каждый может скачать и установить их. Если рассматриваемая функция отсутствует у обеих систем или реализована недостаточно хорошо, то будет рассматриваться ее реализация в других системах, например, в Microsoft Content Management Server.

**Создание окружения**

На первом этапе необходимо *создать все типы контента* и *схемы их метаописаний*, а также настроить *систему на определенный поток работ* (если система поддерживает создание потоков работ, а не использует единственный встроенный). Понятие *типа контента* аналогично понятию класса, а элементы контента представляют собой набор экземпляров таких “классов”. Типами контента являются, например, текст и изображение; экземпляром контента конкретный документ или картинка. Часто в системах управления контентом, написанных на языке Java, тип контента – это класс Java, который инкапсулирует все остальные типы. Это решение используется и в RedHat CMS и в OpenCms.

Кроме поддержки непосредственно создания типов контента, система управления контентом должна предоставлять и некоторые дополнительные возможности, например, задание *ограничений на информацию и ее проверка до опубликования*. Данная возможность очень важна – например, бывали случаи, когда электронные магазины из-за ошибок администраторов выкладывали свои товары по смешным ценам. В системах управления контентомRedHat и OpenCMS для этого придется задать ограничения на поля классов, представляющих типы контента. Кроме этого,система должна предоставлять возможность *интернационализации и локализации*контента, а также сама отвечать требованиям интернационализации и локализации.

Многоязычность должна поддерживаться как на уровне административного интерфейса системы, так и на уровне самих Web-приложений, работающих под управлением системы управления контентом. Для этого система должна предоставлять соответствующие функции и структуры данных. В системе управления контентомRedHat встроенная локализация отсутствует. Напротив, в системе управления контентомOpenCMS можно выбирать различные языки представления (пока только немецкий и английский).

  
*Рисунок 5. Рабочий процесс в Microsoft Content Management Server.*

Следующая важная возможность – *хранение информации о версии контента*. Это позволяет задать номер версии любых операций изменения контента и при необходимости восстановить его. В любой момент можно отказаться от изменений и, практически в режиме реального времени, откатиться на одну из предыдущих зафиксированных версий. Строгий контроль версий необходим для определения ответственности отдельных лиц, а также для резервного и аварийного восстановления системы. Система управления контентомRedHat для каждого элемента контента хранит все его версии, и вся работа ведется с последней версией (latest). Кроме того, существует и текущая (live) версия, которая видна пользователям сайта. Система управления контентом OpenCMS может хранить все версии элементов контента, и при желании можно вернуться к любой из них.

Кроме управления контентом, система должна предоставлять возможность создавать *метаданные о нем*. *Метаданные*– это сведения о данных, свойства данных. Примером метаданных служат ключевые (характерные) слова документов, предназначенные для поисковых или отчетных систем. Системы управления контентом, рассматриваемые в данном обзоре, не поддерживают метаданные, хотя можно специально ввести дополнительные типы контента, представляющие собой метаданные. В Microsoft Content Management Server для каждого элемента контента можно задать набор свойств, и использовать последний для различных целей, например, для поиска данных или разбиения элементов контента на различные категории.

Одной из важнейших возможностей, которые должна предоставлять система управления контентом, является поддержка рабочих процессов (workflow). Рабочий процесс(workflow) – это набор методов и технологий для автоматизации бизнес-процесса в целом или его части посредством документов, когда информация или задания передаются от одного участника к другому для выполнения определенных действий. Сегодня широко применяются автоматизированные системы управления рабочими процессами. Системы управления контентом также поддерживают управление рабочими процессами.

Система управления контентомRedHat позволяет связывать с каждым типом контента рабочий процесс, который тот должен пройти, прежде чем стать видимым пользователям. Каждый процесс в RedHat CMS состоит из набора заданий, между которыми можно задать зависимости (например, чтобы выполнить одно задание, необходимо выполнить несколько предыдущих), и для которых можно назначить конкретных исполнителей. Таким образом, можно создавать собственные рабочие процессы различной степени сложности. Аналогичным образом работает и система управления контентомOpenCMS**.**

Типичный процесс публикации информации в World Wide Web реализован в Microsoft Content Management Server. Обычным приемом обеспечения оформления информационного наполнения являются шаблоны представления информации. Поэтому первым этапом процесса является создание наборов шаблонов. Типичный шаблон содержит разметку HTML и места, куда в дальнейшем будут вставлены данные (placeholder’ы в терминологии Microsoft). Далее на основе этих шаблонов авторы информационного наполнения создают страницы и представляют их редакторам для одобрения. Редакторы, в свою очередь, могут либо отклонить страницу и вернуть ее автору на доработку, либо одобрить ее и передать модератору сайта. В первом случае процесс повторяется снова, во втором же модератор сайта проверяет расположение страницы на сайте, дату и срок ее публикации. Если все в порядке, страница становится видна пользователям. Несмотря на то, что рабочий процесс в Microsoft Content Management Server фиксирован и не может быть изменен в дальнейшем, подобное решение подходит большинству пользователей, которым необходимо публиковать информацию в World Wide Web.

**Создание контента**

После того, как все типы контента созданы и рабочие процессы запущены, авторы информационного наполнения начинают *создавать, изменять и удалять элементы контента*указанного типа. В системе управления контентом RedHat для каждого вновь созданного типа контента необходимо написать собственный код для редактирования и обновления элементов контента данного типа (в терминах RedHat это называется Authoring Kit). *Authoring Kit* типа контента состоит из набора визуальных компонентов для просмотра и редактирования данных, связанных с элементами этого типа. RedHat CMS уже содержит некоторый набор визуальных компонентов, например, для редактирование текста, выбора изображений, выбора шаблона представления.

Кроме непосредственно редактирования элементов контента, необходимо предусмотреть *разбиение контента по категориям или рубрикам*. Обычно категории организуются в некоторые иерархические структуры, в узлах которых содержатся элементы контента. Система управления контентомRedHat поддерживает иерархию секция-категория-элемент. Система управления контентомOpenCMS поддерживает группировку элементов содержимого в каналы, но поскольку элементом содержимого является класс, то можно создавать связи между классами, при этом некоторые из них и будут представлять категории.

Рисунки и изображения, а также некоторые другие ресурсы являются важной составляющей каждой информационной системы. И разработчики, и дизайнеры должны знать, как унифицированным способом их использовать. Поэтому система управления контентом должна поддерживать *базу (библиотеку) ресурсов*для использования в информационных материалах. Работа с изображениями, анимациями и другими подобными ресурсами должна быть понятна пользователю, при этом обычно создается некоторое хранилище ресурсов. Туда могут входить изображения, фильмы, презентации и т.д. Это позволяет более эффективно использовать, разделять и администрировать ресурсы. В системе управления контентомRedHat такая работа с изображениями (и другими мультимедийными ресурсами) отсутствует. Система управления контентомOpenCMS поддерживает галереи ресурсов (например, картинок).

Возможность поиска объектов и их частей также важна для систем управления контентом. Некоторые системы имеют собственные механизмы поиска, другие ведут поиск средствами нижележащих хранилищ данных, которые они используют. Система управления контентом RedHat поддерживает поиск на основе механизма поиска базы данных Oracle. В системе управления контентом OpenCMS поиск основан на Apache Lucene (свободно распространяемая система индексации документов).

Система управления контентом является только одной из множества систем, которые используются на предприятии. Использование системы управления контентом будет успешным, если она будет *обмениваться информацией с уже существующими системами*. Механизмы этого объединения должны быть основаны на открытом или промышленном стандарте. Контент может подгружаться в базу данных или напрямую отображаться в графических шаблонах в режиме реального времени. Система управления контентомRedHat не поддерживает интеграцию с внешними системами. Единственным способом взаимодействия с внешними системами является взаимодействие на уровне нижележащих хранилищ данных, т.е. загрузка и выгрузка информации непосредственно из реляционного хранилища. В системе управления контентом OpenCMSподдерживается взаимодействие с EJB**,**где можно использовать Java Connectors**.** Java Connectors определяет стандартный Connector Service Provider Interface (Интерфейс поставщика услуг коннектора), который, как правило, реализует разработчик корпоративной информационной системы предприятия. Многие производители КИС (BEA, IBM, SAP, iPlanet) представляют адаптеры ресурсов, реализующие Connector SPI для обеспечения взаимодействия управляющих средств контейнера/сервера J2EE с EIS.

**Управление пользователями**

Управление пользователями включает создание, изменение и удаление учетных записей отдельных пользователей и их групп, а также назначение прав для работы с элементами контента. Важной частью требований является наличие пользовательских профилей (profiles), с помощью которых можно сгенерировать персональное представление информации для каждого пользователя. Полезной является и возможность пользователя делегировать свои права. Это позволяет пользователям переназначать исполнителя конкретной работы и избегать простоев из-за отсутствия отдельного лица.

Системы управления контентом RedHat и OpenCMS управляют учетными записями пользователей на основе собственных групп, не используя существующие идентификационные системы, например, Windows. Аутентификация средствами Windows позволила бы значительно упростить администрирование. При этом система управления контентом могла бы использовать операционную систему локального компьютера или контролера домена для проверки и сопровождения учетной записи пользователя

Представление информации создается на основе данных, а также предпочтений конкретного пользователя. Персонификация достигается путем использование профилей – специальных записей, в которых хранится информация, специфичная для конкретных пользователей. Система управления контентом RedHat не предоставляет возможности хранения персональных данных пользователей (хотя ее можно и реализовать, введя новые типы контента). Система управления контентом OpenCMS допускает создание индивидуального представления контента. Встроенная система администрирования OpenCMS позволяет сохранять связанные с пользователем данные.

**Публикация информации**

Механизм публикации информации в системе управления контентом отвечает за процесс создания, редактирования и удаления шаблонов страниц, а также за сопоставление типов контента и шаблонов страниц. В состав дополнительных возможностей системы публикации может входить предварительная генерация статической версии сайта. Эта опция очень полезна в случае размещения информационной системы на оборудовании с ограниченными возможностями.

Как уже говорилось, чаще всего для решения проблемы представления в системах управления контентом используется технология шаблонов, определяющих внешний вид страницы. В идеале, разработчику шаблонов не нужно знать никаких технических тонкостей. На ранних этапах существования WWW шаблоны представляли “заготовки” HTML-кода, из которого путем манипуляций в HTML-редакторе получались готовые страницы. Сегодня такими заготовками манипулируют уже не дизайнеры в своих редакторах, а серверные web-приложения. Таким образом, современный шаблон Web-страницы представляет собой блок HTML, который благодаря специальным тегам или внедренным сценариям, облегчает включение динамически сгенерированного содержания на этапе выполнения. При использовании подобных шаблонов программистам необходим некоторый стандартизированный интерфейс для работы с ними – шаблонный движок (в английском языке существует устоявшийся термин – template engine), который может иметь разнообразные дополнительные функции, например, поддерживать кэширование шаблонов, их динамическое обновление и т.д.

Существует большое количество подходов к созданию шаблонов Web-страниц и множество шаблонных движков, написанных на всевозможных языках программирования и предназначенных для использования на различных серверных платформах. Более подробный обзор шаблонных движков и примеры их применения на практике приводятся в (Михеев Петр “Подходы к созданию шаблонов web-страниц в серверных Java-приложениях” Технология клиент-сервер, № 1, 2003).

Система управления контентом RedHat использует XSLT-преобразования на стороне сервера, которые в данном случае и являются шаблонами представления. Сценарий генерации страницы в этом случае описывается следующей схемой: клиент запрашивает требуемую страницу; сервер находит и сериализует в XML все элементы контента, находящиеся на этой странице; сервер находит и применяет соответствующее преобразование к этому документу; результат преобразования (требуемемая страница) возвращается клиенту. В отличии от RedHat CMS, OpenCMS использует Java Server Pages (JSP) и собственные библиотеки тегов для создания шаблонов страниц. Наличие библиотек тегов позволяет организовать работу так, чтобы дизайнеры и программисты могли решать свои задачи практически независимо друг от друга. При этом дизайнеры разрабатывают визуальное представление, используя HTML-, XML- и JSP-теги, а разработчики концентрируют свое внимание на реализации этих тегов, предназначенных, например, для доступа к свойствам элементов контента.

**Заключение**

Системы управления контентом определяют будущее сайтостроения. Автоматизируя процесс публикации информации, системы управления контентом упрощают разработку и снижают стоимость поддержки сайтов. Однако многообразие типов представленных на рынке систем такого назначения, широкий спектр решаемых ими задач и не до конца устоявшиеся требования к ним не позволяют пока однозначно определить лидера. Прежде чем сделать выбор, разработчику стоит рассмотреть как можно больше альтернативных систем. Если же случится так, что вы захотите разработать собственную систему, я буду рад, если эта статья хотя бы чуть-чуть поможет вам в этом.

Примеры отечественных систем управления контентом

Q-Publishing — это уникальный механизм, позволяющий запустить интерактивный Web-сайт в рекордно короткие сроки, не привлекая к работе программистов и используя лишь WYSIWYG-средства. Программный продукт Q-Publishing представляет собой расширяемую систему, созданную Quantum Art для решения задач по доставке и контролированию динамического контента. Q-Publishing — это эффективный и удобный инструмент для создания, администрирования и обновления Web-ресурсов, используя который можно сформировать неограниченное число групп, члены которых будут выполнять определенные обязанности и обладать различными правами при создании, форматировании и доставке динамического контента.

Программа Q-Publishing позволяет Web-мастерам более продуктивно сотрудничать с другими членами своих групп и обновлять информацию регулярно и быстро. Это решение — уникальная возможность автоматизировать документооборот и эффективно координировать выполняемые операции.

Q-Publishing является удобным механизмом определения структуры контента, представляемого на динамически формируемом Web-сайте. Пользователи Q-Publishing имеют возможность создавать и модифицировать шаблоны всех объектов, формирующих многочисленные Web-страницы.

Optimizer — это технология создания современных сайтов, предоставляющая эффективный инструмент для последующего сопровождения.

Под сопровождением понимается публикация документов, развитие структуры сайта и управление интерактивными сервисами. Технология не требует от операторов каких-либо специальных знаний, кроме основных принципов работы с компьютером, и избавляет от необходимости ручной правки шаблонов и страниц. Технология Optimizer создавалась с учетом требований сотрудников, работавших над информационной поддержкой сайтов. Поэтому, с точки зрения операторов, ресурс, созданный на основе технологии Optimizer, удобен, понятен и надежен в работе.

Optimizer сочетает в себе достоинства систем управления контентом (CMS) и гибкой платформы разработки Web-проектов, позволяющей реализовывать множество задач в сжатые сроки. Для Web-студий, заинтересованных в использовании технологии Optimizer, открыта партнерская программа, в рамках которой проводится обучение и предоставляется документация разработчика.

Saitistika — это система для управления корпоративным Web-контентом (Content Management System), обеспечивающая удобный способ построения, управления, масштабирования и изменения Web-систем любого уровня сложности — от небольших промо-сайтов до Интернет-, экстранет-, интранет-систем.

При разработке системы, помимо функциональной составляющей, особое внимание уделялось ее внешнему виду и удобству работы с ней (usability). Благодаря дружественному интерфейсу и логичности структуры для работы с Saitistika достаточно квалификации рядового пользователя.

Цена системы определена с учетом потребностей и возможностей среднего и крупного российского бизнеса, который испытывает затруднения в приобретении дорогостоящих зарубежных аналогов.

Платформа Puzzle представляет собой систему разработки и поддержки любых Web-проектов — от Web-представительства компании и электронного магазина до корпоративных информационных порталов и службы управления поставками.

Благодаря тому, что Puzzle состоит из отдельных компонентов, имеется возможность ее быстрого расширения. Например, если первоначально предпочтение было отдано гостевой книге в качестве средства общения с посетителями, в дальнейшем очень легко превратить ее в полноценный Web-форум, добавить списки рассылки и т.п.

Кроме того, если в ходе работы проекта нужно каким-либо образом изменить или доработать функциональность того или иного компонента, эта задача облегчается за счет многоуровневой архитектуры: изменения на одном уровне прозрачны для остальных. Эти факторы, естественно, обеспечивают уменьшение стоимости работ по развитию проекта.

# 16. Механизмы ADO.NET.

***ADO (ActiveX Data Objects)*** — это библиотека компонентов СОМ, получившая в последние несколько лет множество воплощений. ADO состоит, прежде всего, из объектов Connection, Command, Recordset и Field. С помощью ADO открывается соединение с базой данных, после чего некоторые данные извлекаются и помещаются в набор записей, состоящих из полей; эти данные затем претерпевают манипуляции и обновления на сервере, после чего соединение закрывается. Кроме того, ADO предлагает так называемый *отключенный набор записей (disconnected record set)*, который используется, когда соединение с базой нежелательно удерживать открытым в течение длительного времени.

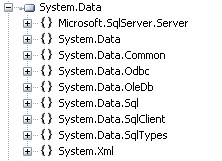
Существует несколько проблем, которые ADO не решает удовлетворительным образом. Наиболее заметная из них — громоздкость (в плане физического размера) отключенного набора записей. Потребность в этом средстве возрастает по мере развития веб-ориентированных вычислений, поэтому в данном случае понадобился свежий подход. Переход от ADO к ADO.NET не должен быть слишком трудным, поскольку между этими технологиями все же имеется некоторое сходство.

Более того, если вы используете SQL Server, существует замечательный набор управляемых классов, которые настроены на обеспечение максимальной производительности базы данных. Одного этого достаточно для перехода на ADO.NET.

***ADO.NET*** — нечто большее, чем надстройка над каким-нибудь существующим API-интерфейсом. Сходство с ADO минимально; классы и методы доступа к данным довольно существенно отличаются.

ADO.NET поставляется с тремя пространствами имен клиента базы данных: одно для **SQL Server**, другое для источников данных **Open Database Connectivity (ODBC)** и третье для любой базы данных, доступной через **OLE DB**. Если выбрана база данных, отличная от SQL Server, отдавайте предпочтение OLE DB, если только не окажется, что нет другого выбора кроме ODBC. Если в качестве базы данных используется Oracle, можете посетить сайт Oracle .NET Developer и получить там их поставщика .NET — ODP.NET на странице [www.oracle.com/technology/tech/windows/odpnet/index.html](https://professorweb.ru/my/ADO_NET/base/level1/www.oracle.com/technology/tech/windows/odpnet/index.html).

С точки зрения программиста, тело ADO.NET составляет базовая сборка с именем **System.Data.dll**. В этом двоичном файле находится значительное количество пространств имен, многие из которых представляют типы конкретного поставщика данных ADO.NET:



Большинство шаблонов проектов Visual Studio 2010 автоматически ссылаются на эту ключевую библиотеку доступа к данным. Однако для импортирования нужных пространств имен необходимо изменить кодовые файлы, например:

**using** System;

**using** System.Data;

**using** System.Data.SqlClient;

...

Учтите также, что кроме System.Data.dll, существуют и другие ориентированные на ADO.NET сборки (например, *System.Data.OracleClient.dll* и *System.Data.Entity.dll*), которые необходимо вручную указывать в текущем проекте с помощью диалогового окна Add Reference (Добавление ссылки).

**Три стороны ADO.NET**

Библиотеки ADO.NET можно применять тремя концептуально различными способами: в подключенном режиме, в автономном режиме и с помощью технологии Entity Framework. При использовании **подключенного уровня (connected layer)**, кодовая база явно подключается к соответствующему хранилищу данных и отключается от него. При таком способе использования ADO.NET обычно происходит взаимодействие с хранилищем данных с помощью объектов подключения, объектов команд и объектов чтения данных.

**Автономный уровень (disconnected layer)**, позволяет работать с набором объектов DataTable (содержащихся в DataSet), который представляет на стороне клиента копию внешних данных. При получении DataSet с помощью соответствующего объекта адаптера данных подключение открывается и закрывается автоматически. Понятно, что этот подход помогает быстро освобождать подключения для других вызовов и повышает масштабируемость систем.

Получив объект DataSet, вызывающий код может просматривать и обрабатывать данные без затрат на сетевой трафик. А если нужно занести изменения в хранилище данных, то адаптер данных (вместе с набором операторов SQL) задействуется для обновления данных — при этом подключение открывается заново для проведения обновлений в базе, а затем сразу же закрывается.

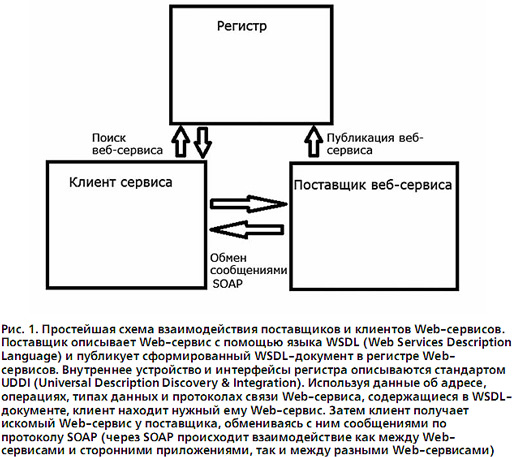
После выпуска .NET 3.5 SP1 в ADO.NET появилась поддержка нового API, которая называется **Entity Framework** (сокращенно EF). Технология EF показывает, что многие низкоуровневые детали работы с базами данных (например, сложные SQL-запросы) скрыты от программиста и отрабатываются за него при генерации соответствующего LINQ-запроса (например, LINQ с Entities).

# 17. Безопасность Интернет-сервисов.

Можно дать следующее определение: это вызываемый удаленно программный компонент, имеющий заданные функциональные возможности, доступный по стандартным протоколам сети Интернет и пригодный для многократного использования. Web-сервисы не зависят ни от программной платформы, ни от языка программирования, поскольку базируются на стандарте XML.

Web-сервис - это вызываемый удаленно программный компонент, имеющий заданные функциональные возможности, доступный по стандартным протоколам сети Интернет и пригодный для многократного использования.

Чтобы понять принцип работы Web-сервисов, давайте рассмотрим типичную схему взаимодействия их поставщиков и клиентов (рис. 1).

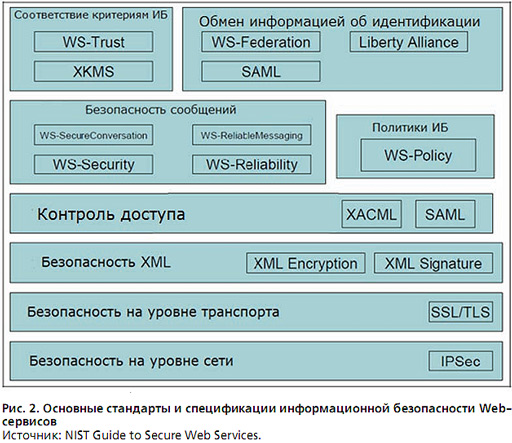


Уже приведенная схема демонстрирует нам особенности Web-сервисной среды, которые влияют на специфику их защиты. Назовем эти особенности:

* высокая доступность данных - Web-сервисы, собственно, и разрабатывались с целью повышения доступности и интегрируемости гетерогенных приложений;
* соединения с другими Web-сервисами и приложениями не ограничены физическими границами какой-либо защищенной сети;
* обмен данными происходит через общедоступную сеть Интернет, что предоставляет злоумышленникам расширенные возможности для их перехвата;
* данные передаются в формате, пропускаемом многими межсетевыми экранами (XML);
* возможность групповой адресации сообщений, затрудняющая применение средств информационной безопасности, ориентированных на одноадресную передачу;
* наличие особо уязвимых компонентов архитектуры Web-сервисов, в частности регистров, что позволяет вывести из строя сразу многие Web-сервисы в случае успешной DoS-атаки на уязвимый участок;
* минимальное участие человека в процессах реализации Web-сервисов - для большинства пользователей они представляются "черными" ящиками.

Согласно мнению аналитиков организации OASIS Web Services Interoperability, главные угрозы, нацеленные на Web-сервисы, - это несанкционированные изменения сообщений, потеря их конфиденциальности и аутентичности отправителей, DoS-атаки, то есть угрозы информационной безопасности Web-сервисов, практически идентичны угрозам, направленным на другие цифровые ресурсы. Обеспечение информационной безопасности Web-сервисов предполагает использование общепринятых технологий информационной безопасности - шифрования, цифровых подписей, парольной защиты и т.д. Можно сказать, что стандарты информационной безопасности Web-сервисов соответствуют специфичной архитектуре ИБ Web-сервисов, но механизмы реализации данной архитектуры вполне традиционны.

В настоящее время разработан целый ряд стандартов и спецификаций информационной безопасности Web-сервисов (рис. 2).



Одним из наиболее популярных стандартов является WS-Security, описывающий процессы аутентификации и авторизации в среде обмена SOAP-сообщениями. WS-Security предусматривает аутентификацию пользователя по средствам пар логин/пароль, сертификатов Х.509 или протокола Kerberos. Эти же технологии реализуют цифровую подпись, позволяющую удостовериться в целостности сообщения. Разработчики WS-Security также позаботились и о шифровании SOAP-сообщений, определив механизмы использования в среде SOAP стандарта XML Encryption.

Защита Web-сервисов должна быть комплексной: от информационных угроз необходимо защищать не только SOAP-сообщения, но также все другие составляющие архитектуры Web-сервисов: их интерфейсы, средства обнаружения сервисов (регистры) и т.д. Например, последняя версия перспективного стандарта UDDI обеспечивает такие немаловажные с точки зрения информационной безопасности функции, как целостность данных и шифрование содержимого регистра Web-сервисов.

В заключение можно сказать, что область стандартизации информационной безопасности Web-сервисов постоянно развивается и совершенствуется, и потому для обеспечения высокого уровня И Б мы рекомендуем использовать самые последние версии стандартов и спецификаций.

# 18. Аутентификация пользователей. Веб-сервер Apache.

Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д. Поддерживает IPv6.

Apache HTTP-сервер — свободный веб-сервер.

**"Аутентификацией"** называется процесс проверки подлинности пользователя, т.е. пользователь Вася Иванов, каким-то способом подтверждает, что за компьютером в настоящий момент находится действительно он, а не кто-нибудь другой. Для этого чаще всего используется пара логин-пароль, но для реально секретных ресурсов могут применяться и более надёжные способы, например, сканирование отпечатка пальца или сетчатки глаза.

**"Авторизация"** происходит после успешного входа пользователя в систему и определяет к каким именно ресурсам может получить доступ данный пользователь. Так, например, Вася Иванов может, иметь доступ к модерированию раздела **"Статьи"** на сайте **www.example.com**, но доступ к модерированию раздела **"Вопрос-Ответ"** для него будет закрыт.

Apache поддерживает 2 вида аутентификации: **Basic-аутентификацию** (Базовая аутентификация) и **Digest-аутентификацию** (Дайджест аутентификация). Подробно механизмы аутентификации описаны в **RFC 2617**.

Настраивается аутентификация либо в файле **httpd.conf**, либо в файле **.htaccess**. При настройке посредством **.htaccess**, необходимо сначала убедиться, что директива **AllowOverride** в файле **httpd.conf** разрешает получение настроек из файла **.htaccess**. Директива **AllowOverride** может принимать следующие значения:

**AllowOverride None** - файлы **.htaccess** игнорируются веб-сервером. Данное значение оказывает положительное влияние на быстродействие веб-сервера.

**AllowOverride All** - обрабатываются все без исключения директивы из файлов .htaccess.

**AllowOverride AuthConfig** - разрешены директивы аутентификации-авторизации, такие как **AuthName**, **AuthType**, **AuthUserFile**, **AuthGroupFile**, **Require** и т.д.

Так же, директива **AllowOverride** может принимать ряд других значений, но к аутентификации-авторизации они отношения не имеют.

1. Basic-аутентификация
2. Basic-аутентификация работает так - клиент шлёт запрос на доступ к защищённой области

Code:

GET /site/private HTTP/1.0

Сервер присылает ответ:

Code:

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Basic realm="private"

После получения ответа, клиентский браузер отображает окно, примерно такого вида [​IMG]​ куда нужно ввести логин и пароль. После ввода логина и пароля на сервер отправляется такой запрос

Code:

GET /site/private HTTP/1.0

Authorization: Basic YWRtaW46MTIzNDU=

В зависимости от правильности введённых данных сервер, либо разрешит доступ к защищаемому ресурсу, либо запретит.

Главным минусом Basic-аутентификации является то, что все данные пересылаются через сеть в открытом виде, т.е. любой желающий из твоей локалки может отснифать эти данные и поиметь доступ к тем же ресурсам, что и ты. Во избежание подобных казусов рекомендуется дополнительно использовать SSL-шифрование. Если ты внимательно посмотришь на последний пример, то заметишь, что строчка

Code:

YWRtaW46MTIzNDU=

, есть ни что иное как

Code:

admin:12345

, закодированное при помощи **Base64** алгоритма.

Теперь, попробуем замутить такую аутентификацию на практике. Для этого потребуется:

1. Создать файл с паролями.
2. Прописать защищаемый ресурс в конфигурацию Apache (в файле httpd.conf или в файле .htaccess).
3. Создать файл для работы с группами и настроить групповой доступ (этот пункт не является обязательным).
4. Создаём файл с паролями.

Для этой цели используется утилита **htpasswd**, входящая в стандартную поставку сервера Apache. Запускается она так:

Code:

htpasswd -c [путь к файлу с паролями] [имя пользователя]

Ключ **-с** используется при первом запуске **htpasswd** и указывает на то, что нужно создать новый файл. Если файл с таким именем уже существует - он будет затёрт.

**Пример:**

Code:

htpasswd -c /var/www/html/secret/.htpasswd adm

**Результат:**

После запуска команды, будет дважды запрошен пароль для пользователя adm, после чего будет создан файл **/var/www/html/secret/.htpasswd** содержащий такую строчку

Code:

adm:хеш пароля

Рассмотрим, остальные ключи **htpasswd**:

**-n** Результат работы **htpasswd** (имя пользователя:хеш пароля) будет выведен на экран, а не в файл.

**-p** Пароль хранится в виде обычного текста безо всякого шифрования. Данный формат поддерживается только в операционных системах Windows, Netware и BEOS.

**Пример:**

Code:

adm:12345

**-d** Хеш пароля вычисляется с использованием стандартной Unix-функции **CRYPT**. Это алгоритм шифрования по умолчанию. Используется только на \*nix серверах.

**Пример:**

Code:

adm:sCRPeQ4DXDQqg

John The Ripper определил хеш как Standard DES [24/32 4K]

**-m** Хеш пароля вычисляется по алгоритму MD5.

**Пример:**

Code:

adm:$apr1$kCYEN/..$Ii.SRRJ77C.bB2.nOSZHI1

**-s** Хеш пароля вычисляется по алгоритму SHA1.

**Пример:**

Code:

adm:{SHA}jLIjfQZ5yojbZGTqxg2pY0VROWQ=

**-D** Заданный пользователь удаляется из файла с паролями.

**+Прописываем защищаемый ресурс в конфигурацию Apache (файл httpd.conf).**

Теперь нужно прописать защищаемый ресурс в конфигурацию сервера Apache. Делается это при помощи следующих директив:

**AuthType** - тип аутентификации **Basic/Digest**.

**AuthName** - название для области, требующей авторизации. Это название появится в диалоговом окне ввода логина и пароля.

**AuthUserFile** - местонахождение файла с паролями.

**AuthGroupFile** - местонахождение файла групп.

**Require** - требования, необходимые для доступа к защищённой области. Так, значение **valid-user** позволит произвести авторизацию для любого пользователя, прописанного в файле **.htpasswd**.

Следующий код позволит нам защитить область **http://www.example.com/secret** от несанкционированного доступа.

**Пример:**

Code:

<Directory "/var/www/html/secret">

AuthType Basic

AuthName "Private Area"

AuthUserFile /var/www/html/secret/.htpasswd

Require valid-user

</Directory>

После внесения изменений в файл **httpd.conf** необходимо перезапустить демон **httpd**.

**+Прописываем защищаемый ресурс в конфигурацию Apache (файл .htaccess).**

Настройка конфигурации Apache через файл **.htaccess** проводится аналогично предыдущему способу c тем отличием, что изменения вносятся "на лету" и **httpd-демон**перезапускать не придётся, так как Apache заново считывает содержимое файла **.htaccess** при каждом обращении к защищённой области.

**Пример:**

Code:

AuthType Basic

AuthName "Private Area"

AuthUserFile /var/www/html/secret/.htpasswd

Require user ivanov petrov

Пример похож на предыдущий, но есть одно отличие. Доступ к секретной области сайта будет разрешён не для всех пользователей из файла **.htpasswd**, а только для пользователей **ivanov** и **petrov**.

**+Настраиваем групповой доступ**

Для большей гибкости в Apache был введён механизм групп, т.е. авторизация производится не на уровне отдельных пользователей, а на уровне групп пользователей. Для работы с группами создадим файл **.htgroup** такого вида

Code:

admins: ivanov petrov

users: sidorov vasechkin svistunov

т.е. у нас есть две группы пользователей **admins** и **users**. Членами группы **admins** являются пользователи **ivanov** и **petrov**, а членами группы **users** являются пользователи **sidorov**, **vasechkin** и **svistunov**.

**Пример:**

Code:

AuthType Basic

AuthName "Private Area"

AuthUserFile /var/www/html/secret/.htpasswd

AuthGroupFile /var/www/html/secret/.htgroup

Require group admins

В приведённом примере доступ к секретной области сайта имеют только члены группы **admins**.

2. Digest-аутентификация.

Digest-аутентификация представляет собой более продвинутый и сложный вид аутентификации, чем Basic-аутентификация. Главным отличием здесь является то, что логин-пароль пользователя пересылаются через сеть не в открытом виде, а шифруются по алгоритму MD5. Настройка Digest-аутентификации похожа на настройку Basic-аутентификации. Основные шаги остаются прежними:

1. Создать файл с паролями.
2. Прописать защищаемый ресурс в конфигурацию Apache (в файле httpd.conf или в файле .htaccess).
3. Создать файл для работы с группами и настроить групповой доступ (этот пункт не является обязательным).
4. Создаём файл с паролями.

Файл паролей создаётся при помощи стандартной утилиты **htdigest**

Code:

htdigest -c [путь к файлу с паролями] [название секретной области] [имя пользователя]

Ключ **-c** указывается при необходимости создать новый файл, а обязательный аргумент **[название секретной области]** - это наименование секретной области, которое позже будет указано в директиве **AuthName**.

**Пример:**

Code:

htdigest -c /var/www/html/secretplace/.htpasswd private adm

После запуска команды нужно будет ввести пароль и его подтверждение. В результате, мы получим файл **/var/www/html/secretplace/.htpasswd** похожего содержания

Code:

adm:private:58b799bb9a734fc5948ebb6cf5c25f3e

Хеш считается по алгоритму MD5.

**+Прописываем защищаемый ресурс в конфигурацию Apache (файл .htaccess).**

**Пример:**

Code:

AuthType Digest

AuthName private

AuthUserFile /var/www/html/secretplace/.htpasswd

Require user adm

**Результат:**

Доступ к ресурсу **http://www.example.com/secretplace** будет иметь только пользователь **adm**.

**+Настраиваем групповой доступ**

Создаем файл **.htgroups** следующего содержания

Code:

admins: ivanov petrov

users: sidorov vasechkin svistunov

**Пример:**

Code:

AuthType Digest

AuthName private

AuthUserFile /var/www/html/secretplace/.htpasswd

AuthGroupFile /var/www/html/secretplace/.htgroup

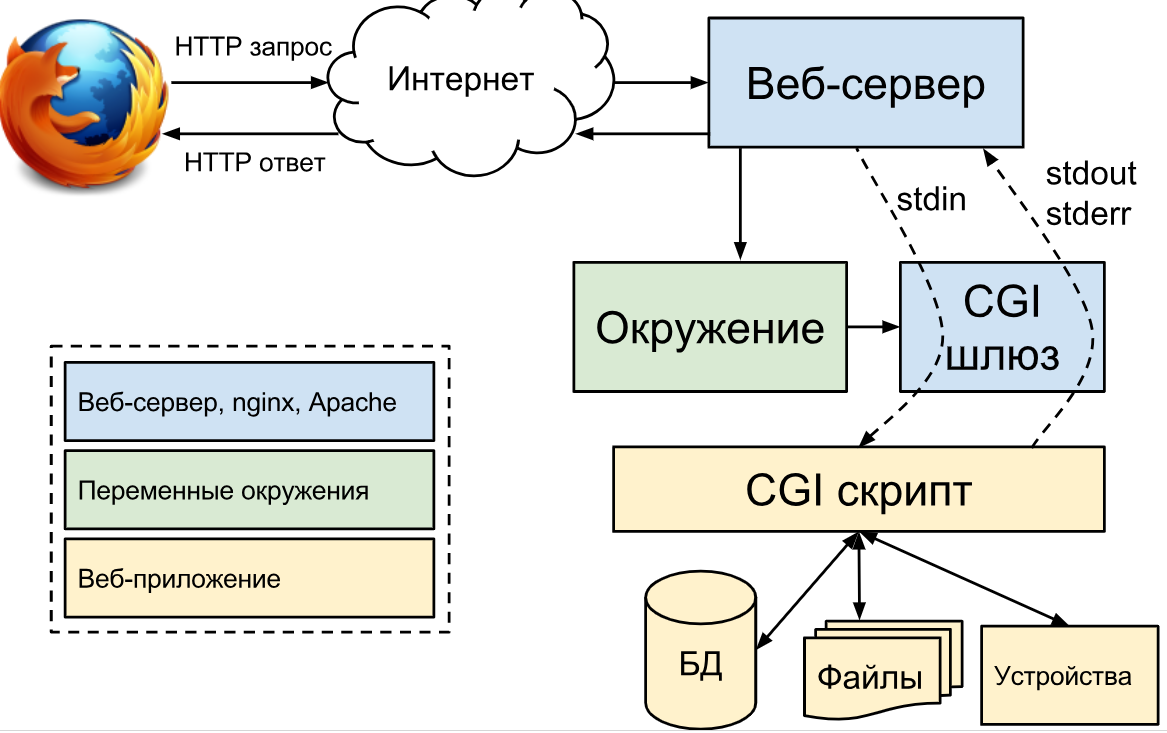
Require group admins

# 19. CGI-программирование.

**CGI** (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа».

Поскольку гипертекст статичен по своей природе, веб-страница не может непосредственно взаимодействовать с пользователем. До появления *JavaScript*, не было иной возможности отреагировать на действия пользователя, кроме как передать введенные им данные на веб-сервер для дальнейшей обработки. В случае CGI эта обработка осуществляется с помощью внешних программ и скриптов, обращение к которым выполняется через стандартизованный (см. RFC 3875: CGI Version 1.1) интерфейс — общий шлюз.

Упрощенная модель, иллюстрирующая работу CGI:



Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты.

**Как работает CGI?**

Обобщенный алгоритм работы через CGI можно представить в следующем виде:

1. Клиент запрашивает CGI-приложение по его URI.
2. Веб-сервер принимает запрос и устанавливает переменные окружения, через них приложению передаются данные и служебная информация.
3. Веб-сервер перенаправляет запросы через стандартный поток ввода (stdin) на вход вызываемой программы.
4. CGI-приложение выполняет все необходимые операции и формирует результаты в виде HTML.
5. Сформированный гипертекст возвращается веб-серверу через стандартный поток вывода (stdout). Сообщения об ошибках передаются через stderr.
6. Веб-сервер передает результаты запроса клиенту.

**Области применения CGI**

Наиболее частая задача, для решения которой применяется CGI — создание интерактивных страниц, содержание которых зависит от действий пользователя. Типичными примерами таких веб-страниц являются форма регистрации на сайте или форма для отправки комментария. Другая область применения CGI, остающаяся за кулисами взаимодействия с пользователем, связана со сбором и обработкой информации о клиенте: установка и чтение «печенюшек»-cookies; получение данных о браузере и операционной системе; подсчет количества посещений веб-страницы; мониторинг веб-трафика и т.п.

Эти возможности обеспечиваются тем, что CGI-скрипт может быть подключен к базе данных или обращаться к файловой системе сервера. Таким образом CGI-скрипт может сохранять информацию в таблицах БД или файлах и получать ее оттуда по запросу, чего нельзя сделать средствами HTML.

**Предупреждение**

CGI — это не язык программирования! Это простой протокол, позволяющий веб-серверу передавать данные через stdin и читать их из stdout. Поэтому, в качестве CGI-обработчика может использоваться любая серверная программа, способная работать со стандартными потоками ввода-вывода.

*Источник: https://lectureswww.readthedocs.io/5.web.server/cgi.html*

# 20. Язык серверных сценариев PHP.

PHP (Hypertext Preprocessor - Препроцессор Гипертекста)– это широко используемый язык сценариев общего назначения с открытым исходным кодом.

PHP - язык программирования, специально разработанный для написания web-приложений (скриптов, сценариев), исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на С и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить веб-мастер, не знающий пока других языков программирования.

Огромным плюсом PHP, в отличие от, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от скорости компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в html страницы.

PHP способен генерировать и преобразовывать не только HTML документы, но и изображения разных форматов - JPEG, GIF, PNG, файлы PDF и FLASH. PHP способен формировать данные в любом текстовом формате, включая XHTML и XML.

PHP - кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и многих других. PHP поддерживается на большинстве вебсерверов, таких, как Apache, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server и других.

Для большинства серверов PHP поставляется в 2-х вариантах - в качестве модуля и в качестве CGI препроцессора.

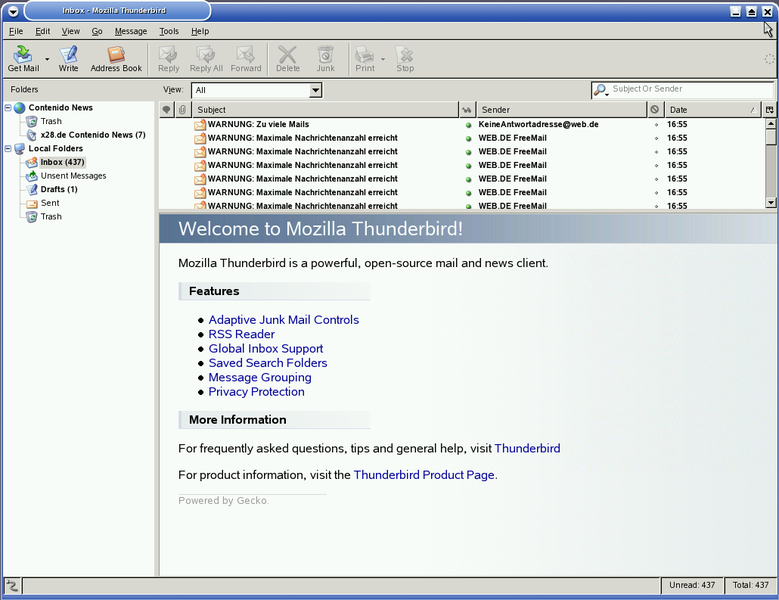
PHP поддерживает работу с ODBC и большое количество баз данных: MySQL, MSQL, Oracle, PostgreSQL, SQLite и др.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL - оптимальный вариант для создания интернет-сайтов различной сложности.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web -программирования.

# 21. Электронная почта.

Электро́нная по́чта (англ. email, e-mail [iˈmeɪl], от англ. electronic mail) — технология и служба по пересылке и получению электронных сообщений (называемых «письма», «электронные письма» или «сообщения») между пользователями компьютерной сети (в том числе — Интернета).



Электронная почта по составу элементов и принципу работы практически повторяет систему обычной (бумажной) почты, заимствуя как термины (почта, письмо, конверт, вложение, ящик, доставка и другие), так и характерные особенности — простоту использования, задержки передачи сообщений, достаточную надёжность и в то же время отсутствие гарантии доставки.

Достоинствами электронной почты являются: легко воспринимаемые и запоминаемые человеком адреса вида имя\_пользователя@имя\_домена (например, somebody@example.com); возможность передачи как простого текста, так и форматированного, а также произвольных файлов (текстовые документы, медиафайлы, программы, архивы и т. д.); независимость серверов (в общем случае они обращаются друг к другу непосредственно); достаточно высокая надёжность доставки сообщения; простота использования человеком и программами, высокая скорость передачи сообщений.

Недостатки электронной почты: наличие такого явления, как спам (массовые рекламные и вирусные рассылки); возможные задержки доставки сообщения (до нескольких суток); ограничения на размер одного сообщения и на общий размер сообщений в почтовом ящике (персональные для пользователей).

В настоящее время любой начинающий пользователь может завести свой бесплатный электронный почтовый ящик, достаточно зарегистрироваться на одном из интернет-порталов.

# 22. Почтовые клиенты.

**Почто́вая програ́мма** (*клиент электронной почты*, *почтовый клиент*, *мейл-клиент*, *мейлер*) — программное обеспечение, устанавливаемое на компьютере пользователя и предназначенное для получения, написания, отправки и хранения сообщений электронной почты одного или нескольких пользователей (в случае, например, нескольких учётных записей на одном компьютере) или нескольких учётных записей одного пользователя.

Функции

Основные функции почтовых клиентов — это прием сообщений, обеспечение их просмотра, сортировка сообщений, автоматизация создания ответных сообщений и поддержка адресной книги.

Большие почтовые программы, т. н. «всё в одной», такие как Mozilla Thunderbird, The Bat! и Microsoft Outlook, сегодня комбинируют работу MSA, MDA и MRA в одном приложении. Более простые почтовые агенты (англ. mail user agent, MUA), например Mutt, также являются почтовыми программами.

В отличие от почтового сервера, клиент электронной почты обычно отправляет сообщение не прямо на соответствующий сервер получателя, а на один и тот же почтовый сервер, который выступает как релей. Обычно это почтовый сервер провайдера или компании. Отправка почты чаще всего осуществляется по протоколу SMTP.

Клиент электронной почты принимает почту с одного или нескольких почтовых серверов, часто это тот же самый сервер, который служит для отправки. Приём почты обычно осуществляется по протоколам POP или IMAP.

Также в функции клиента электронной почты может входить: сортировка, хранение сообщений, поиск по архиву сообщений, ведение адресной книги, фильтрация принимаемых сообщений по разным критериям, конвертация форматов, шифрование, организация интерфейсов с офисными программами и другие функции.

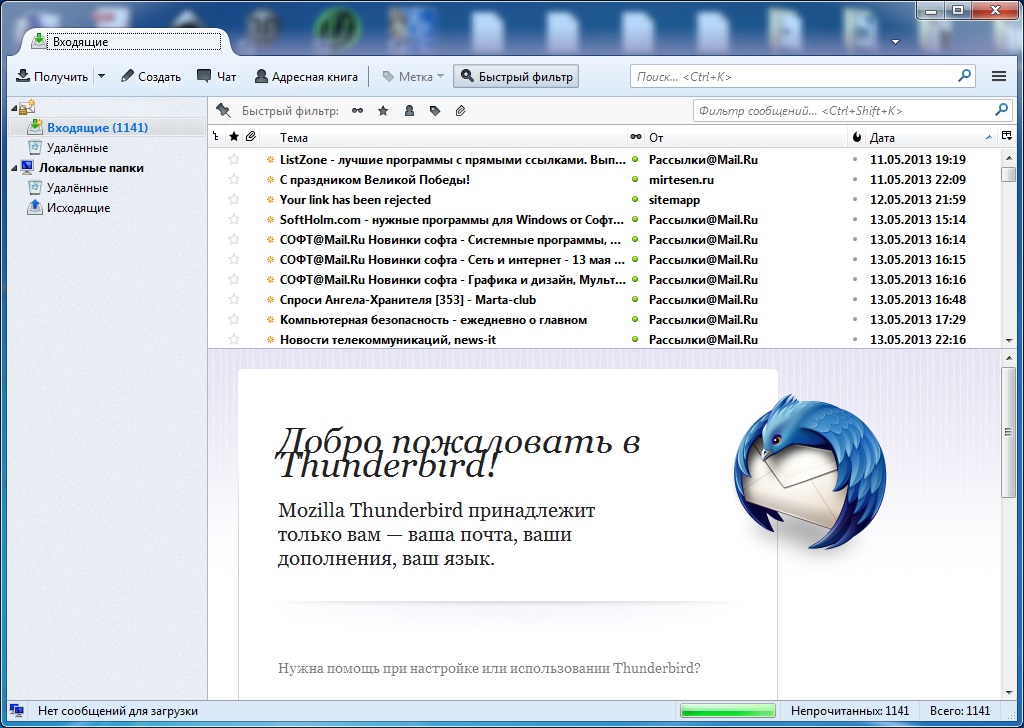
NNTP

Часто почтовые программы позволяют общаться в группах новостей (Usenet), основанных на технологии NNTP. Существуют NNTP-шлюзы в сеть Фидонет.

Программы

|  |  |  |
| --- | --- | --- |
| [**Microsoft Windows**](https://ru.wikipedia.org/wiki/Microsoft_Windows) | [**Unix-подобные ОС**](https://ru.wikipedia.org/wiki/Unix-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B5_%D0%9E%D0%A1) | **Мобильные телефоны с**[**J2ME**](https://ru.wikipedia.org/wiki/J2ME) |
| * [Becky! Internet Mail](https://ru.wikipedia.org/w/index.php?title=Becky!_Internet_Mail&action=edit&redlink=1) * [Eudora](https://ru.wikipedia.org/wiki/Eudora) * [Foxmail](https://ru.wikipedia.org/wiki/Foxmail) * [Free Mail Commander](https://ru.wikipedia.org/w/index.php?title=Free_Mail_Commander&action=edit&redlink=1) * [Koma Mail](https://ru.wikipedia.org/w/index.php?title=Koma_Mail&action=edit&redlink=1) * [Lotus Notes](https://ru.wikipedia.org/wiki/Lotus_Notes) * [Mailbird](https://ru.wikipedia.org/w/index.php?title=Mailbird&action=edit&redlink=1) * [Microsoft Outlook](https://ru.wikipedia.org/wiki/Microsoft_Outlook) * [Microsoft Outlook Express](https://ru.wikipedia.org/wiki/Microsoft_Outlook_Express) (устарела) * [Mozilla Thunderbird](https://ru.wikipedia.org/wiki/Mozilla_Thunderbird) * [Opera Mail](https://ru.wikipedia.org/wiki/Opera_Mail) * [Pegasus Mail](https://ru.wikipedia.org/w/index.php?title=Pegasus_Mail&action=edit&redlink=1) * [Popcorn](https://ru.wikipedia.org/wiki/Popcorn) * [Postbox](https://ru.wikipedia.org/w/index.php?title=Postbox&action=edit&redlink=1) * [si.Mail](https://ru.wikipedia.org/w/index.php?title=Si.Mail&action=edit&redlink=1) * [Sylpheed](https://ru.wikipedia.org/wiki/Sylpheed) * [The Bat!](https://ru.wikipedia.org/wiki/The_Bat!) * [The Bee](https://ru.wikipedia.org/w/index.php?title=The_Bee&action=edit&redlink=1) * [Windows Mail](https://ru.wikipedia.org/wiki/Windows_Mail) * [Zimbra Desktop](https://ru.wikipedia.org/w/index.php?title=Zimbra_Desktop&action=edit&redlink=1) | * [Apple Mail](https://ru.wikipedia.org/wiki/Apple_Mail) * [KMail](https://ru.wikipedia.org/wiki/KMail) (в среде KDE) * [Evolution](https://ru.wikipedia.org/wiki/Evolution) * [Mozilla Thunderbird](https://ru.wikipedia.org/wiki/Mozilla_Thunderbird) * [Mutt](https://ru.wikipedia.org/wiki/Mutt) * [Claws Mail](https://ru.wikipedia.org/wiki/Claws_Mail) * [Lotus Notes](https://ru.wikipedia.org/wiki/Lotus_Notes) * [Sylpheed](https://ru.wikipedia.org/wiki/Sylpheed) * [gnus](https://ru.wikipedia.org/wiki/Gnus) (в GNU EMACS) * [slrn](https://ru.wikipedia.org/wiki/Slrn) * [Pine](https://ru.wikipedia.org/wiki/Pine) | * [MailMan](https://ru.wikipedia.org/w/index.php?title=MailMan&action=edit&redlink=1) * [Email Viewer](https://ru.wikipedia.org/w/index.php?title=Email_Viewer&action=edit&redlink=1) |

Mozilla Thunderbird - Почтовая программа



# 23. Управление протоколами обмена почтовыми сообщениями.

Электронная почта существуют уже более двух десятилетий: до 1990 года она использовалась преимущественно в научных организациях, в 90-е - получила широкую известность и стала использоваться повсеместно. По самым скромным оценкам в мире более 50 миллионов человек пользуются услугами электронной почты. В целом же трафик электронной почты занимает только 3.7% всего сетевого. Как и любой форме коммуникаций, электронной почте присущ определенный стиль и набор соглашений. В частности, общение по электронной почте носит неформальный и демократичный характер.

Электронная почта даёт возможность посылать и получать сообщения, отвечать на письма корреспондентов автоматически, используя их адреса, рассылать копии письма сразу нескольким получателям, переправлять полученное письмо по другому адресу, использовать вместо адресов (числовых или доменных имен) логические имена, создавать несколько подразделов почтового ящика для разного рода корреспонденции, включать в письма текстовые файлы, пользоваться системой "отражателей почты" для ведения дискуссий с группой ваших корреспондентов и т.д.

Развитие технологии Internet привело к появлению современных протоколов для обмена сообщениями, которые предоставляют большие возможности для обработки писем, разнообразные сервисы и удобство в работе. Так, например, протокол SMTP, работающий по принципу клиент-сервер, предназначен для отправки сообщений с компьютера к адресату. Обычно доступ к серверу SMTP не защищается паролем, так что можно использовать для отправки писем любой известный сервер в сети. В отличие от серверов для отправки писем, доступ к серверам для хранения сообщений защищается паролем. Поэтому необходимо использовать сервер или службу, в которой существует учётная запись. Эти серверы работают по протоколам POP и IMAP, которые различаются способом хранения писем.

В соответствии с протоколом POP3 поступающие на определенный адрес сообщения хранятся на сервере до того момента, пока они не будут в течение очередного сеанса загружены на компьютер. После загрузки сообщений, можно отключиться от сети и приступить к чтению почты. Таким образом, использование почты по протоколу POP3 является наиболее быстрым и удобным в использовании.

Протокол IMAP удобен тем людям, которые пользуются постоянным подключением к сети. Сообщения, поступившие на адрес, также хранятся на сервере, но, в отличие от POP3, при проверке почты сначала будут загружены только заголовки сообщений. Само письмо можно будет прочитать после выбора заголовка сообщения (оно загрузиться с сервера). Ясно, что при коммутируемом соединении работа с почтой по этому протоколу приводит к неоправданным потерям времени.

Существует несколько протоколов приема передачи почты между многопользовательскими системами.

Краткое описание некоторых из них:   
**SMTP (Simple Mail Transfer Protocol)** - это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP, причем передача должна быть обязательно инициирована самой передающей системой.   
**POP, POP2, POP3 (Post Office Protocol)** - три достаточно простых невзаимозаменяемых протокола, pазpаботанные для доставки почты пользователю с центрального mail-сеpвеpа, ее удаления с него и для идентификации пользователя по имени/паролю. POP включает в себя SMTP, который используется для передачи почты, исходящей от пользователя. Почтовые сообщения могут быть получены в виде заголовков, без получения письма целиком.   
**IMAP2, IMAP2bis, IMAP3, IMAP4, IMAP4rev1 (Internet Message Access Protocol).**

* + **IMAP** осуществляет хранение почты на сеpвеpе в файловых диpектоpиях, а также предоставляет клиенту возможность производить поиск строк в почтовых сообщениях на самом сеpвеpе.
  + **IMAP2** - используется в редких случаях.
  + **IMAP3** - несовместимое ни с чем pешение, не используется.
  + **IMAP2bis** - pасшиpение IMAP2, позволяет сеpвеpам pазбиpаться в MIME-стpуктуpе (Multipurpose Internet Mail Extensions) сообщения, используется до сих пор.
  + **IMAP4** - пеpеpаботанный и pасшиpенный IMAP2bis, который можно использовать где угодно.
  + **IMAP4rev1** - pасшиpяет IMAP большим набором функций, включая те, котоpые используются в DMSP (Distributed Mail System for Personal Computers).

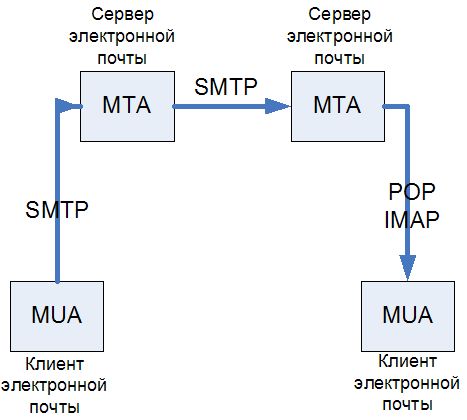
**ACAP (Application Configuration Access Protocol)** - протокол, pазpаботанный для работы с IMAP4; добавляет возможность поисковой подписки и подписки на доски объявлений, почтовые ящики и используется для поиска адресных книг.

**DMSP (или PCMAIL)** - протокол для приема/отправки почты, особенность которого заключается в том, что пользователь может иметь более одной рабочей станции в своем пользовании. Рабочая станция содержит статусную информацию о почте, диpектоpию, через которую происходит обмен, которая при подключении к серверу обновляется до текущего состояния на mail-сеpвеpе.

**MIME** - стандарт, определяющий механизмы для отправки разного рода информации с помощью электронной почты, включая текст на языках, отличных от английского, для которых используются символьные кодировки, отличные от ASCII, а также 8-битный бинарный контент, такой как картинки, музыка, фильмы и программы.

1. Схема работы почтовых протоколов стека TCP/IP
   * *SMTP*

В Интернете для доставки электронной почты машина-источник устанавливает TCP-соединение с портом 25 машины приемника. Этот порт прослушивается почтовым демоном, и их общение происходит с помощью протокола SMTP (Simple Mail Transfer Protocol простой протокол электронной почты). Этот демон принимает входящие соединения и копирует сообщения из них в соответствующие почтовые ящики. Если письмо не возможно доставить, отправителю отправляется сообщение об ошибки, содержащее первую часть этого письма.



**MTA (Mail Transfer Agent)** - агент передачи почты - является основным компонентом системы передачи почты Internet, который представляет данный сетевой компьютер для сетевой системы электронной почты. Обычно пользователи работают не с MTA, а с программой **MUA (Mail User Agent)** - клиентом электронной почты. Схематично принцип взаимодействия показан на рисунке.

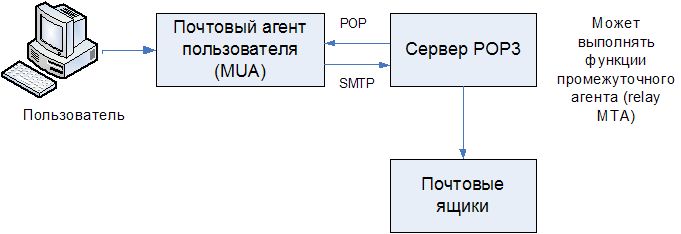
Протокол SMTP представляет собой простой ASCII протокол. Установив ТСР-соединение с портом 25. передающая машина выступающая в роли клиента, ждет запроса принимающей машины, работающей в режиме сервера. Сервер начинает диалог с того что посылает текстовую строку, содержащую его идентификатор и сообщающую о его готовности (или неготовности) к приему почты. Если сервер не готов, клиент разрывает соединение и продолжает попытку позднее.

Если сервер готов принимать почту, клиент объявляет, от кого поступила почта и кому она предназначается. Если получатель почты существует, сервер дает клиенту добро на пересылку сообщения. Затем клиент посылает сообщение. А сервер подтверждает его получение. Контрольные суммы не проверяются, так как протокол ТСР обеспечивает надежный байтовый поток. Если у отправителя есть еще почта. Она также отправляется. После передачи всей почты в обоих направлениях соединение разрывается.

*POP*

**Post Office Protocol Version 3** - протокол почтового отделения, версия 3 - это сетевой протокол, используемый почтовым клиентом для получения сообщений электронной почты с сервера. Обычно используется в паре с протоколом SMTP.

Предыдущие версии протокола (POP, POP2) устарели. Альтернативным протоколом для сбора сообщений с почтового сервера является IMAP. По умолчанию использует TCP-порт 110. Существуют реализации POP3-серверов, поддерживающие TLS и SSL.



После установки соединения протокол РОР3 проходит три последовательных состояния

1. Авторизация клиент проходит процедуру аутентификации
2. Транзакция клиент получает информацию о состоянии почтового ящика, принимает и удаляет почту.
3. Обновление сервер удаляет выбранные письма и закрывает соединение.

Не смотря на то, что протокол РОР3 действительно поддерживает возможность получения одного или нескольких писем и оставления их на сервере, большинство программ обработки электронной почты просто скачивают все письма и опустошают почтовый ящик на сервере.

* 1. *IMAP*

Пользователю, имеющему одну учетную запись у одного провайдера и всегда соединяющегося с провайдером с одной и той же машины, вполне достаточно протокола РОР3. Этот протокол используется повсеместно благодаря его простоте и надежности. Но у многих пользователей есть одна учетная запись в учебном заведении или на работе, но они хотят иметь к ней доступ и из дома, и с места работы (учебы), и во время командировки те из разных мест. Хотя протокол РОР3 и позволяет разрешить такую ситуацию. Но проблема в том что при таком использовании электронной почты вся корреспонденция пользователя очень быстро распространится по случайным машинам, с которых он получал доступ в Интернет, и некоторые из этих компьютеров могут вообще не принадлежать пользователь.

Это неудобство привело к созданию альтернативного протокола для получения почты, IMAP.

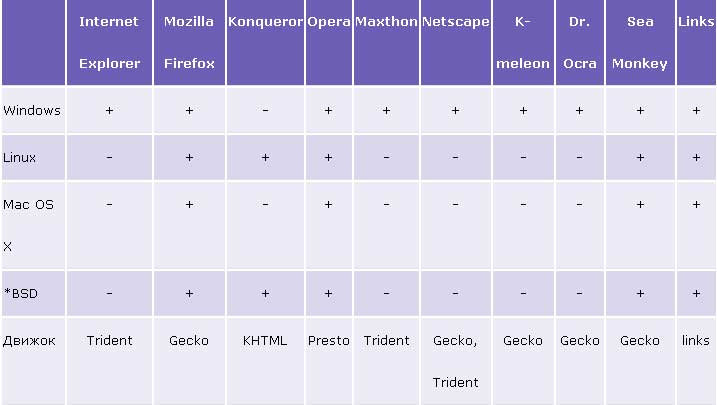
**IMAP (англ. Internet Message Access Protocol)** - интернет-протокол прикладного уровня для доступа к электронной почте.

IMAP предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем.

IMAP был разработан для замены более простого протокола POP3 и имеет следующие преимущества по сравнению с последним:

* 1. Письма хранятся на сервере, а не на клиенте. Возможен доступ к одному и тому же почтовому ящику с разных клиентов. Поддерживается также одновременный доступ нескольких клиентов. В протоколе есть механизмы с помощью которых клиент может быть проинформирован об изменениях, сделанных другими клиентами.
  2. Поддержка нескольких почтовых ящиков (или папок). Клиент может создавать, удалять и переименовывать почтовые ящики на сервере, а также перемещать письма из одного почтового ящика в другой.
  3. Возможно создание общих папок, к которым могут иметь доступ несколько пользователей.
  4. Информация о состоянии писем хранится на сервере и доступна всем клиентам. Письма могут быть помечены как прочитанные, важные и т. п.
  5. Поддержка поиска на сервере. Нет необходимости скачивать с сервера множество сообщений для того чтобы найти одно нужное.
  6. Поддержка онлайн-работы. Клиент может поддерживать с сервером постоянное соединение, при этом сервер в реальном времени информирует клиента об изменениях в почтовых ящиках, в том числе о новых письмах.
  7. Предусмотрен механизм расширения возможностей протокола.

Текущая версия протокола имеет обозначение IMAP4rev1 (IMAP, версия 4, ревизия 1). Протокол поддерживает передачу пароля пользователя в зашифрованном виде. Кроме того, IMAP-трафик можно зашифровать с помощью SSL.

1. Сводная таблица по почтовым клиентам и серверам для различных ОС.  
   

*Источник: http://latysheva2007.narod.ru/theme13.html*

# 24. Управлением клиентами почтового сервера.

При создании учётной записи в любом почтовом клиенте необходимо использовать следующие настройки:

**Настройки нешифрованного соединения**

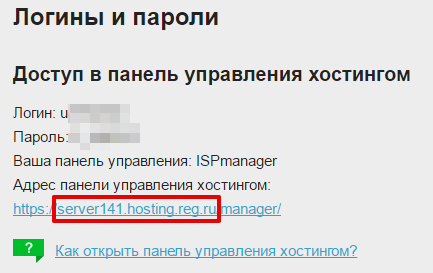
1. почтовый адрес: **name@ваш.домен**, например **info@mysite.ru**;
2. учётная запись (логин): **name@ваш.домен**, например **info@mysite.ru**;
3. пароль: **«пароль от почтового ящика»**;
4. сервер входящей почты: **mail.ваш.домен**, например **mail.mysite.ru**;
   * для настройки **POP3** используйте порт **110**;
   * для настройки **IMAP** используйте порт **143**;
5. сервер исходящей почты: **mail.ваш.домен**, например **mail.mysite.ru**;
   * для настройки **SMTP** используйте **25** или **587** порт.

**Настройки шифрованного SSL/TLS соединения**

Настройка шифрованного соединения доступна только на хостинге Linux.

1. почтовый адрес: **name@ваш.домен**, например **info@mysite.ru**;
2. учётная запись (логин): **name@ваш.домен**, например **info@mysite.ru**;
3. пароль: **«пароль от почтового ящика»**;
4. сервер входящей почты: **«имя сервера хостинга»**, например **server104.hosting.reg.ru**;
   * для настройки **POP3** используйте порт **995**;
   * для настройки **IMAP** используйте порт **993**;
5. сервер исходящей почты: **«имя сервера хостинга»**, например **server104.hosting.reg.ru**;
   * для настройки **SMTP** используйте **465** порт.

**Как узнать «имя сервера хостинга»?**

«Имя сервера хостинга» можно узнать в личном кабинете в разделе Информация о включённых сервисах и паролях доступа. Имя сервера указано в блоке **«Доступ в панель управления хостингом»**:

Также имя сервера можно увидеть в адресной строке браузера, когда вы находитесь в панели управления хостингом.

# 25. Администрирование пользователей и контактов.

Пользователи почты сходны с почтовыми контактами. И первые, и вторые имеют внешние адреса электронной почты, а также содержат информацию о людях, находящихся вне организации Exchange Online, и отображаются в общей адресной книге, а также в других списках адресов. Однако почтовый пользователь, в отличие от почтового контакта, имеет учетные данные для входа в организацию Exchange или Office 365 и обладает правом доступа к ресурсам.

**Создание пользователя почты**

**Использование Центра администрирования Exchange (EAC) для создания пользователя почты**

1. В Центре администрирования Exchange (EAC) выберите **Получатели** > **Контакты** > **Новый** > **Пользователь почты**.
2. На странице **Новый пользователь почты** в поле **\* Псевдоним** введите псевдоним пользователя почты. Длина псевдонима не должна превышать 64 символа. Псевдоним должен быть уникальным в лесу. Это поле является обязательным.
3. Чтобы указать тип адреса электронной почты для пользователя почты, выполните одно из следующих действий.
   * Чтобы указать адрес SMTP электронной почты для внешнего адреса электронной почты пользователя почты, щелкните **SMTP**.

**Примечание**

Exchange проверяет правильность формата адресов SMTP. Если запись не соответствует формату SMTP, на экран будет выведено сообщение об ошибке после нажатия кнопки **Сохранить**, чтобы создать пользователя почты.

* + Чтобы указать настраиваемый тип адреса, щелкните переключатель и введите тип настраиваемого адреса. Например, можно указать адрес X.500, GroupWise или Lotus Notes.

1. В поле **\* Внешний адрес электронной почты** введите внешний адрес электронной почты пользователя почты. Сообщения электронной почты, отправляемые данному почтовому пользователю, переадресуются на этот адрес электронной почты. Это поле является обязательным.
2. Для этого выберите один из приведенных ниже вариантов.
   * **Существующий пользователь**. Выберите, чтобы включить поддержку почты для существующего пользователя.

Нажмите кнопку **Обзор**, чтобы открыть диалоговое окно **Выбор пользователя  весь лес**. В данном диалоговом окне отображается список учетных записей пользователей в данной организации, для которых не включена поддержка почты или у которых нет почтовых ящиков. Выберите учетную запись пользователя, для которой необходимо включить поддержку почты, а затем нажмите кнопку **ОК**. При выборе этого параметра не требуется указывать данные учетной записи пользователя, потому что они уже присутствуют в службе Active Directory.

* + **Новый пользователь**. Выберите, чтобы создать новую учетную запись пользователя в службе Active Directory, а также чтобы включить поддержку почты для этого пользователя. В случае выбора этой опции необходимо указать необходимые данные учетной записи пользователя.

1. Если в действии 5 выбран пункт **Новый пользователь**, на странице **Новый пользователь почты**необходимо заполнить следующие поля. В противном случае перейдите к шагу 7.
   * **Имя**. В этом поле можно ввести имя пользователя почты.
   * **Инициалы**. В этом поле можно ввести инициалы пользователя почты.
   * **Фамилия**. В этом поле можно ввести фамилию пользователя почты.
   * **\* Отображаемое имя** Поле для ввода отображаемого имени пользователя. Это имя указывается в списке контактов Центра администрирования Exchange и адресной книге вашей организации. По умолчанию в это поле добавляются имена, введенные в поля **Имя**, **Инициалы** и **Фамилия**. Если эти поля не используются, все равно необходимо ввести имя в это поле, поскольку оно является обязательным для заполнения. Длина имени не может превышать 64 символов.
   * **\* Имя**. В этом поле можно ввести имя пользователя почты. Это имя, которое указано в службе каталогов. В это поле также добавляются имена, введенные в поля **Имя**, **Отчество** и **Фамилия**. Если эти поля не используются, все равно необходимо ввести имя в это поле, поскольку оно является обязательным для заполнения. Кроме того, длина имени не может превышать 64 символа.

**Примечание**

В поле **имя** доступен только в Exchange Server. Он недоступен в Exchange Online.

* + **Подразделение** Можно выбрать другое подразделение, кроме указанного по умолчанию (которое расположено в области получателя). Если область получателей установлена на уровне леса, то значение по умолчанию настроено в контейнере пользователей в домене, содержащем компьютер под управлением Центра администрирования Exchange. Если область получателей настроена на определенный домен, то контейнер пользователей в этом домене выбирается по умолчанию. Если область получателей настроена на определенное подразделение организации, то оно выбирается по умолчанию.

Нажмите кнопку **Обзор**, чтобы выбрать другое подразделение. В этом диалоговом окне отображаются все подразделения в лесу, которые находятся внутри определенной области. Выберите нужное подразделение и нажмите кнопку **ОК**.

**Примечание**

В поле **подразделение** доступен только в Exchange Server. Он недоступен в Exchange Online.

* + **\* Имя пользователя для входа**. В этом поле можно ввести имя, которое пользователь почты будет использовать для входа в домен. Имя для входа у пользователя состоит из псевдонима пользователя, который находится слева от знака @, и суффикса справа от этого знака. Обычно в качестве суффикса выступает имя домена, в котором размещается учетная запись пользователя.

**Примечание**

В Exchange Online это поле имеет метку **ИД пользователя**.

* + **\* Новый пароль**. В этом поле можно ввести пароль, который пользователь почты должен использовать для входа в домен.

**Примечание**

Убедитесь, что введенный пароль соответствует требованиям к длине, сложности и журналу паролей, предъявляемым для домена, в котором создается учетная запись пользователя.

* + **\* Подтверждение пароля** В этом поле нужно подтвердить пароль, введенный в поле **Пароль**.
  + **Потребовать смены пароля при следующем входе в систему**. Установите этот флажок, если необходимо, чтобы пользователи почты сменили пароль при первом входе в домен.

Если этот флажок установлен, при первом входе нового пользователя почты будет отображено диалоговое окно, в котором необходимо изменить пароль. Пользователь почты не сможет выполнять никакие задачи, пока пароль не будет успешно изменен.

1. Когда вы закончили, нажмите кнопку **Сохранить**, чтобы создать пользователя почты.

# 26. Безопасность почтовых сообщений.

Электронная почта небезопасна, данные и метаданные могут многое о вас рассказать, и от вторжения в личную жизнь спастись нереально. Но все не так безнадежно. Как сказал Эдвард Сноуден: «Времена, когда защита частной жизни граждан полностью зависела от государства, остались в прошлом. Мы больше не обязаны просить у правительства неприкосновенности частной жизни, это наше право».

«Все разведывательные службы – абсолютно все – боятся простых и безопасных коммуникационных инструментов», – говорит Джейкоб Эпплбаум.

Асимметричное шифрование – самое надежное и простое решение, но можно сделать и еще кое-что. Достаточно выполнить несколько несложных шагов, чтобы значительно повысить уровень защищенности email-коммуникаций.

Шаг 1. Шифрование

Pretty Good Privacy (PGP) – программа, позволяющая превращать содержание писем в бессмыслицу для всех, кроме отправителя и получателя. Подходит для некоторых весьма простых в использовании почтовых клиентов.

Шаг 2. Не стоит пользоваться американскими почтовиками

При выборе почтового сервиса стоит учитывать и географический фактор, но не стоит всецело на него полагаться. Например, в Германии и Швейцарии законы, защищающие частную жизнь граждан, более эффективны, чем в США или Великобритании. Поэтому безопаснее использовать почтовые клиенты других стран, например, корейский Naver.

Шаг 3. Не следует доверять почтовому провайдеру

Отказ от американских почтовых сервисов – это хорошее начало, но в идеале следует использовать почтовые провайдеры с нулевым разглашением информации. Нулевое разглашение означает, что сервер не имеет доступа к исходному тексту данных.

Шаг 4. Почта на своем сервере (это не так уж сложно)

Запустить свой собственный почтовый сервер не так трудно, как кажется. Таким образом, можно снизить риск взлома email-аккаунта. Это значит, что сам пользователь будет администрировать свой почтовый ящик, и если кому-то понадобится получить его содержимое, то им придется создавать бэкдор для этого конкретного сервера. Вероятность того, что кому-то понадобитесь именно вы, не особенно велика.

В заключение, несколько решений для повышения уровня защищенности электронной почты.

* Mozilla thunderbird с расширением Enigmail;
* Почтовый клиент Mailpile;

# 27. Аутентификация пользователей, интеграция с Active Directory.

Аутентификация пользователя – это проверка, действительно ли проверяемый пользователь является тем, за кого он себя выдает. Различные методы аутентификации необходимы, фактически, во всех системах ограничения и разграничения доступа к данным – как распределенных, так и предназначенных для защиты отдельного компьютера.

Active Directory («Активный каталог», AD) — службы каталогов корпорации Microsoft для операционных систем семейства Windows Server. Первоначально создавалась, как LDAP-совместимая реализация службы каталогов, однако, начиная с Windows Server 2008, включает возможности интеграции с другими службами авторизации, выполняя для них интегрирующую и объединяющую роль. Позволяет администраторам использовать групповые политики для обеспечения единообразия настройки пользовательской рабочей среды, разворачивать программное обеспечение на множестве компьютеров через групповые политики или посредством System Center Configuration Manager (ранее — Microsoft Systems Management Server), устанавливать обновления операционной системы, прикладного и серверного программного обеспечения на всех компьютерах в сети, используя Службу обновления Windows Server. Хранит данные и настройки среды в централизованной базе данных. Сети Active Directory могут быть различного размера: от нескольких десятков до нескольких миллионов объектов.

Аутентификация Active Directory может быть настроена через аутентификацию auth: LDAP, однако это сделать гораздо проще с помощью специально созданного расширения DokuWIki. Для взаимодействия с AD в DokuWiki используется отличная библиотека adLDAP, которая основана на работе James Van Lommel. Эта библиотека включена в установку DokuWiki, так что нет необходимости её отдельно загружать и устанавливать. И ко всему прочему эта библиотека позволяет использовать простую (Single-Sign-On) авторизацию NTLM и Kerberos. Для начала настройки необходимо включить предустановленный плагин.

Серверное расширение включено в дистрибутив DokuWiki начиная с версии rc2009-12-02 «Mulled Wine».

Также будет необходимо включить поддержку LDAP на PHP сервере. По умолчанию PHP не поддерживает LDAP. В большинстве случаев нужно просто отредактировать php.ini чтобы включить поддержку LDAP, однако надо проверить установку PHP, если на вашем сервере установлена нестандартная сборка PHP.

**Настройка**

Во избежании потери данных при перезаписи их менеджером настройки рекомендуется все настройки производить в файле conf/local.protected.php.

Возможно потребуется установить как минимум следующие опции:

**<?php**

*// общие настройки DokuWiki*

$conf['useacl'] = 1;

$conf['disableactions'] = 'register';

$conf['authtype'] = 'ad';

*// настройки Active Directory*

$conf['auth']['ad']['account\_suffix'] = '@my.domain.org';

$conf['auth']['ad']['base\_dn'] = 'DC=my,DC=domain,DC=org';

$conf['auth']['ad']['domain\_controllers'] = 'srv1.domain.org, srv2.domain.org'; *//через запятую можно перечислить*

*//несколько контроллеров домена*

Можно указать дополнительные параметры:

$conf['auth']['ad']['ad\_username'] = 'root';

$conf['auth']['ad']['ad\_password'] = 'pass';

$conf['auth']['ad']['sso'] = 1;

$conf['auth']['ad']['real\_primarygroup'] = 1;

$conf['auth']['ad']['use\_ssl'] = 1;

$conf['auth']['ad']['use\_tls'] = 1;

$conf['auth']['ad']['debug'] = 1;

$conf['auth']['ad']['recursive\_groups'] = 1; *// Если в AD содержится много групп, переключение этого параметра*

*// в 0 улучшит скорость работы, но неявное членство в группах перестанет работать*

ad\_username и ad\_password необходимы для реализации подписки на изменения. Этот аккаунт используется для запроса информации о пользователе из AD.

Для установки прав суперпользователя можно использовать следующую конструкцию:

$conf['manager'] = '@LDAPGROUPNAME';

$conf['superuser'] = '@LDAPGROUPNAME';

Любые другие настройки, указанные в $conf['auth']['ad'], напрямую передаются в библиотеку adldap. Детальное описание этих настроек можно получить в документации по adLDAP.

В комбинации с аутентификацией Single-Sign-On также можно добавить настройки домена Windows. То есть проводить аутентификацию на разных AD в зависимости от домена NTLM или Kerberos конкретного пользователя. Для этого надо использовать название домена (в нижнем регистре) как подключ в $conf['auth']['ad']. Т.е. для того, чтобы идентифицировать всех пользователей, пришедших из домена Windows Foobar через сервер AD, отличный от сервера по умолчанию, нужно добавить следующие строчки в конфигурационный файл:

$conf['auth']['ad']['foobar']['account\_suffix'] = '@foobar.domain.org';

$conf['auth']['ad']['foobar']['base\_dn'] = 'DC=foobar,DC=domain,DC=org';

$conf['auth']['ad']['foobar']['domain\_controllers'] = 'otherad.domain.org';

$conf['auth']['ad']['foobar']['ad\_username'] = 'otherroot';

$conf['auth']['ad']['foobar']['ad\_password'] = 'otherpass';

Если в организации используется система из нескольких контроллеров домена с единым родительским контроллером, может потребоваться указать порт 3268, вместо порта по умолчанию 389. В противном случае DokuWiki может не получить информации о группах пользователей дочернего домена. Самый простой способ сделать это - исправить исходники adLDAP.php, так как все вызовы к ldap\_connect содержат порт в качестве отдельного аргумента функции. 5

**Изменение профиля пользователя и пароля**

Пользователи могут изменить их настройки (имя, почтовый адрес и пароль) используя кнопку «Профиль». Это может потребовать выделения соответствующих привилегий для пользователя, от имени которого DokuWiki работает с AD (этот пользователь указан в настройках ad\_username и ad\_password). При этом изменение пароля доступно только через SSL или TLS соединение.

Также необходимо помнить, что автоматически создаваемые DokuWiki пароли не удовлетворяют политикам безопасности, использованным по умолчанию в Active Directory. В таком случае либо исправьте соответствующие политики AD, либо выключите функцию «Напомнить пароль» в DokuWiki в настройках config:disableactions.

**Имена пользователей и групп**

Имена пользователей и групп при экспорте автоматически преобразуются в формат DokuWiki и могут отличаться от указанных на сервере Active Directory. Пробелы заменяются символом подчёркивания, а символы \ и # удаляются.

Пример: Domain Users станет Domain\_Users в DokuWiki. Если необходимо отредактировать файл с ACLвручную, нужно помнить, что правильная форма: Domain%5fUsers. Где «%5f» символ подчёркивания.

Об этом необходимо помнить, указывая имена пользователей и групп при настройке прав доступа в DokuWiki.

**Включение Single Sign On**

Single Sign On (SSO) обозначает, что DokuWiki будет использовать логин пользователя, под которым работает пользователь Windows без необходимости введения логина и пароля на сайте. Эта функция использует настройки сервера и читает значение из переменной окружения REMOTE\_USER. При этом подсистема аутентификации через AD в DokuWiki будет использовать это имя для получения дополнительной информации, например информации о том, к каким группа пользователей относится имя.

Для того, чтобы эта функция заработала, необходимо включить настройку sso в local.protected.php и скорее всего настроить служебный аккаунт, добавив ему возможность получать информацию о пользователях AD:

$conf['auth']['ad']['sso'] = 1;

$conf['auth']['ad']['ad\_username'] = 'MyManager';

$conf['auth']['ad']['ad\_password'] = 'ManagerPass';

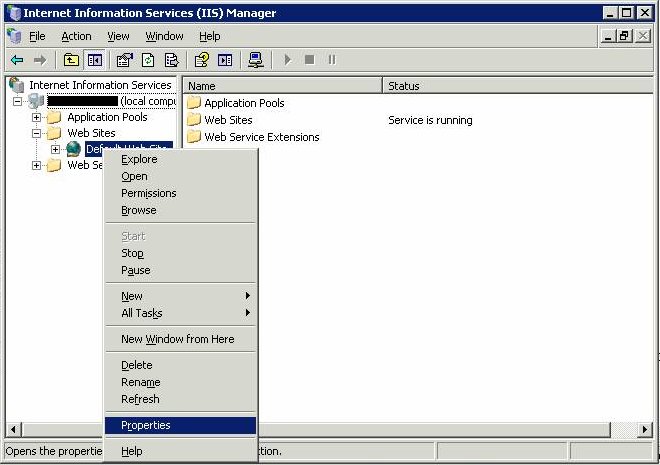
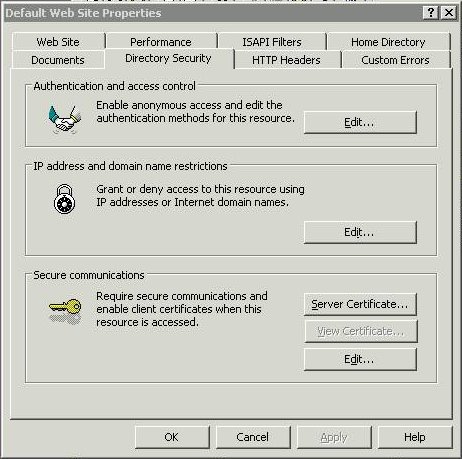
Также необходимо дополнительно установить дополнительные настройки на сервере и в браузере.

**Web Server**

**NTLM on IIS**

First configure IIS to use the Windows Logon for authentication (see screenshots):

1. Open the IIS configuration console using «Start» → «Run» → inetmgr
2. Right click on the «Default Web Site» entry and choose «Properties»
3. Switch to the «Directory Security» tab
4. Click the «Edit» button on «Anonymous access and authentication control».
5. Disable «Anonymous access»
6. Enable «Integrated Windows Authentication»

[](http://dokuwiki.ru/lib/exe/fetch.php?tok=d2dfda&media=http://www.dokuwiki.org/_media/auth:step1.jpg) [](http://dokuwiki.ru/lib/exe/fetch.php?tok=8be249&media=http://www.dokuwiki.org/_media/auth:step2.jpg) [](http://dokuwiki.ru/lib/exe/fetch.php?tok=cc42b5&media=http://www.dokuwiki.org/_media/auth:step3.jpg)

Then make sure NTLM is used as authentication protocol. This has to be done on the commandline:

1. Open a command line: «Start» → «Run» → cmd
2. Change to the admin script directory: cd \Inetpub\Adminscripts
3. Check the current protocol: cscript adsutil.vbs get w3svc/NTAuthenticationProviders
4. if it doesn't say NTLM, set it: cscript adsutil.vbs set w3svc/NTAuthenticationProviders «NTLM»

Now restart IIS.

**NTLM on Apache (Windows)**

Download <http://sourceforge.net/projects/mod-auth-sspi/> Copy the mod\_auth\_sspi.so file into your apache modules directory. Add into httpd.conf:

LoadModule sspi\_auth\_module modules/mod\_auth\_sspi.so

<Directory "c:/wamp/www/">

AuthName "My Intranet"

AuthType SSPI

SSPIAuth On

SSPIAuthoritative On

require valid-user

</Directory>

Now restart Apache

**NTLM on Apache (Linux)**

**Ubuntu Server 11.04**

**Скачиваем модуль mod\_ntlm для Apache.**

$ sudo wget http://modntlm.sourceforge.net/mod\_ntlm2.tar.gz

**Распаковываем архив:**

$ tar -zxvf mod\_ntlm2.tar.gz

**Необходимо обновить систему:**

$ sudo apt-get update

**Устанавливаем gcc:**

$ sudo apt-get install gcc

**Устанавливаем Apache prefork и gcc 3.4:**

$ sudo apt-get install apache2-prefork-dev gcc-3.4

**Линкуем компилятор gcc версии 3.4 с установленным в системе:**

$ ln -fs /usr/bin/gcc-3.4 /usr/bin/gcc

**Далее компилируем mod\_ntlm.c**

$ sudo apxs2 -i -a -c mod\_ntlm2/mod\_ntlm.c

**Далее, в /etc/apache2/mods-available создать файл ntlm.load и добавить в него строку с загрузкой модуля:**

LoadModule ntlm\_module /usr/lib/apache2/modules/mod\_ntlm.so

**Запустить модуль**

$ sudo a2enmod ntlm

**В файле /etc/apache2/httpd.conf добавляем:**

<Location />

AuthType NTLM

NTLMAuth On

NTLMAuthoritative OFF

NTLMDomain domain.ru

NTLMServer name-server.domain.ru

NTLMBackup name-server.domain.ru

require valid-user

</Location>

**Конфигурируем Apache2 /etc/apache2/apache2.conf**

KeepAlive On

MaxKeepAliveRequests 100

KeepAliveTimeout 15

**Перезапускаем Apache:**

$ sudo service apache2 restart

# 28. Борьба с вирусами.

Компьютерные вирусы. Что это такое и как с этим бороться? На эту тему уже написаны десятки книг и сотни статей, борьбой с компьютерными вирусами профессионально занимаются сотни (или тысячи) специалистов в десятках (а может быть, сотнях) компаний. Казалось бы, тема эта не настолько сложна и актуальна, чтобы быть объектом такого пристального внимания. Однако это не так. Компьютерные вирусы были и остаются одной из наиболее распространенных причин потери информации. Известны случаи, когда вирусы блокировали работу организаций и предприятий. Более того, несколько лет назад был зафиксирован случай, когда компьютерный вирус стал причиной гибели человека - в одном из госпиталей Нидерландов пациент получил летальную дозу морфия по той причине, что компьютер был заражен вирусом и выдавал неверную информацию.

Несмотря на огромные усилия конкурирующих между собой антивирусных фирм, убытки, приносимые компьютерными вирусами, не падают и достигают астрономических величин в сотни миллионов долларов ежегодно. Эти оценки явно занижены, поскольку известно становится лишь о части подобных инцидентов.

При этом следует иметь в виду, что антивирусные программы и «железо» не дают полной гарантии защиты от вирусов. Примерно так же плохо обстоят дела на другой стороне тандема «человек-компьютер». Как пользователи, так и профессионалы-программисты часто не имеют даже навыков «самообороны», а их представления о вирусе порой являются поверхностными. Что же такое компьютерный вирус?

***Компьютерные вирусы, их анализ и разработка методов обнаружения и лечения.***

Компьютерный вирус – это специально написанная программа, обязательным (необходимым) свойством которого является возможность создавать свои дубликаты (не обязательно совпадающие с оригиналом) и внедрять их в вычислительные сети и/или файлы, системные области компьютера и прочие выполняемые объекты. При этом дубликаты сохраняют способность к дальнейшему распространению.

***Классификация компьютерных вирусов***

Вирусы можно разделить на***классы*** по следующим основным признакам:

* среда обитания;
* операционная система (ОС);
* особенности алгоритма работы;
* деструктивные возможности.

по ***среде обитания*** вирусы можно разделить на:

* файловые;
* загрузочные;
* макро;
* сетевые.

***Файловые вирусы*** либо различными способами внедряются в выполняемые файлы (наиболее распространенный тип вирусов), либо создают файлы-двойники (компаньон-вирусы), либо используют особенности организации файловой системы (link-вирусы).

***Загрузочные вирусы*** записывают себя либо в загрузочный сектор диска (boot-сектор), либо в сектор, содержащий системный загрузчик винчестера (Master Boot Record), либо меняют указатель на активный boot-сектор.

***Макро-вирусы*** заражают файлы-документы и электронные таблицы нескольких популярных редакторов.

***Сетевые вирусы*** используют для своего распространения протоколы или команды компьютерных сетей и электронной почты.

Существует большое количество сочетаний - например, ***файлово-загрузочные*** вирусы, заражающие как файлы, так и загрузочные сектора дисков. Такие вирусы, как правило, имеют довольно сложный алгоритм работы, часто применяют оригинальные методы проникновения в систему, используют стелс и полиморфик-технологии. Другой пример такого сочетания - ***сетевой макро-вирус***, который не только заражает редактируемые документы, но и рассылает свои копии по электронной почте.

Заражаемая ***операционная система*** (вернее, ОС, объекты которой подвержены заражению) является вторым уровнем деления вирусов на классы. Каждый файловый или сетевой вирус заражает файлы какой-либо одной или нескольких OS - DOS, Windows, Win95/NT, OS/2 и т.д. Макро-вирусы заражают файлы форматов Word, Excel, Office97. Загрузочные вирусы также ориентированы на конкретные форматы расположения системных данных в загрузочных секторах дисков.

Среди ***особенностей алгоритма работы*** вирусов выделяются следующие пункты:

* резидентность;
* использование стелс-алгоритмов;
* самошифрование и полиморфичность;
* использование нестандартных приемов.

***Резидентный*** вирус при инфицировании компьютера оставляет в оперативной памяти свою резидентную часть, которая затем перехватывает обращения операционной системы к объектам заражения и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения компьютера или перезагрузки операционной системы. Нерезидентные вирусы не заражают память компьютера и сохраняют активность ограниченное время. Некоторые вирусы оставляют в оперативной памяти небольшие резидентные программы, которые не распространяют вирус. Такие вирусы считаются нерезидентными.

Резидентными можно считать макро-вирусы, посколько они постоянно присутствуют в памяти компьютера на все время работы зараженного редактора. При этом роль операционной системы берет на себя редактор, а понятие «перезагрузка операционной системы» трактуется как выход из редактора.

В многозадачных операционных системах время «жизни» резидентного DOS-вируса также может быть ограничено моментом закрытия зараженного DOS-окна, а активность загрузочных вирусов в некоторых операционных системах ограничивается моментом инсталляции дисковых драйверов ОС.

Использование ***стелс - алгоритмов*** позволяет вирусам полностью или частично скрыть себя в системе. Наиболее распространенным стелс-алгоритмом является перехват запросов ОС на чтение/запись зараженных объектов. Стелс-вирусы при этом либо временно лечат их, либо «подставляют» вместо себя незараженные участки информации. В случае макровирусов наиболее популярный способ — запрет вызовов меню просмотра макросов. Один из первых файловых стелс-вирусов — вирус «Frodo», первый загрузочный стелс-вирус — «Brain».

***Самошифрование и полиморфичность*** используются практически всеми типами вирусов для того, чтобы максимально усложнить процедуру детектирования вируса. Полиморфик-вирусы (polymorphic) - это достаточно трудно обнаружимые вирусы, не имеющие сигнатур, т.е. не содержащие ни одного постоянного участка кода. В большинстве случаев два образца одного и того же полиморфик-вируса не будут иметь ни одного совпадения. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика.

Различные ***нестандартные приемы*** часто используются в вирусах для того, чтобы как можно глубже спрятать себя в ядре ОС (как это делает вирус «ЗАРАЗА»), защитить от обнаружения свою резидентную копию (вирусы «ТРУО», «Trout2»)? затруднить лечение от вируса (например, поместив свою копию в Flash-BIOS) и т.д.

По ***деструктивным возможностям*** вирусы можно разделить на:

* безвредные, т.е. никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);
* неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и пр. эффектами;
* опасные вирусы, которые могут привести к серьезным сбоям в работе компьютера;
* очень опасные, в алгоритм работы которых, заведомо заложены процедуры, которые могут привести к потере программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти, и даже, как гласит одна из непроверенных компьютерных легенд, способствовать быстрому износу движущихся частей механизмов - вводить в резонанс и разрушать головки некоторых типов винчестеров.

Но даже если в алгоритме вируса не найдено ветвей, наносящих ущерб системе, этот вирус нельзя с полной уверенностью назвать безвредным, так как проникновение его в компьютер может вызвать непредсказуемые и порой катастрофические последствия. Ведь вирус, как и всякая программа, имеет ошибки, в результате которых могут быть испорчены как файлы, так и сектора дисков (например, вполне безобидный на первый взгляд вирус «DenZuk» довольно корректно работает с 360К дискетами, но может уничтожить информацию на дискетах большего объема). До сих пор попадаются вирусы, определяющие «СОМ или ЕХЕ» не по внутреннему формату файла, а по его расширению. Естественно, что при несовпадении формата и расширения имени файл после заражения оказывается неработоспособным. Возможно также «заклинивание» резидентного вируса и системы при использовании новых версий DOS, при работе в Windows или с другими мощными программными системами. И так далее. Загрузочные вирусы; Макровирусы; Полиморфик-вирусы; Прочие «вредные программы»; Резидентные вирусы; Сетевые вирусы; Стелс-вирусы; Файловые вирусы; IRC-черви.

***Методы обнаружения и удаления компьютерных вирусов***

При работе с современным персональным компьютером пользователя (а особенно начинающего) может подстерегать множество неприятностей: потеря данных, зависание системы, выход из строя отдельных частей компьютера и другие. Одной из причин этих проблем (наряду с ошибками в программном обеспечении) и неумелыми действиями самого оператора ПЭВМ, могут быть проникшие в систему компьютерные вирусы. Эти программы подобно биологическим вирусам размножаются, записываясь в системные области диска, или, приписываясь к файлам,  производят различные нежелательные действия, которые, зачастую, имеют катастрофические последствия. Чтобы не стать жертвой этой напасти, каждому пользователю следует хорошо знать принципы защиты от компьютерных вирусов.

Способы противодействия компьютерным вирусам можно разделить на несколько групп: профилактика вирусного заражения и уменьшение предполагаемого ущерба от такого заражения; методика использования антивирусных программ, в том числе обезвреживание и удаление известного вируса; способы обнаружения и удаления неизвестного вируса.

* Анализ алгоритма вируса
* Антивирусные программы
* Восстановление пораженных объектов
* Обнаружение неизвестного вируса
* Профилактика заражения компьютера

С давних времён известно, что к любому яду рано или поздно можно найти противоядие. Таким противоядием в компьютерном мире стали программы, называемые антивирусными. Данные программы можно классифицировать по пяти основным группам: фильтры, детекторы, ревизоры, доктора и вакцинагоры.

***Антивирусы-фильтры*** - это резидентные программы, которые оповещают пользователя о всех попытках какой-либо программы записаться на диск, а уж тем более отформатировать его, а также о других подозрительных действиях (например о попытках изменить установки CMOS). При этом выводится запрос о разрешении или запрещении данного действия. Принцип работы этих программ основан на перехвате соответствующих векторов прерываний. К преимуществу программ этого класса по сравнению с программами-детекторами можно отнести универсальность по отношению, как к известным, так и неизвестным вирусам, тогда как детекторы пишутся под конкретные, известные на данный момент программисту виды. Это особенно актуально сейчас, когда появилось множество вирусов-мутантов, не имеющих постоянного кода. Однако, программы-фильтры не могут отслеживать вирусы, обращающиеся непосредственно к BIOS, а также ВООТ-вирусы, активизирующиеся ещё до запуска антивируса, в начальной стадии загрузки DOS, К недостаткам также можно отнести частую выдачу запросов на осуществление какой-либо операции: ответы на вопросы отнимают у пользователя много времени и действуют ему на нервы. При установке некоторых антивирусов-фильтров могут возникать конфликты с другими резидентными программами, использующими те же прерывания, которые просто перестают работать.

Наибольшее распространение в нашей стране получили ***программы-детекторы***, а вернее программы, объединяющие в себе детектор и доктор. Наиболее известные представители этого класса - Aidstest, Doctor Web, Microsoft AntiVims, антивирус Касперского и др. .Антивирусы-детекторы рассчитаны на конкретные вирусы и основаны на сравнении последовательности кодов содержащихся в теле вируса с кодами проверяемых программ. Такие программы нужно регулярно обновлять, так как они быстро устаревают и не могут обнаруживать новые виды вирусов. Ревизоры - программы, которые анализируют текущее состояние файлов и системных областей диска и сравнивают его с информацией, сохранённой ранее в одном из файлов данных ревизора. При этом проверяется состояние BOOT-сектора, таблицы FAT, а также длина файлов, их время создания, атрибуты, контрольная сумма. Анализируя сообщения программы-ревизора, пользователь может решить, чем вызваны изменения: вирусом или нет. При выдаче такого рода сообщений не следует предаваться панике, так как причиной изменений, например, длины программы может быть вовсе и не вирус. Так был случай, когда один начинающий пользователь не на шутку перепутался, когда антивирус AVSP выдал ему сообщение об изменениях в файле CONFIG.SYS. Оказалось, что до этого на компьютер была осуществлена инсталляция менеджера памяти QEMM, который пишет свой драйвер в CONFIG.SYS.

К последней группе относятся самые ***антивирусы - вакцинаторы***. Они записывают в вакцинируемую программу признаки конкретного вируса так, что вирус считает ее уже зараженной.

Все эти программы универсальные и перспективные, сочетающие функции антивирусного сканера, резидентного сторожа и доктора.

В качестве перспективного подхода к защите от компьютерных вирусов в последние годы все чаще применяется сочетание программных и аппаратных методов защиты. Среди аппаратных устройств такого плана можно отметить специальные антивирусные платы, которые вставляются в стандартные слоты расширения компьютера.

Методы обнаружения вирусов

Обнаружение вирусов обычно происходит или при сканировании файлов для поиска известных вирусов, описанных в антивирусной базе, или при обнаружении подозрительного поведения любой из программ. Как правило, эти методы обнаружения вирусов реализованы отдельными модулями.

В первом случае антивирус, сканируя файл, сверяется с антивирусными базами, составленными производителем антивируса. Если какой-то участок кода файла соответствует коду вируса описанного в базе, антивирус предлагает выполнить следующие действия:

1. Удалить найденный зараженный файл.

2. Заблокировать доступ к этому файлу.

3. Отправить инфицированный файл в карантин (для недопущения дальнейшего распространения вируса).

4. Выполнить лечение зараженного файла (попытаться удалить тело вируса из файла).

5. Попросить выполнить лечение или удаление после перезапуска системы, если в данный момент это невозможно.

Во втором случае антивирус прослеживает поведение всех программ. Если программа пытается выполнить какое-то подозрительное действие - она блокируется. Антивирус информирует пользователя о подозрительном поведении программы.

Такие превентивные методы обнаружения вирусов дают возможность противостоять новым вирусам, которые еще не описаны в антивирусных базах.

Однако, некоторые антивирусы дают этому методу первостепенное значение, подчас запугивая и запутывая пользователя слишком большим количеством сообщений о подозрительном поведении программ.

Для того чтобы антивирус был максимально эффективен при обнаружении и лечении вирусов, он должен иметь возможность своевременно обновляться (обычно через Интернет).

Нет смысла ставить на один компьютер несколько антивирусов. Из-за постоянного подвисания это сделает работу за таким компьютером просто невозможной.

На сегодняшний день существует огромное количество антивирусных программ. Они сильно различаются и по цене, и по функционалу. Есть даже бесплатные антивирусы с неплохими возможностями.

Лечение вирусов

При условии, что у вас стоит одна из наиболее популярных антивирусных программ с последними антивирусными базами, встроенный монитор активности должен сам исправно сигнализировать об опасности заражения. Но вероятность заражения вирусами, к сожалению, есть всегда. Если ваш компьютер начал работать с непонятными «глюками», постоянно подтормаживает, сам перезагружается, и т.п. - запустите сканер проверки компьютера на вирусы. В идеале такую проверку нужно делать раз в месяц вне зависимости проявляются ли какие-то «симптомы» вирусного заражения, или нет.

Очень часто обычное полное сканирование вполне успешно справляется с задачами обнаружения и лечения вирусов. В некоторых случаях антивирус может попросить перезагрузку.

Но иногда антивирус не может справиться с каким-то новым вирусом в обычном режиме. Проведите лечение вирусов в безопасном режиме. А в идеале, если есть возможность, присоедините винчестер с зараженного компьютера на другой ПК и проверьте его там. Или загрузитесь вне системы, допустим, с загрузочного диска. В этом случае не происходит автоматической загрузки вредоносных программ, которые запускаются вместе с системными службами при запуске операционной системы на зараженном ПК.

При сканировании ПК вне системы очень хорошо зарекомендовала себя бесплатная утилита Dr.Web CureIt! Подобные утилиты есть также и у McAfee, и у Касперского.

Иногда, в особо сложных случаях может помочь только ручное удаление вируса. Для этого произведите поиск по названию вируса в интернете. На каждый вирус достаточно информации по его лечению.

Не открывайте неизвестные интригующие ссылки, проверяйте незнакомые носители информации, не заходите сайты, о которых есть информация, что они заражают компьютеры, т.п. Эти простые действия значительно облегчат жизнь вашему антивирусу.

# 29. Основные задачи сетевого администрирования.

Администрирование – процедуры управления, регламентирующие некоторые процессы или их часть. Как правило, оно фиксирует и руководит процессами и ситуациями, нуждающимися в ограничениях или целевом управлении.

1. Планирование сети.

Несмотря на то, что планированием и монтажом больших сетей обычно занимаются специализированные компании-интеграторы, сетевому администратору часто приходится планировать определенные изменения в структуре сети — добавление новых рабочих мест, добавление или удаление сетевых протоколов, добавление или удаление сетевых служб, установка серверов, разбиение сети на сегменты и т.д. Данные работы должны быть тщательно спланированы, чтобы новые устройства, узлы или протоколы включались в сеть или исключались из нее без нарушения целостности сети, без снижения производительности, без нарушения инфраструктуры сетевых протоколов, служб и приложений.

1. Установка и настройка сетевых узлов (устройств активного сетевого оборудования, персональных компьютеров, серверов, средств коммуникаций).

Данные работы могут включать в себя — замену сетевого адаптера в ПК с соответствующими настройками компьютера, перенос сетевого узла (ПК, сервера, активного оборудования) в другую подсеть с соответствующим изменениями сетевых параметров узла, добавление или замена сетевого принтера с соответствующей настройкой рабочих мест.

1. Установка и настройка сетевых протоколов.

Данная задача включает в себя выполнение таких работ — планирование и настройка базовых сетевых протоколов корпоративной сети, тестирование работы сетевых протоколов, определение оптимальных конфигураций протоколов.

1. Установка и настройка сетевых служб.

Корпоративная сеть может содержать большой набор сетевых служб. Кратко перечислим основные задачи администрирования сетевых служб:

* + установка и настройка служб сетевой инфраструктуры (службы DNS, DHCP, WINS, службы маршрутизации, удаленного доступа и виртуальных частных сетей);
  + установка и настройка служб файлов и печати, которые в настоящее время составляют значительную часть всех сетевых служб;
  + администрирование служб каталогов (Novell *NDS*, Microsoft Active Directory), составляющих основу корпоративной системы безопасности и управления доступом к сетевым ресурсам;
  + администрирование служб обмена сообщениями (системы электронной почты);
  + администрирование служб доступа к базам данных.

1. Поиск неисправностей.

Сетевой администратор должен уметь обнаруживать широкий спектр неисправностей — от неисправного сетевого адаптера на рабочей станции пользователя до сбоев отдельных портов коммутаторов и маршрутизаторов, а также неправильные настройки сетевых протоколов и служб.

1. Поиск узких мест сети и повышения эффективности работы сети.

В задачу сетевого администрирования входит анализ работы сети и определение наиболее узких мест, требующих либо замены сетевого оборудования, либо модернизации рабочих мест, либо изменения конфигурации отдельных сегментов сети.

1. Мониторинг сетевых узлов.

Мониторинг сетевых узлов включает в себя наблюдение за функционированием сетевых узлов и корректностью выполнения возложенных на данные узлы функций.

1. Мониторинг сетевого трафика.

Мониторинг сетевого трафика позволяет обнаружить и ликвидировать различные виды проблем: высокую загруженность отдельных сетевых сегментов, чрезмерную загруженность отдельных сетевых устройств, сбои в работе сетевых адаптеров или портов сетевых устройств, нежелательную активность или атаки злоумышленников (распространение вирусов, атаки хакеров и др.).

1. Обеспечение защиты данных.

Защита данных включает в себя большой набор различных задач: резервное копирование и восстановление данных, разработка и осуществление политик безопасности учетных записей пользователей и сетевых служб (требования к сложности паролей, частота смены паролей), построение защищенных коммуникаций (применение протокола IPSec, построение виртуальных частных сетей, защита беспроводных сетей), планирование, внедрение и обслуживание инфраструктуры открытых ключей (PKI).

# 30. Операционная система MS Windows NT/2000/XP/2003.

Windows NT

Windows NT — линейка операционных систем (ОС) производства корпорации Microsoft и название первых версий ОС.

Windows NT была разработана после прекращения сотрудничества Microsoft и IBM над OS/2, развивалась отдельно от других ОС семейства Windows (Windows 3.x и Windows 9x) и, в отличие от них, позиционировалась как надёжное решение для рабочих станций (Windows NT Workstation) и серверов (Windows NT Server). Windows NT дала начало семейству операционных систем, в которое входят: собственно Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012.

Архитектурные модули Windows NT

Архитектура Windows NT имеет модульную структуру и состоит из двух основных уровней — компоненты, работающие в режиме пользователя и компоненты режима ядра. Программы и подсистемы, работающие в режиме пользователя имеют ограничения на доступ к системным ресурсам. Режим ядра имеет неограниченный доступ к системной памяти и внешним устройствам. Ядро системы NT называют гибридным ядром или макроядром. Архитектура включает в себя само ядро, уровень аппаратных абстракций (HAL), драйверы и ряд служб (Executives), которые работают в режиме ядра (Kernel-mode drivers) или в пользовательском режиме (User-mode drivers).

Пользовательский режим Windows NT состоит из подсистем, передающих запросы ввода/вывода соответствующему драйверу режима ядра посредством менеджера ввода/вывода. Есть две подсистемы на уровне пользователя: подсистема окружения (запускает приложения, написанные для разных операционных систем) и интегрированная подсистема (управляет особыми системными функциями от имени подсистемы окружения). Режим ядра имеет полный доступ к аппаратной части и системным ресурсам компьютера. И также предотвращает доступ к критическим зонам системы со стороны пользовательских служб и приложений.

Отличия Windows 2000 от NT 4.0

Некоторыми из наиболее существенных улучшений в Windows 2000 по сравнению с Windows NT 4.0 являются:

Поддержка службы каталогов Active Directory. Серверная часть Active Directory поставляется с изданиями Server, Advanced Server и Datacenter Server, в то время как полную поддержку службы на стороне клиента осуществляет издание Professional.

Службы IIS версии 5.0. По сравнению с IIS 4.0 эта версия включает, помимо прочего, версию 3.0 системы веб-программирования ASP.

Файловая система NTFS версии 3.0 (также называемая NTFS 5.0 по внутренней версии Windows 2000 — NT 5.0). В этой версии NTFS впервые появилась поддержка квот, то есть ограничений на максимальный объём хранимых файлов для каждого пользователя.

Обновлённый пользовательский интерфейс, включающий Active Desktop на основе Internet Explorer версии 5 и подобный, таким образом, интерфейсу Windows 98. Цветовая схема оформления была переработана.

Языковая интеграция: предыдущие версии Windows выпускались в трёх вариантах — для европейских языков (однобайтные символы, письмо только слева направо), для дальневосточных языков (многобайтные символы) и для ближневосточных языков (письмо справа налево с контекстными вариантами букв). Windows 2000 объединяет эти возможности; все её локализованные версии сделаны на единой основе.

Шифрованная файловая система EFS, с помощью которой можно зашифровать файлы и папки.

Основные особенности Windows XP

Операционная система Microsoft Windows XP, создана на основе технологии NT и является прямой наследницей системы Windows 2000. Вместе с тем, все лучшие нововведения, включенные в Windows Me, можно обнаружить и в Windows XP. При сохранении высоких показателей надежности, безопасности и быстродействия, система стала более простой в освоении, в ней появилось множество средств, предназначенных для индивидуальных домашних пользователей.

Система поставляется в нескольких вариантах, ориентированных на разные особенности применения. Версия Microsoft Windows XP Home Edition предназначена для индивидуальных пользователей, чаще всего работающих на домашнем компьютере. В этой версии особый упор сделан на работу с рисунками, аудио и видео. Версия Microsoft Windows XP Professional предназначена, как ясно из названия, для профессионалов. Эта версия наиболее часто используется в организациях. Если же вы дома выполняете сложные работы по созданию и редактированию изображений, моделированию и конструированию, или любые другие сложные работы, то эта версия подойдет и для вашего домашнего компьютера. Версия Microsoft Windows XP Server предназначена для установки на сервере - мощном компьютере, обеспечивающем работу множества пользователей в компьютерной сети. Работа в локальных сетях выходит за рамки данной книги, так что версия для сервера рассматриваться нами не будет. В книге описана основная версия операционной системы - Windows XP Professional. Практически не отличается от нее версия Windows XP Home Edition. Об имеющихся незначительных отличиях будет особо указано.

Следует отметить, что для эффективной работы с операционной системой Windows XP требуется достаточно мощный современный компьютер. Прежде всего, в компьютере должно быть установлено не менее 128 мегабайт памяти. Лучше установить 256 мегабайт, чтобы система работала быстрее. Процессор может использоваться любой, но не слишком старый. Если тактовая частота процессора не менее 300 мегагерц, то он подойдет. Хотя лучше, конечно, использовать процессор с частотой более одного гигагерца. Жесткий диск должен вмещать не только файлы операционной системы и временные файлы, но и иметь достаточно свободного места, например, для создания образа компакт-диска перед его записью. Реально требуется диск размером не менее двух-трех гигабайт. А если учесть, что на диске надо установить другие программы и оставить место для разных документов, то диск объемом 10 гигабайт нельзя назвать слишком большим.

В версии Windows XP сильно изменился внешний вид системы. Кнопки, значки, панели теперь выглядят немного иначе. Даже главное меню Windows изменилось. Изменения интерфейса самые значительные после перехода с Windows 3.1 на Windows 95. Вместе с тем имеется возможность использовать и старый интерфейс, если вы к нему привыкли. Особо следует отметить работу программ в режиме совместимости с предыдущими версиями Windows. Вы можете работать с программой, которая написана для Windows 95, и не работает в Windows 2000. Все версии Windows XP имеют множество нововведений. Поддерживается значительно больше разнообразных устройств. Система позволяет легко и удобно обрабатывать видеофильмы, фотографии, рисунки, музыку и песни. Теперь с помощью Windows XP любой сможет построить домашнюю сеть на основе двух-трех компьютеров, обеспечив совместное использование файлов, папок, принтера, факса и доступа в Интернет.

При работе с Windows XP для записи информации на записываемые и перезаписываемые компакт-диски вам не потребуется устанавливать дополнительные программы. Записать компакт-диск можно прямо из проводника Windows. Кстати, проводник сильно изменился. Кроме поддержки сжатых папок, особых папок для хранения рисунков, музыки и видео, добавлена панель с командами, состав которых меняется в зависимости от выполняемых вами действий.

Полезным компонентом системы является редактор видеофильмов. Теперь вы сможете выполнить профессиональный нелинейный монтаж своих любительских фильмов. Существенно легче стало работать с цифровыми фотокамерами и сканерами. Вам не потребуется никакой дополнительной программы, чтобы ввести фотографию в компьютер, незначительно преобразовать ее и распечатать на принтере. Универсальный проигрыватель аудио и видео теперь поддерживает больше форматов и позволяет сменить свой внешний вид. Вы можете сами создавать звуковые файлы в популярном формате МРЗ. Универсальный проигрыватель поддерживает также воспроизведение цифровых видеодисков (DVD-дисков), что позволит вам насладиться высочайшим качеством изображения и звука при просмотре современных фильмов. Для развлечений в Windows XP включено несколько новых игр, часть из которых позволяют вам играть в Интернете.

Существенно улучшена и защита системы. Теперь при случайном удалении важных системных файлов, они будут автоматически восстановлены. Имеется возможность восстановления системы к более раннему состоянию, после установки новых программ и оборудования. Улучшенная поддержка технологии Plug & Play позволяет подключить к компьютеру многие современные бытовые приборы.

Дальнейшее развитие получили и средства для работы с Интернетом. Существенно переработана справочная система, улучшена система безопасности. Многочисленные изменения коснулись средств администрирования и управления работой множества пользователей в локальной вычислительной сети.

В системе имеется и множество других нововведений, о которых вы узнаете по мере прочтения книги и знакомства с Windows XP. Однако перед началом работы с системой мы рекомендуем вам познакомиться с основными понятиями, используемыми в Windows XP. Если вы знакомы с предыдущими версиями Windows, большинство понятий для вас окажутся знакомыми.

Windows Server 2003

Windows Server 2003 (кодовое название при разработке — Whistler Server, внутренняя версия — Windows NT 5.2) — операционная система семейства Windows NT от компании Microsoft, предназначенная для работы на серверах. Она была выпущена 24 апреля 2003 года.

Windows Server 2003 является новой версией Windows 2000 Server и серверным вариантом операционной системы Windows XP. Изначально Microsoft планировала назвать этот продукт «Windows .NET Server» с целью продвижения своей новой платформы Microsoft .NET. Однако впоследствии это название было отброшено, чтобы не вызвать неправильного представления о .NET на рынке программного обеспечения.

# 31. Сетевые протоколы NetBIOS, TCP/IP.

NetBIOS (Network Basic Input/Output System, Сетевая Базовая Система Ввода/Вывода) - это интерфейс для работы в локальных сетях, разработанный фирмой Sytek для компании IBM в 1983 году. Как гласит RFC1001: "NetBIOS defines a software interface, not a protocol. There is no "official" NetBIOS service standard". И всё же NetBIOS, по моему скромному мнению, и интерфейс и протокол, поэтому, с вашего позволения, назовем его стандартом (не смотря на то, что официально он так и не был полностью стандартизован). Довольно часто NetBIOS называют сетевым протоколом, но это не совсем корректно, поскольку NetBIOS реализован в Windows в виде отдельного функционала в разных частях ОС - в виде интерфейса в библиотеках пользовательского режима и режима ядра, и в виде модуля в стеке сетевого протокола. Интерфейс NetBIOS представляет из себя стандартный набор для разработки приложений (API), протокол NetBIOS функционирует на транспортном/сеансовом уровне стека и используется для передачи данных, управления сеансом и прочих нужд.

Для чего же в своё время потребовался NetBIOS? Для осуществления возможности взаимодействия станций в небольших сетях. Что включает в себя возможность взаимодействия по сети? Это назначение станции сетевого имени, по которому она будет доступна в сети, это возможность найти станцию в сети по её имени, возможность соединиться с ресурсами станции и начать с обмениваться с ними данными. Это и возможность получить список сетевых станций, которые подключены к сегменту сети и многое другое, что может быть определено термином "сетевое взаимодействие".

Особенностью NetBIOS является возможность работы "поверх" различных протоколов, таких как IPX, NetBEUI и TCP/IP. В своё время реализация протокола NetBIOS в Windows была существенно переработана и ориентирована на использование протокола TCP/IP (как наиболее перспективного), получив новое название “NetBIOS over TCP/IP” (NetBIOS через TCP/IP). NetBIOS через TCP/IP имеет псевдоним NetBT (NBT). NetBIOS через TCP/IP представляет собой промежуточный уровень между NetBIOS и TCP/IP и создан для того, чтобы приложения на базе NetBIOS могли работать в сетях TCP/IP, то есть предназначен для отображения имен NetBIOS в IP-адреса и, наоборот. Как мы уже упоминали, NetBIOS разрабатывался на заре становления сетевых технологий, и с того времени часто модифицировался, однако параллельно с ним создавались и другие стандарты сетевого взаимодействия, которые существенно опережали NetBIOS по функционалу. На данный момент NetBIOS считается устаревшим стандартом и не рекомендуется к использованию, заместо него, в части организации передачи данных по сети, Microsoft предлагает использовать сокеты (windows sockets), почтовые ящики (mailslots), именованные каналы (named pipes). Для оставшегося функционала, то есть для сервисных целей разработан новый протокол LLMNR.

NetBIOS через TCPIP не поддерживает IPv6. Поэтому для целей организации сети в локальном сегменте без участия опорных серверов (DNS/WINS) с использованием IPv4/IPv6 разработан протокол LLMNR.

Однако, не смотря на устаревание NetBIOS, поддержка его сохранена и по сей день, код NetBIOS через TCP/IP всё еще присутствует в составе последних версий Windows. Причиной столь огромной популярности стандарта является тот факт, что до определенного времени NetBIOS оставался основным интерфейсом программирования сетевых приложений, и с использованием функционала NetBIOS было написано огромное количество разнообразного программного обеспечения.

NetBT является неотъемлемой частью сетевого стека TCP/IP ОС Windows и инсталлируется вместе с протоколом TCP/IP. Части функционала NetBT встречаются в коде библиотек, переменных окружения (%COMPUTERNAME%, %USERDOMAIN%), коде некоторых современных антивирусных продуктов, почтовых серверов, баз данных. NetBIOS до сих пор используется в алгоритме добавления рабочей станции в домен, в процедурах работы с сетевым окружением, подключения сетевых дисков. Об исключительном значении протокола NetBIOS через TCP/IP говорит уже и тот факт, что штатными средствами самой ОС протокол NetBIOS может быть только отключен, но никак не удален. Удаление же его возможно только вместе с удалением протокола TCP/IP.

До определенного времени считалось актуальным высказывание “Cеть Windows не живет без NetBIOS”, но начиная с версии Windows 2000, стало возможным избавиться от использования соединений на основе NetBIOS через TCP/IP и перейти исключительно на соединения по протоколу SMB (Server Message Block). Подобные соединения еще называются TCP/IP Direct Hosting (или DirectSMB). В соединениях на основе SMB отсутствует начальная стадия установки сессии TCP под названием “NetBIOS session setup” и используется единственный порт TCP/445.

SMB - это простой протокол удаленной работы с ресурсами (дисками, устройствами) компьютера. Удивительно, но SMB долгое время довольно тесно взаимодействовал с NetBIOS в части именования и разрешения имен, а так же требовал наличия установленной сессии NetBIOS для взаимодействия между узлами (работал поверх NetBIOS). Теперь же SMB представляет собой самостоятельный протокол и получил дальнейшее развитие в виде стандарта под названием CIFS, полностью независимого от NetBIOS.

NetBIOS

NetBIOS (Network Basic Input/Output System) — протокол для работы в локальных сетях на персональных ЭВМ типа IBM/PC, разработан в виде интерфейса, который не зависит от фирмы-производителя. Был разработан фирмой Sytek Corporation по заказу IBM в 1983 году. Он включает в себя интерфейс сеансового уровня (англ. NetBIOS interface), в качестве транспортных протоколов использует TCP и UDP.

Особенностью NetBIOS является возможность его работы поверх разных протоколов, самыми распространёнными/известными из которых являются NetBEUI, IPX и стек протоколов TCP/IP; причём если старые версии Windows ориентировались на более лёгкие в реализации и менее ресурсоёмкие NetBEUI и IPX, то современные Windows ориентируются на TCP/IP. При использовании NetBEUI и IPX NetBIOS сам обеспечивает надёжность доставки данных (функциональность SPX не использовалась), а при использовании TCP/IP надёжность доставки обеспечивает TCP, за что удостоился отдельного имени «NBT».

Интерфейс NetBIOS представляет собой типичный интерфейс взаимодействия приложений (API) для обеспечения сетевых операций ввода-вывода и управления низлежащим транспортным протоколом. Приложения, использующие NetBIOS API интерфейс, могут работать только при наличии протокола, допускающего использование такого интерфейса.

TCP/IP

TCP/IP - это аббревиатура термина Transmission Control Protocol/Internet Protocol (Протокол управления передачей/Протокол Internet). В терминологии вычислительных сетей протокол - это заранее согласованный стандарт, который позволяет двум компьютерам обмениваться данными. Фактически TCP/IP не один протокол, а несколько. Именно поэтому вы часто слышите, как его называют набором, или комплектом протоколов, среди которых TCP и IP - два основных.

Программное обеспечение для TCP/IP, на вашем компьютере, представляет собой специфичную для данной платформы реализацию TCP, IP и других членов семейства TCP/IP. Обычно в нем также имеются такие высокоуровневые прикладные программы, как FTP (File Transfer Protocol, Протокол передачи файлов), которые дают возможность через командную строку управлять обменом файлами по Сети.

TCP/IP - зародился в результате исследований, профинансированных Управлением перспективных научно-исследовательских разработок (Advanced Research Project Agency, ARPA) правительства США в 1970-х годах. Этот протокол был разработан с тем, чтобы вычислительные сети исследовательских центров во всем мире могли быть объединены в форме виртуальной "сети сетей" (internetwork). Первоначальная Internet была создана в результате преобразования существующего конгломерата вычислительных сетей, носивших название ARPAnet, с помощью TCP/IP.

О TCP/IP можно было бы рассказать много больше, но есть три ключевых момента:

* TCP/IP - это набор протоколов, которые позволяют физическим сетям объединяться вместе для образования Internet. TCP/IP соединяет индивидуальные сети для образования виртуальной вычислительной сети, в которой отдельные главные компьютеры идентифицируются не физическими адресами сетей, а IP-адресами.
* В TCP/IP используется многоуровневая архитектура, которая четко описывает, за что отвечает каждый протокол. TCP и UDP обеспечивают высокоуровневые служебные функции передачи данных для сетевых программ, и оба опираются на IP при передаче пакетов данных. IP отвечает за маршрутизацию пакетов до их пункта назначения.
* Данные, перемещающиеся между двумя прикладными программами, работающими на главных компьютерах Internet, "путешествуют" вверх и вниз по стекам TCP/IP на этих компьютерах. Информация, добавленная модулями TCP/IP на стороне отправителя, "разрезается" соответствующими TCP/IP-модулями на принимающем конце и используется для воссоздания исходных данных.

Протоколы транспортного уровня TCP и UDP

Протоколы транспортного уровня в стеке TCP/IP представлены двумя протоколами TCP и UDP. Протокол TCP позволяет устанавливать виртуальный канал передачи данных между компьютерами. После установления канала, программа может направлять в него данные непрерывным потоком, как на стандартное устройство ввода-вывода.

Протокол TCP сам разобьет данные на пакеты, при помощи алгоритма «скользящего окна», обеспечит подтверждение факта получения пакетов принимающей стороной и повторную передачу пакетов, если в этом будет необходимость. Кроме того, в протоколе TCP реализованы достаточно сложные механизмы регулирования загрузки сети и устранения заторов в сети.

Протокол UDP более быстр, чем протокол TCP, однако ненадежен.

Данные передаются без установления виртуального канала, в предположении, что принимающая сторона ждет данные. Программа должна сама позаботиться о разбитии передаваемых данных на пакеты, протокол не содержит средств подтверждения файла доставки сообщения и средств коррекции ошибок – все эти задачи должна решать программа.

При рассмотрении протоколов транспортного уровня необходимо остановиться на понятиях «порт» и «сокет».

Порт в протоколах транспортного уровня – это не физически существующий порт ввода-вывода, а «виртуальный» порт, который программно изолирует данные, передаваемые по одному порту, от данных, передаваемых по другому порту.

*Порты нумеруются от 0 до 65535.*

*Сокет* (socket) – это описатель сетевого соединения между двумя сетевыми приложениями, которое включает в себя:

- IP-адрес и номер порта локальной машины

- IP-адрес и номер порта удаленной машины.

Сокет однозначно описывает сетевое соединение. У двух различных соединений, хотя бы один из приведенных выше параметров должен отличаться.

**Примеры:**

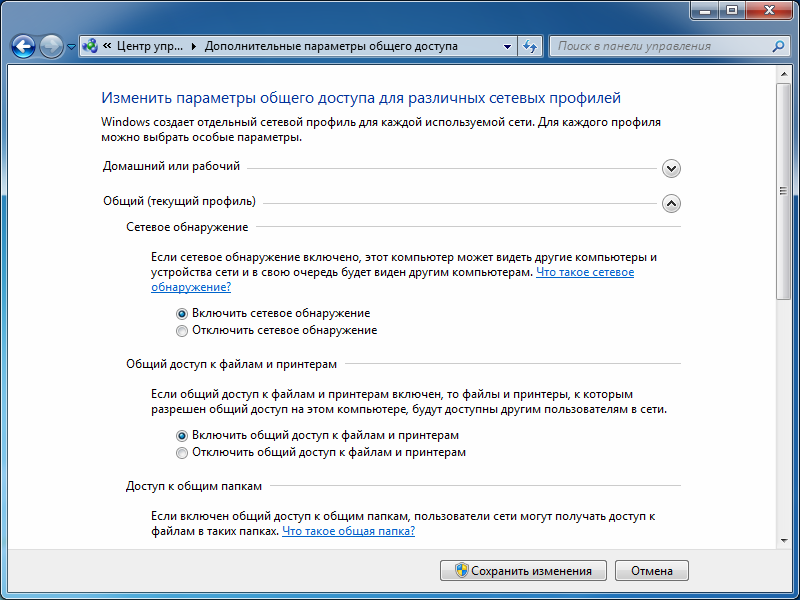
UDP: все, что вам неинтересно, если вы всегда получаете все данные

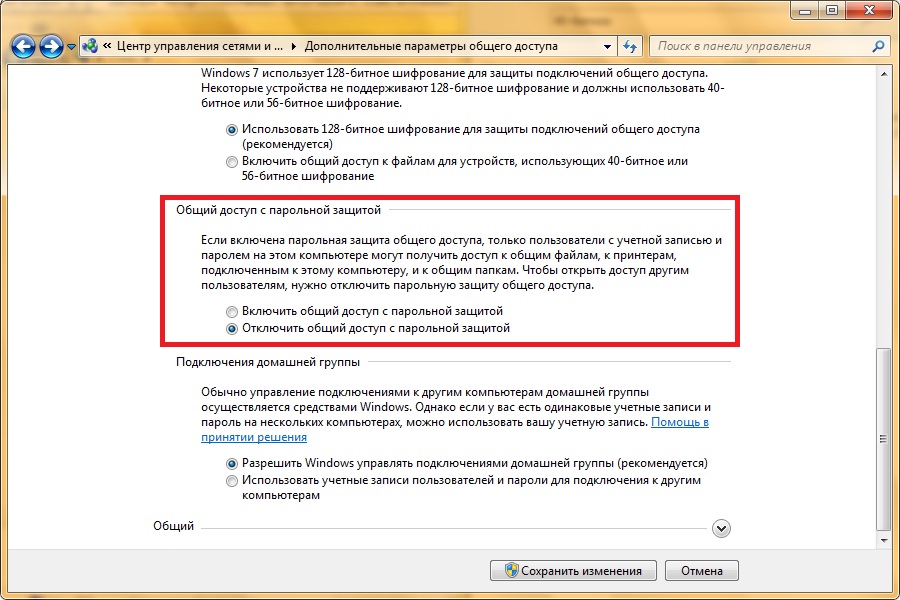
* Туннелирование / VPN (потерянные пакеты в порядке - туннельный протокол позаботится об этом)
* Потоки мультимедиа (потерянные кадры не отражаются картинке)
* Игры
* Локальные механизмы вещания (одно приложение, работающее на разных машинах, «обнаруживающее» друг друга)

TCP: почти все, где вам нужно получить все переданные данные

* Web
* SSH, FTP, telnet
* SMTP, отправка почты
* IMAP / POP, получая почту

# 32. Совместный доступ к файлам и принтерам в Windows.



 Даже в самой простой сети можно реализовать множество полезных функций. Однако одно из самых больших преимуществ домашних и корпоративных компьютерных сетей — это возможность обмениваться данными.

В Windows 7 традиционный общий доступ к файлам — это двухэтапный процесс. Сначала вы открываете папку на своем компьютере для общего доступа, а потом другой пользователь считывает или модифицирует файлы в общей папке. Для защиты общих данных от любопытных глаз в Windows применяется система учетных записей, о которой мы говорим в этой главе. Кроме того, система разрешений позволяет указывать, какие действия могут выполнять с вашими файлами другие пользователи. Все очень четко и красиво.

Функция общего доступа нужна не только для пересылки файлов с одного компьютера на другой. Несколько пользователей, работающих над общим проектом, могут редактировать одни и те же файлы, не путаясь в множестве разнообразных версий. (Разумеется, есть определенные пределы. Например, если один пользователь откроет файл Word, то другой не сможет отредактировать этот документ. В то же время базы данных, например Microsoft Access, в определенных условиях поддерживают одновременный доступ к файлу нескольких пользователей.)

Понятно, что компьютер, на котором хранится общая папка, должен быть включен, иначе пользователи не смогут обратиться к ее содержимому, однако владельцу файлов не обязательно входить в систему.

Есть одна сложность: по умолчанию Windows 7 позволяет всем пользователям сети читать файлы, но изменять их не разрешает никому. Однако злоумышленник может разведать уязвимости, щелкнуть на паре кнопок и получить полный доступ.

Настраивая общий доступ к папке, вы, но сути, открываете пользователям сети черный ход на свой компьютер, потенциально подставляя под угрозу коифиден-циальные данные. Очень важно постоянно помнить о безопасности, особенно когда компьютер подключен к Интернету. В противном случае вы рискуете раскрыть злоумышленникам свои персональные данные. Помимо этого, незащищенная система больше подвержена заражению вирусами, троянскими программами и прочими злонамеренными приложениями. Я не хочу сказать, что пользоваться функцией общего доступа к файлам вообще не стоит. Просто, как и к любой другой задаче, к этой возможности нужно подходить с умом.

# 33. Команды управления сетевыми службами в Windows.

Управление системными службами через командную строку

Службы обеспечивают ключевые функции рабочих станций и серверов. Для управления системными службами на локальных и удаленных системах используется команда контроллера служб (service controller command) SC, имеющая набор подкоманд, здесь описывается лишь их некоторая часть.

Для удобства пользования в командной строке используются сокращенные названия служб Windows (например, Spooler - диспетчер очереди печати, Dhcp - DHCP-клиент, W32Time - Служба времени Windows и т.п.).

Команды SC - настройка, запуск и останов системный служб Windows

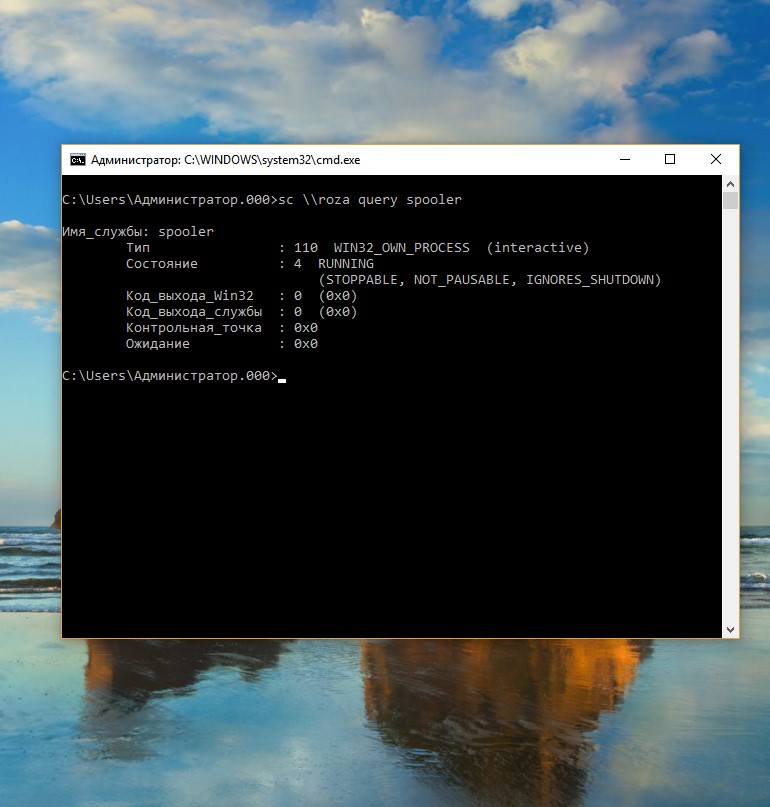
* SC config — настройка учетных записей регистрации и запуска служб;
* SC query — вывод списка всех служб, настроенных на компьютере;
* SC qc отображение конфигурации определенной службы;
* SC start — запуск служб;
* SC stop — останов служб;
* SC pause — приостановка работы служб;
* SC continue — возобновление работы служб;
* SC failure — задание действий, выполняемых при сбое службы;
* SC qfailure — просмотр действий, выполняемых при сбое службы.

Примеры работы с командами SC

Во всех командах можно указывать имя удаленного компьютера, со службами которого вы хотите работать. Для этого вставьте UNC-имя (например, \\Roza) или IP-адрес (например, 192.168.1.1) компьютера перед используемой подкомандой. Вот синтаксис: **sc ИмяСервера Подкоманда.**

Пример команды sc на компьютере с Windows 10:

* **sc \\Roza query Spooler**



Чтобы получить список всех служб, настроенных в системе, введите в командной строке команду: **sc query type=service state=all** или **sc ИмяСервера query type=service state=all** где ИмяСервера - UNC-имя или IP-адрес удаленного компьютера.

# 34. Набор команд управления сетевыми службами Net.

NET – управление сетевой конфигурацией в командной строке Windows. Для получения полноценного доступа к функциям управления настройками сети утилиты NET.EXE требуются права администратора системы. Синтаксис и возможности команды зависят от версии Windows. Команда NET SEND, использовавшаяся для обмена сообщениями между компьютерами локальной сети в Windows 2000/XP, в операционных системах Windows Vista и более поздних не поддерживается.

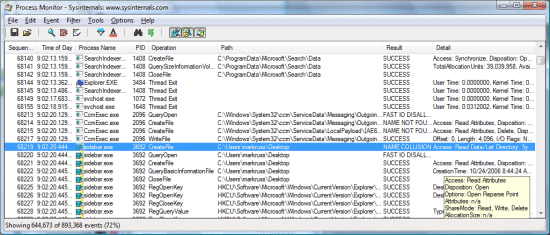
Формат командной строки:

**NET  
[ ACCOUNTS | COMPUTER | CONFIG | CONTINUE | FILE | GROUP | HELP |  
HELPMSG | LOCALGROUP | PAUSE | SESSION | SHARE | START |  
STATISTICS | STOP | TIME | USE | USER | VIEW ]**  
При использовании команды без параметров или с параметром **help** отображается краткая подсказка по использованию.

Для получения краткой подсказки по подкоманде, можно использовать синтаксис:  
**net share /?** - отобразить подсказку по использованию команды **NET SHARE**. В ответ отображается краткая справка

# 35. Мониторинг работы системы.

Мониторинг процессов системы крайне важен в самых разных ситуациях. Начиная от выявления вирусной активности и заканчивая простым слежением за системой для последующего высвобождения системных ресурсов. Самый простой способ мониторинга процессов – Диспетчер задач Windows. Но этот инструмент весьма ограничен в функциональных возможностях и не может дать той обширной информации по всем процессам системы, которую может предоставить программа Process Monitor.



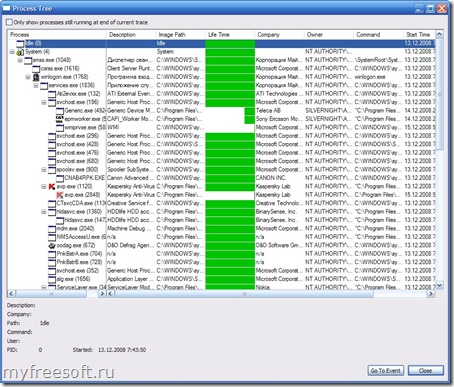
Process Monitor позволяет в полной мере получить всю необходимую информацию по использованию компьютера различными процессами. С помощью этой утилиты можно узнать все подробности о каждом процессе, о том, какие ветки системного реестра он использует, с какими файлами работает и многое другое. Среди основных функциональных возможностей Process Monitor есть:

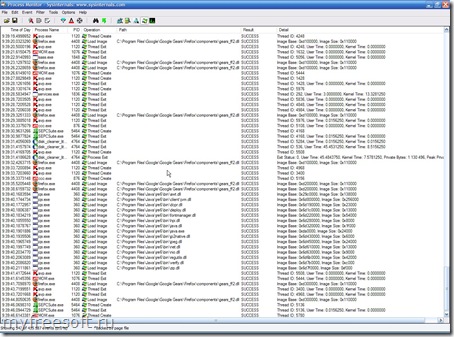
\* Мониторинг всех процессов с момента запуска и до их завершения. Это позволяет в полной мере оценить действия того или иного процесса.

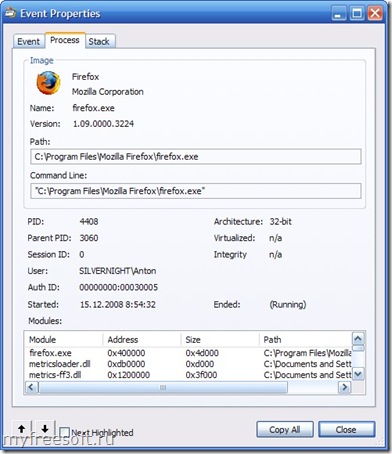
\* Система фильтрации позволяет задать в настройках параметры, позволяющие следить только за нужными приложениями. Это необходимо для того, чтобы не потеряться в огромном потоке информации обо всех процессах, запущенных на компьютере.

\* С помощью специального инструмента «Дерево процессов» (“Process Tree”) можно отслеживать иерархию запуска того или иного процесса.

\* Узнать самые мельчайшие подробности о том или ином процессе можно с помощью детальной информации о нем.

[](https://myfreesoft.ru/wp-content/uploads/2008/12/windowslivewriterf522b4992132-b4e5screenshot-01-2.jpg)

[](https://myfreesoft.ru/wp-content/uploads/2008/12/windowslivewriterf522b4992132-b4e5screenshot-02-2.jpg)

[](https://myfreesoft.ru/wp-content/uploads/2008/12/windowslivewriterf522b4992132-b4e5screenshot-03-2.jpg)

# 36. Веб-интерфейс для организации удаленного управления компьютером под управлением Windows 2000/2003.

Администраторы сетей на базе Windows давно уже сетуют на сложности удаленного управления системами в этой среде. Полного набора средств для выполнения такого рода задач нет ни в самой Windows NT, ни в прилагаемых к ней комплектах ресурсов, а то, что есть, не вполне удовлетворяет специалистов как с точки зрения функциональности, так и по параметрам безопасности (типичный пример тому - служба Telnet системы NT). И дело, пожалуй, не столько в инструментарии, сколько в том, что средства дистанционного управления не предусмотрены ни в базовой поставке ОС, ни в ее службах или компонентах. Можно, конечно, воспользоваться графическими пакетами дистанционного управления, скажем pcAnywhere32 компании Symantec, но это связано с дополнительными расходами в виде лицензионных платежей, сумма которых зависит от числа серверов в сети. И потом, возможности программ такого рода ограничены хотя бы потому, что значительная часть полосы пропускания уходит на обслуживание запросов собственно графического интерфейса Windows.

Если не считать нареканий на нестабильность и на то, что систему приходится часто перезапускать, претензии к NT связаны в основном с отсутствием программ с интерфейсом командной строки и встроенных средств дистанционного администрирования. Как-то на конференции я слышал, как Марк Минаси, внештатный редактор журнала Windows 2000 Magazine, прокомментировал ситуацию так: «Что требуется для управления удаленным сервером UNIX? Протокол Telnet. А если это сервер NT? Тут уже не обойтись без авиабилета до места эксплуатации сервера».

Конечно, такое сравнение систем на базе Windows и ОС типа UNIX, предполагающих использование интерфейса командной строки, может вызвать улыбку, однако оно обнажает весьма серьезный вопрос - необходимость оснащения серверов Windows современными средствами дистанционного администрирования. Впрочем, к чести сотрудников корпорации Microsoft надо сказать, что в ответ на жалобы пользователей NT они всерьез взялись за эту проблему при разработке Windows 2000.

Служба терминалов

Предложенное Microsoft решение складывается из двух составляющих. Во-первых, в ОС Windows 2000 были реализованы усовершенствованные утилиты командной строки, и это сократило число задач, требующих использования графических средств. В рамках этого подхода разработчики создали новые утилиты командной строки, такие, как NetShell (Neth), и усовершенствовали утилиты командной строки системы NT. Во-вторых, специалисты Microsoft реализовали удаленный доступ к серверам через графический интерфейс, расширив возможности технологии, реализованной в семействе продуктов Windows 2000 Server (речь идет о службе терминалов Windows 2000 Server Terminal Services).

В системе NT терминальные службы для тонких клиентов реализованы в версии NT Server 4.0 - Terminal Server Edition (WTS). Эта версия NT Server 4.0 построена на базе обеспечивающего взаимодействие с несколькими пользователями ядра, которое значительно отличается от других версий NT. WTS можно оснастить дополнительными службами и средствами, которые имеются в Citrix MetaFrame. Эти средства предоставляют удаленным клиентам и терминалам на базе Windows (Windows-based terminals, или WBTs) доступ к NT и прикладным программам на базе Windows, но в то же время они могут обеспечить доступ и администраторам, позволяя им управлять сервером терминалов.

В ядре операционной системы Win-dows 2000 разработчики Microsoft реализовали средства терминальной службы для всех версий Windows 2000 Server. Кроме того, здесь предусмотрен особый режим функционирования системы, предназначенный для обеспечения дистанционного администрирования любого сервера Windows 2000. Чтобы иметь возможность работать в этом режиме, не требуются особые лицензии, как в случае с обычным режимом работы службы терминалов - режимом сервера приложений. Достаточно просто активизировать службу Termi-nal Services в режиме административного управления на любом сервере Windows 2000; дополнительные лицензии для клиентов при этом не нужны. Хотя служба Terminal Services обеспечивает взаимодействие только с клиентами Windows, в систему Windows 2000 включены также 16- и 32-разрядные версии клиентов службы терминалов RDP 5.0. Таким образом, серверы Windows наконец-то получили полнофункциональные средства дистанционного администрирования и управления. Иначе говоря, теперь для удаленного администрирования серверов Windows уже нет нужды покупать ни программные пакеты дистанционного управления, ни авиабилет до места эксплуатации сервера. О том, как управлять машинами, которые работают в средах, отличных от Windows 2000 Server, рассказано ниже.

Дистанционное управление другими компонентами сети

Спору нет, включение службы Windows 2000 Server Terminal Services в семейство продуктов Windows 2000 Server - это поистине бесценный подарок для администраторов. Но как решать проблему дистанционного управления машинами, функционирующими в других средах, - такими, как системы Windows 2000 Professional, рабочие станции и серверы Windows NT 4.0, а также системы, работающие под управлением ОС, отличных от Windows? Для удаленного управления этими системами можно использовать стандартные пакеты дистанционного управления: pcAnywhere фирмы Symantec, Carbon Copy корпорации Compaq, Timbbuktu фирмы Netopia и ControlIT компании Computer Associates (CA). В этих продуктах реализованы мощные средства дистанционного управления, передачи файлов и печати.

Есть, впрочем, еще одно решение, примечательное не только своими функциональными возможностями, но и тем, что пользоваться им можно совершенно бесплатно. Компания AT&T Research (вернее, ее подразделение AT&T Labs Cambridge) разработала продукт под названием Virtual Network Computing (VNC), обеспечивающий дистанционное управление клиентами Windows, UNIX и Macintosh. Правда, устанавливать эту программу, да и работать с ней, сложнее, чем со службой Terminal Services и со многими коммерческими пакетами дистанционного управления независимых поставщиков. Зато она отличается высокой производительностью и предоставляет администратору замечательные инструментальные средства. С помощью этой программы можно осуществлять дистанционное управление всей сетью, включая клиентские рабочие станции и серверы прежних лет выпуска. Между прочим, взяв на вооружение VNC, есть смысл подумать и о том, чтобы в будущем сделать это средство стандартным компонентом программ развертывания рабочих станций (скажем, включить его в стандартный диск или в образы Microsoft Installation Services - RIS). Утилита WinINSTALL/Limited Edition (LE), распространяемая корпорацией Microsoft на компакт-диске Windows 2000 Server, позволяет формировать установочный пакет .msi, который администратор затем может разворачивать с помощью групповой политики или решений независимых поставщиков.

# 37.Маршрутизаторы в сетях TCP/IP.

Маршрутиза́тор — специализированный компьютер, который пересылает пакеты между различными сегментами сети на основе правил и таблиц маршрутизации. Маршрутизатор может связывать разнородные сети различных архитектур. Для принятия решений о пересылке пакетов используется информация о топологии сети и определённые правила, заданные администратором.

Обычно маршрутизатор использует адрес получателя, указанный в заголовке пакета, и определяет по таблице маршрутизации путь, по которому следует передать данные. Если в таблице маршрутизации для адреса нет описанного маршрута - пакет отбрасывается.

Маршрутом называется путь, по которому пакеты пересылаются от отправителя к получателю.

Маршрут определяет не полный путь, а только сегмент пути от хоста до шлюза (или от шлюза до шлюза), который может переслать пакеты целевому хосту. Существует пять типов маршрутов:

|  |  |
| --- | --- |
| Элемент | Описание |
| маршрут до хоста | Определяет шлюз, который может переслать пакеты указанному хосту в другой сети. |
| маршрут к сети | Определяет шлюз, который может переслать пакеты другому хосту указанной сети. |
| маршрут по умолчанию | Определяет шлюз, которому будут отправлены пакеты, если не был задан маршрут до целевого хоста или маршрут к сети целевого хоста. |
| циклический маршрут | Маршрут по умолчанию для всех пакетов, отправляемых по адресам локальной сети. IP-адрес циклического маршрута всегда 127.0.0.1. |
| маршрут оповещения | Маршрут по умолчанию для всех пакетов оповещения. Каждой подсети, в которой у сети есть IP-адрес, автоматически присваиваются два маршрута оповещения (один - адресу подсети, а другой - адресу оповещения подсети). |

Список маршрутов хранится в таблице маршрутизации ядра. Описание маршрута содержит такую информацию, как список сетей, достижимых локальным хостом, и список шлюзов для отправки пакетов в удаленные сети. При получении дейтаграммы шлюз ищет в таблицах маршрутизации следующий узел ее маршрута до целевого хоста и отправляет дейтаграмму этому узлу.

В таблицу маршрутизации ядра можно добавлять несколько маршрутов к одному и тому же хосту. Процедура выбора маршрута сначала находит все маршруты, соответствующие запросу, а потом выбирает маршрут с минимальной метрикой расстояния. При наличии нескольких маршрутов одинаковой длины выбирается тот маршрут, который задан наиболее точно. Если несколько маршрутов совпадают по обоим критериям, то эти маршруты применяются по-очереди.

Статическая и динамическая маршрутизация

В TCP/IP предусмотрено два типа маршрутизации: статическая и динамическая.

Статическая маршрутизация означает, что таблицы маршрутизации обслуживаются вручную с помощью команды route. Этот тип маршрутизации рекомендуется применять тогда, когда ваша сеть взаимодействует с одной или двумя другими сетями. Однако если сеть соединена с большим числом сетей, то число шлюзов резко возрастает, и для обслуживания таблиц маршрутизации вручную требуется значительное время.

При динамической маршрутизации таблицы маршрутизации автоматически обновляются демонами. Демоны маршрутизации непрерывно получают информацию, рассылаемую путем оповещения другими демонами маршрутизации, поэтому они постоянно обновляют таблицы маршрутизации.

В TCP/IP предусмотрено два демона, поддерживающих динамическую маршрутизацию: routed и gated. Демон gated поддерживает одновременно Протокол информации о маршрутизации (RIP), Протокол информации о маршрутизации следующего поколения (RIPng), Протокол внешних шлюзов (EGP), Протокол граничных шлюзов (BGP) и BGP4+, протокол (HELLO), Протокол кратчайшего пути (OSPF), протоколы IS-IS и ICMP и ICMPv6/Router Discovery. Кроме того, демон gated поддерживает Простой протокол управления сетью (SNMP). Демон routed поддерживает только Протокол информации о маршрутизации.

В зависимости от опций, указанных при запуске демона маршрутизации, он может работать в одном из двух режимов - пассивном или активном. В активном режиме демон маршрутизации периодически отправляет шлюзам и хостам оповещающие сообщения, содержащие информацию о маршрутизации для их локальных сетей, а также получает информацию о маршрутизации от других хостов и шлюзов. В пассивном режиме демон маршрутизации только получает информацию о маршрутизации и не пытается обновить информацию о маршрутизации удаленных шлюзов (то есть он не распространяет собственную информацию о маршрутизации).

Два описанных типа маршрутизации применяются не только шлюзами, но и хостами сети. Статическая маршрутизация применяется для шлюзов точно так же, как и для других хостов. Однако демоны динамической маршрутизации, которые выполняются не на шлюзах, могут работать только в пассивном (тихом) режиме.

Шлюзы маршрутизации TCP/IP

Шлюз - это один из типов маршрутизаторов. Маршрутизаторы соединяют несколько сетей и выполняют функции маршрутизации пакетов. Например, некоторые маршрутизаторы передают данные по маршруту на уровне сетевого интерфейса или на физическом уровне. Шлюзы осуществляют маршрутизацию на сетевом уровне.

Шлюзы пересылают IP-дейтаграммы, полученные от других шлюзов или хостов, хостам локальной сети, а также передают IP-дейтаграммы из одной сети в другую. Например, если шлюз соединяет две сети Token-Ring, то у него есть две карты адаптера Token-Ring, с каждой из которых связан собственный сетевой интерфейс Token-Ring. Шлюз получает дейтаграммы по одному сетевому интерфейсу и отправляет их с помощью другого интерфейса. Периодически шлюзы проверяют состояние своих сетевых соединений с помощью сообщений о состоянии интерфейса.

При пересылке пакетов шлюзы ориентируются на адрес целевой сети, а не на адрес конкретного хоста. То есть шлюз не должен хранить список маршрутов до всех возможных целевых хостов пакета. Шлюз направляет пакет в сеть, к которой подключен целевой хост. За пересылку пакета целевому хосту будут отвечать маршрутизаторы этой сети. Таким образом, обычно для работы шлюза требуется только ограниченный объем оперативной памяти и, возможно, ограниченный объем дисковой памяти.

Расстояние, проходимое сообщением от отправителя к получателю, измеряется числом транзитных участков между шлюзами. Путь до шлюза, расположенного в сети, к которой подключен источник - это нулевой транзитный участок, путь к сети, которая достижима с этого шлюза - первый транзитный участок, и т.д. Расстояние до получателя сообщения обычно измеряется в числе транзитных участков (иногда оно называется метрикой).

Внутренние и внешние шлюзы маршрутизации

Внутренними шлюзами называются шлюзы, которые относятся к одной и той же автономной системе. Эти шлюзы обмениваются сообщениями с помощью Протокола информации о маршрутизации (RIP), Протокола информации о маршрутизации следующего поколения (RIPng), межсистемного протокола ISIS protocol, протокола кратчайшего пути (OSPF) или протокола HELLO. Внешние шлюзы относятся к различным автономным системам. Они работают на основе Протокола внешних шлюзов (EGP), Протокола граничных шлюзов (BGP) или BGP4+.

Для примера рассмотрим две автономные системы. Первая из них состоит из сетей, которые управляются компанией Widget. Вторая система состоит из сетей, которые управляются компанией Gadget. В компании Widget есть один компьютер с именем apple, выполняющий роль шлюза для соединения с Internet. В компании Gadget есть один компьютер с именем orange, который также выполняет роль шлюза для соединений с Internet. Автономные системы обеих компаний состоят из нескольких внутренних сетей. Шлюзы, соединяющие внутренние сети, представляют собой внутренние шлюзы. Однако apple и orange - внешние шлюзы.

Каждый внешний шлюз взаимодействует не со всеми внешними шлюзами. Вместо этого, внешний шлюз регистрирует набор соседей (других внешних шлюзов), с которыми он обменивается информацией. Соседи определяются не по географическому принципу, а в зависимости от соединений, установленных между шлюзами. Соседний шлюз, в свою очередь, обменивается информацией со своими соседними внешними шлюзами. В результате информация о маршрутизации распространяется среди внешних шлюзов, и их таблицы маршрутизации обновляются.

Информация о маршрутизации представлена в виде пар (N, D), где N - это сеть, а D - расстояние до этой сети, указанное в соответствии с применяемой метрикой расстояния. Каждый шлюз рассылает информацию о достижимых сетях и о расстояниях до этих сетей. Шлюз, получивший такое сообщение, подсчитывает наикратчайший путь до других сетей и передает эту информацию своим соседям. Таким образом, каждый внешний шлюз непрерывно получает информацию о маршрутизации, обновляет свои таблицы маршрутизации и передает эту информацию дальше своим соседям.

Протоколы шлюза

Все шлюзы, внутренние и внешние, обмениваются информацией согласно определенным протоколам. Ниже приведено краткое описание наиболее распространенных протоколов шлюзов **TCP/IP**:

**Протокол HELLO (HELLO)**

**HELLO** - это один из протоколов для обмена информацией между внутренними шлюзами. **HELLO** подсчитывает наикратчайший путь к другим сетям на основе минимальной временной задержки.

**Протокол информации о маршрутизации (RIP)**

**Протокол информации о маршрутизации** - это один из протоколов для обмена информацией между внутренними шлюзами. Как и **протокол HELLO**, **RIP** подсчитывает наикратчайший путь к другим сетям. В отличие от **HELLO**, **RIP** оценивает расстояние не на основе временной задержки, а на основе числа транзитных участков. Демон **gated** сохраняет значения всех метрик в виде временных задержек, поэтому он преобразует число транзитных участков **RIP** во временную задержку.

**Протокол информации о маршрутизации следующего поколения**

**RIPng** - это расширение протокола **RIP** для поддержки **IPv6**.

**Протокол кратчайшего пути (OSPF)**

**OPSF** - это протокол для обмена информацией между внутренними шлюзами. Этот протокол основан на обмене сообщениями о состоянии канала связи между маршрутизаторами и сетями, и он лучше приспособлен для сложных сетей со многими маршрутизаторами, чем **RIP**. Кроме того, он поддерживает работу с несколькими равноправными маршрутами.

**Протокол внешних шлюзов (EGP)**

**Протокол внешних шлюзов** предназначен для обмена информацией между внешними шлюзами. **EGP** не подсчитывает наикратчайший путь к другим сетям. Он применяется внешними шлюзами для того, чтобы получить информацию о достижимости сетей.

**Протокол граничных шлюзов (BGP)**

Этот протокол предназначен для обмена информацией между внешними шлюзами. Он позволяет шлюзам различных автономных систем обмениваться информацией о достижимости сетей и предоставляет больше возможностей, чем **EGP**. Дополнительная информация о каждом маршруте хранится в атрибутах **EGP**, которые позволяют выбрать наилучший маршрут.

**Протокол граничных шлюзов 4+**

**BGP4+** - это расширение протокола **BGP** версии 4 с поддержкой **IPv6** и другими улучшениями.

**Межшлюзовый протокол (IS-IS)**

Протокол **IS-IS** применяется внутренними шлюзами при общении друг с другом. Это протокол, хранящий информацию о состоянии канала. Он может передавать пакеты ISO/CLNP и, как и протокол **OSPF**, применяет для определения маршрута алгоритм кратчайшего пути.

*Источник:* <https://www.ibm.com/support/knowledgecenter/ru/ssw_aix_72/com.ibm.aix.networkcomm/tcpip_routing_intro.htm>