Серверные и клиентские сценарии Web-приложений

Соколова В. В.

Архитектура клиент-сервер

- Процесс разработки Web-приложений достаточно сложен и одной из наиболее важных задач является решение о том, как функциональность приложения должна быть распределена между клиентской и серверной частью.
- Решив эту задачу, разработчики получают двух-звенные, трехзвенные и многозвенные архитектуры. Все зависит от того, сколько промежуточных звеньев включается между клиентом и сервером.

Архитектура клиент-сервер

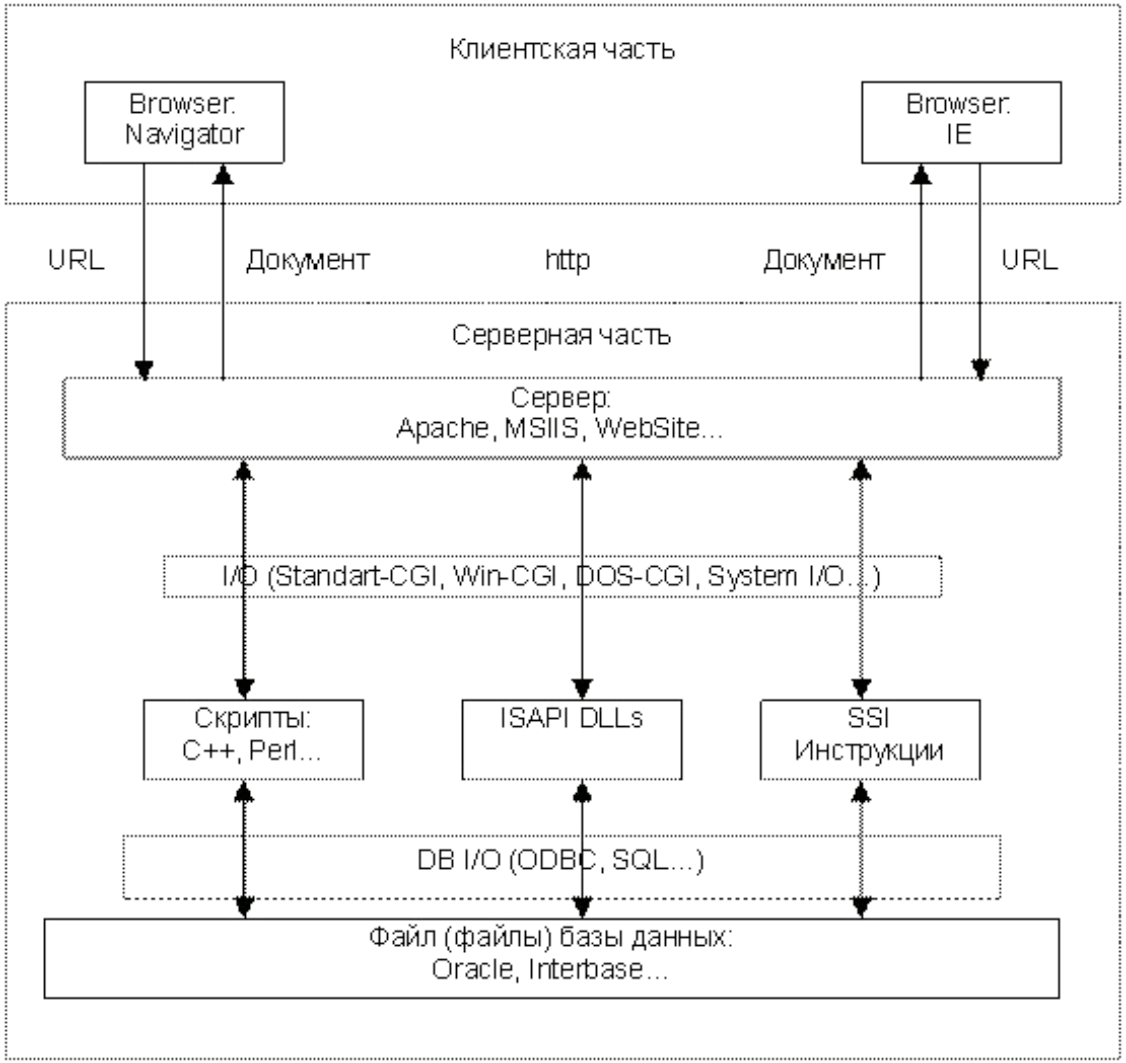
Сеть Интернет организована по схеме клиент-сервер. В классическом случае данная схема функционирует следующим образом:

- клиент формирует и посылает запрос на сервер баз данных;
- сервер производит необходимые манипуляции с данными, формирует результат и передаёт его клиенту;
- клиент получает результат, отображает его на устройстве вывода и ждет дальнейших действий пользоватля.

Цикл повторяется, пока пользователь не закончит работу с сервером.

В сервисе WWW для передачи информации применяется протокол HTTP (HyperText Transmition Protocol).

Схема клиент-сервер WWW-HTTP



Транзакции в HTTP

Основные транзакции в HTTP:

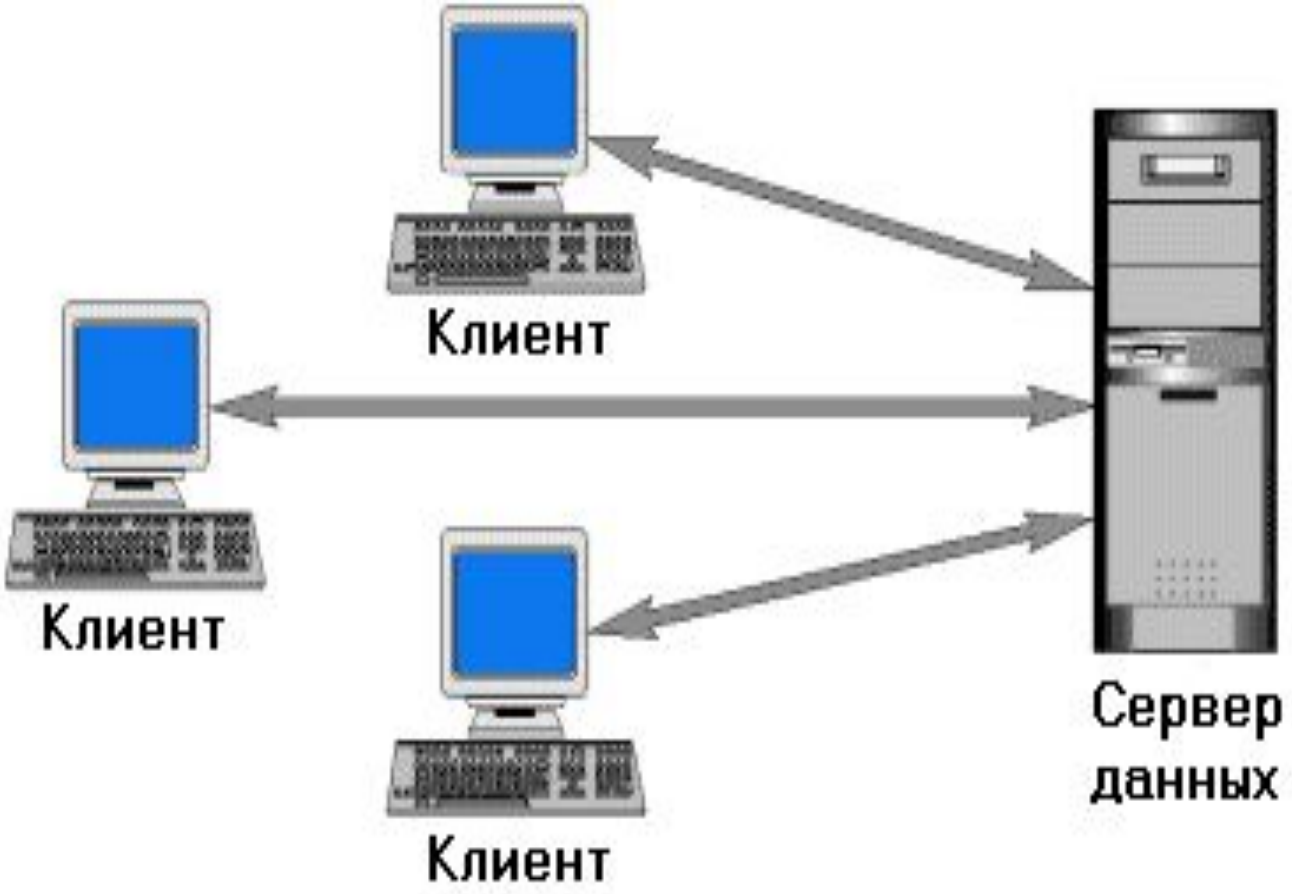
1. Браузер декодирует первую часть URL (Universal Resource Locator) и устанавливает соединение с сервером.
2. Браузер передает остальную часть URL на сервер.
3. Сервер определяет по URL путь и имя файла.
4. Сервер пересылает указанный файл браузеру.
5. Сервер прерывает соединение.
6. Браузер отображает документ.

При данных транзакциях сервер не имеет никакой информации о состоянии браузера, т.е. HTTP можно считать "однонаправленным" протоколом, и взаимодействовать с сервером возможно только через механизм URL, это создает трудности при реализации клиентской части.

Распределение функций в архитектуре "клиент-сервер"

- Основная задача клиентского приложения – это обеспечение интерфейса с пользователем, т. е. ввод данных и представление результатов в удобном для пользователя виде, и управление сценариями работы приложения.
- Основные функции серверной СУБД – обеспечение надежности, согласованности и защищенности данных, управление запросами клиентов, быстрая обработка SQL-запросов.
- В двухзвенной архитектуре вся логика работы приложения (прикладные задачи, бизнес-правила) распределяется между двумя процессами: клиентом и сервером.

Двухзвенная архитектура "клиент-сервер"



Двухзвенная архитектура "клиент-сервер"

1. Архитектура **"толстый клиент – тонкий сервер"**: большая часть функций приложения решалась клиентом, сервер занимался только обработкой SQL-запросов.

Архитектура "толстый" клиент имеет следующие недостатки:

- сложность администрирования;
- усложняется обновление ПО, поскольку его замену нужно производить одновременно по всей системе;
- усложняется распределение полномочий, так как разграничение доступа происходит не по действиям, а по таблицам;
- перегружается сеть вследствие передачи по ней необработанных данных;
- слабая защита данных, поскольку сложно правильно распределить полномочия.

Двухзвенная архитектура "клиент-сервер"

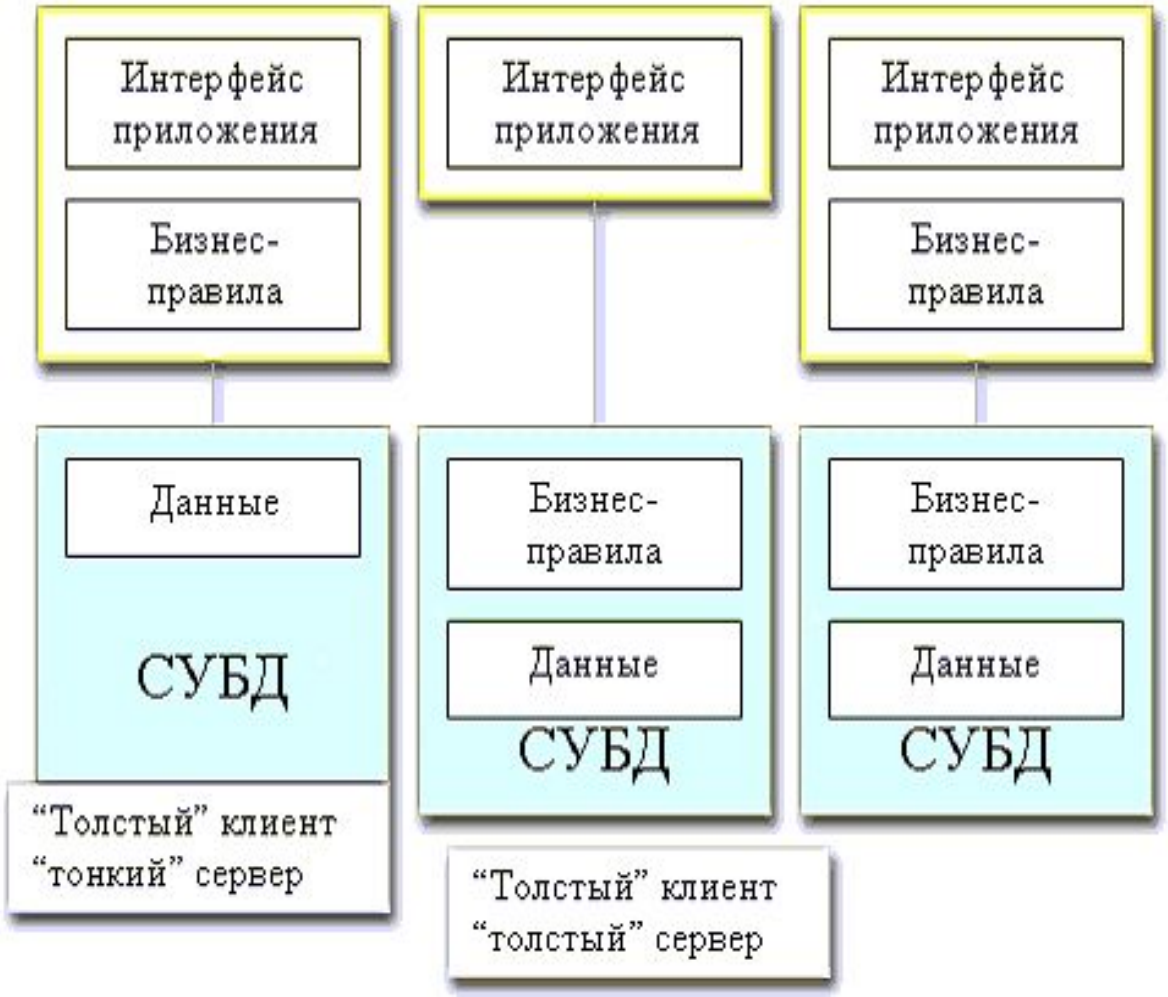
2. Архитектура **"тонкий клиент – толстый сервер"**: использование на сервере хранимых процедур (stored procedure - откомпилированные программы с внутренней логикой работы), привело к тенденции переносить все большую часть функций на сервер. Хранимые процедуры реализовывали часть бизнес-логики и гарантировали выполнение операции в рамках единой транзакции. Такое решение имеет очевидные преимущества, например его легче поддерживать, т. к. все изменения нужно вносить только в одном месте – на сервере.

Архитектура "толстый" сервер имеет следующие недостатки:

- усложняется реализация, так как языки типа PL/SQL не приспособлены для разработки подобного ПО и нет хороших средств отладки;
- производительность программ, написанных на языках типа PL/SQL, значительно ниже, чем созданных на других языках, что имеет важное значение для сложных систем;
- программы, написанные на СУБД-языках, обычно работают недостаточно надежно; ошибка в них может привести к выходу из строя всего сервера баз данных;
- получившиеся таким образом программы полностью непереносимы на другие системы и платформы.

Для решения перечисленных проблем используются многоуровневые (три и более уровней) архитектуры клиент-сервер.

Распределение функций в архитектуре "клиент-сервер"



Многозвенная архитектура "клиент-сервер"

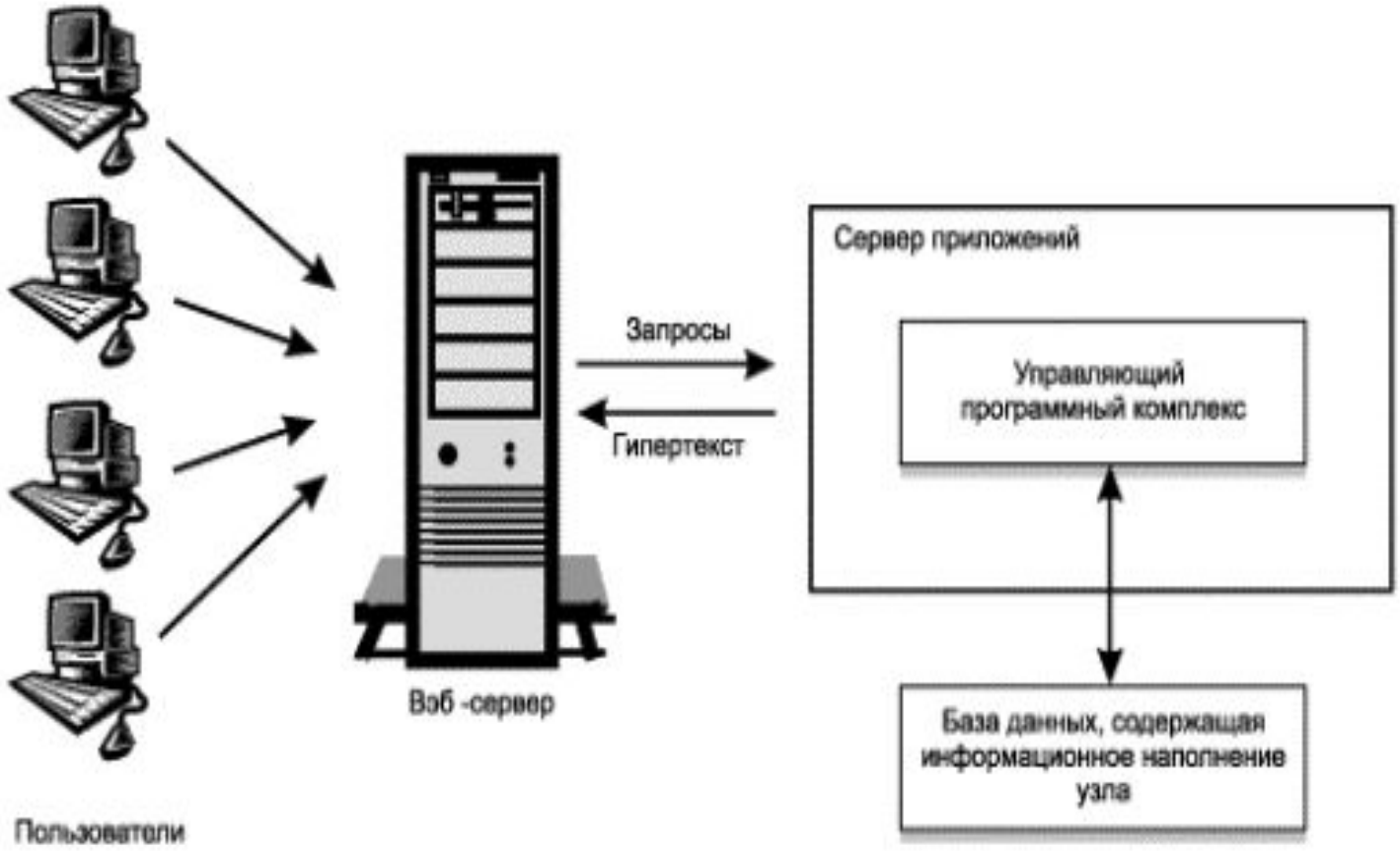
- **Трехзвенная и многозвенная архитектуры "клиент-сервер"**: выполнение прикладных задач и бизнес-правил осуществляется отдельным компонентом приложения (или нескольким компонентам), которые могут работать на специально выделенном компьютере – сервере приложений.

- Сервер приложений обрабатывает следующие компоненты:

1. презентационная логика (Presentation Layer - PL) – предназначена для работы с данными пользователя;
2. бизнес-логика (Business Layer - BL) – предназначена для проверки правильности данных, поддержки ссылочной целостности;
3. логика доступа к ресурсам (Access Layer - AL) – предназначена для хранения данных.

- Подход Remote Data Access (RDA) подразумевает объединение в клиентском приложении PL и BL (однако в случае необходимости выполнения каких-либо изменений в клиентском приложении придется менять исходный код), а серверная часть представляет собой сервер баз данных, реализующий AL.

Трехзвенная архитектура "клиент-сервер"



Многозвенная архитектура "клиент-сервер"

- Любая информационная система, построенная на основе клиент-серверных технологий, должна содержать следующие компоненты:

1. шлюз-сервер, управляющий правами доступа к информационной системе;
2. WWW-сервер;
3. сервер баз данных;
4. сервер приложений и(или) сервер обработки транзакций.

- Взаимодействие WWW сервера с базами данных может быть организовано двумя способами:

1. через сервер (менеджер) транзакций;
2. через API интерфейс WWW сервера или сервера приложений.

Менеджер транзакций

Менеджеры транзакций позволяют одному серверу приложений одновременно обмениваться данными с несколькими серверами баз данных.

Хотя серверы Oracle имеют механизм выполнения распределенных транзакций, но если пользователь хранит часть информации в БД Oracle, часть в БД Informix, а часть в текстовых файлах, то без менеджера транзакций не обойтись.

МТ используется для управления распределенными разнородными операциями и согласования действий различных компонентов информационной системы.

Первые менеджеры транзакций появились в начале 70-х гг. (например, CICS); с тех пор они незначительно изменились идеологически, но весьма существенно - технологически.

Наибольшие идеологические изменения произошли в коммуникационном менеджере, так как в этой области появились новые объектно-ориентированные технологии (CORBA, DCOM и т.д.).

Менеджер транзакций

Менеджер транзакций – это программа или комплекс программ, с помощью которых можно согласовать работу различных компонентов информационной системы.

Логически МТ делится на несколько частей:

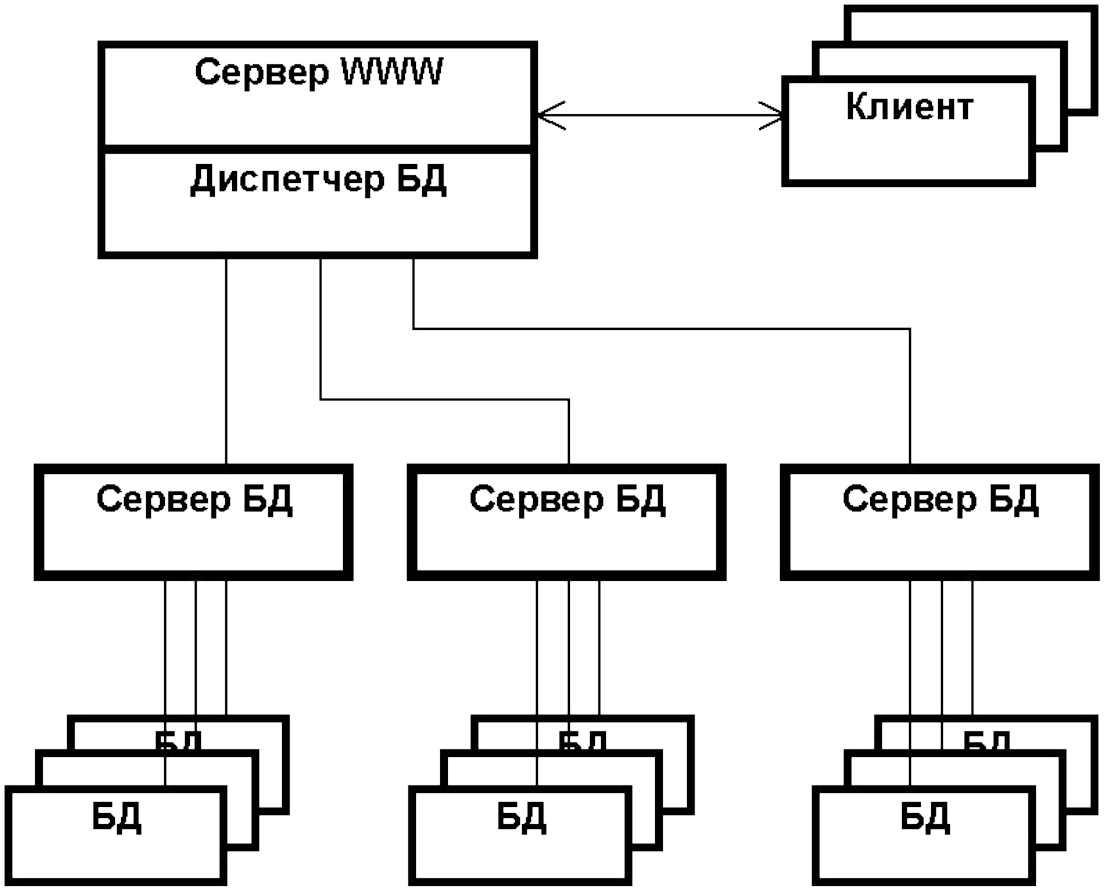
- коммуникационный менеджер (Communication Manager) – контролирует обмен сообщениями между компонентами информационной системы;
- менеджер авторизации (Authorisation Manager) – обеспечивает аутентификацию пользователей и проверку их прав доступа;
- менеджер транзакций (Transaction Manager) – управляет распределенными операциями;
- менеджер ведения журнальных записей (Log Manager) – следит за восстановлением и откатом распределенных операций;
- менеджер блокировок (Lock Manager) – обеспечивает правильный доступ к совместно используемым данным.

Обычно коммуникационный менеджер объединен с авторизационным, а менеджер транзакций работает совместно с менеджерами блокировок и системных записей. Причем такой менеджер редко входит в комплект поставки, поскольку его функции (ведение записей, распределение ресурсов и контроль операций), как правило, выполняет сама база данных (например, Oracle).

Многозвенная архитектура "клиент-сервер"

- Распределенная информационная система представляется в виде трех-четырехуровневой структуры с разграничением функций на каждом уровне и фиксацией протоколов межуровневого потока данных.
- Разграничение на логически замкнутые функциональные уровни необходимо для возможности их реализации на разных физических серверах и добавления в дальнейшем новых возможностей.
- Обмен информацией с уровнем 1 происходит через файловую систему (локальную или сетевую), с уровнем 3 - по протоколам TCP через фиксированный программный порт. В последнем случае для лучшей межплатформенной совместимости данные передаются только в текстовом виде.

Многозвенная архитектура "клиент-сервер"



Многозвенная архитектура "клиент-сервер"

Уровень 1. *Собственно данные* представляют собой обычные файлы данных в формате, необходимом для работы сервера БД. Данные хранятся в виде набора файлов в отдельном каталоге для каждой БД. Кроме собственно данных, каталог может включать информацию о предопределенных форматах для отображения данных и файл заголовка для расширенного названия БД.

Уровень 2. *Сервер баз данных* реализует основные функции выборки информации из БД. Для публичной информационной системы эти функции сводятся к следующим:

- получение запроса с уровня 3;
- логический разбор строки запроса;
- исполнение запроса;
- возврат данных на уровень 3.

В соответствии с этим сервер БД обрабатывает следующие запросы.

Информационный – запрос на информацию о конкретной базе данных. Во входном потоке - идентификатор базы данных сервера БД, в выходном - заголовок, количество записей и комментарий указанной БД, описание поле БД.

Словарный – запрос на список ключевых слов с параметрами. Во входном потоке - идентификатор БД, шаблон ключевого слова, порядковый номер ключевого слова, количество слов в выходном буфере, в выходном - список затребованных ключевых слов и их частота.

Форматный – запрос на предоставление списка предопределенных форматов вывода данных. Во входном потоке - идентификатор БД, в выходном - пронумерованный список предопределенных форматов для данной БД.

Основной – запрос на предоставление данных в требуемом формате с параметрами. Во входном потоке - идентификатор БД, строка запроса, номер записи начала вывода, количество записей для вывода, идентификатор формата, в выходном - форматированная выборка из БД.

Служебный – запрос на номер версии сервера БД. В выходном потоке - номер версии текущего сервера БД, пронумерованный список доступных БД, идентификатор внутренней кодировки сервера БД.

Уровень 3. Сервер WWW с модулем управления серверами БД - *диспетчер БД* - предназначен для обработки запросов пользователей, формирования запросов к серверам БД и возврата клиентам полученной информации по протоколу HTTP и спецификациям HTML. Оптимальным вариантом является Windows NT + IIS с поддержкой JAVA и ASP (Active Server Pages) ввиду тесной интеграции IIS с операционной системой и возможностью организации многопоточной обработки данных сравнительно простыми и дешевыми средствами. Управляющий модуль (диспетчер БД) может быть реализован в виде динамической библиотеки и (или) набора объектов ASP.

Диспетчер БД выполняет следующие функции:

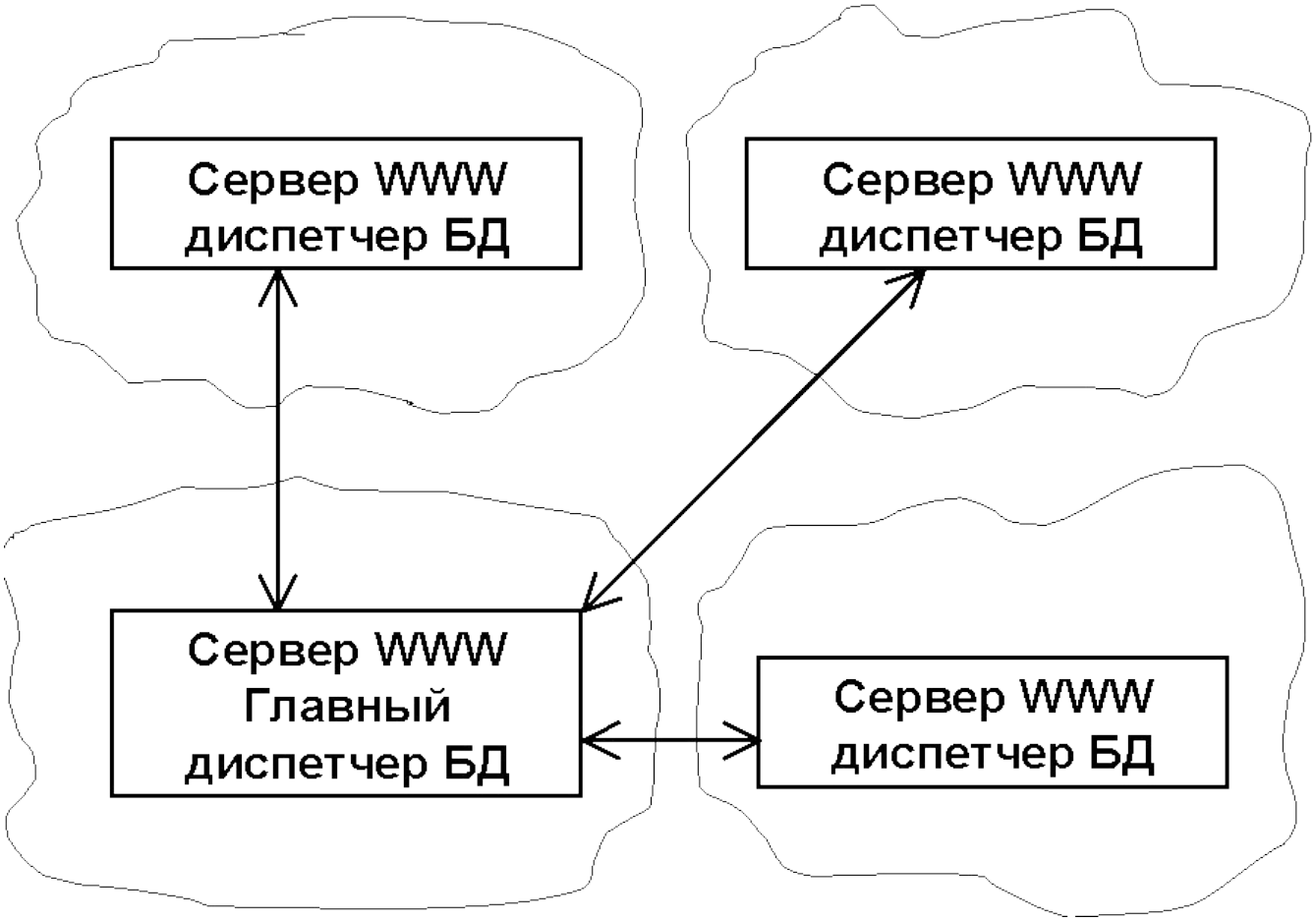
- хранение и предоставление пользователям текущей информации о доступных БД;
- формирование запросов к серверам БД и возвращение клиентам полученной информации в требуемой кодировке;
- хранение информации о правах доступа на каждую доступную БД и проверка их для каждого пользователя;
- учет и сбор статистики обращений к БД в соответствии с текущими установками;
- синхронизация версий серверов БД и их обновление;
- при наличии уровня 4 передача служебной информации о себе и о поддерживаемых базах данных на уровень 4.

Для организации полнофункциональной системы достаточно перечисленных трех уровней. Однако при построении территориально распределенной системы с ярко выраженными районами и ненадежными линиями связи между ними желательно локализовать все три уровня в каждом районе с интеграцией последних на уровне 4.

Уровень 4. *Главный диспетчер (ГД)* информационной системы представляет собой сервер WWW, функционально идентичный серверу уровня 3, но наделенный дополнительной функцией хранения информации о всей информационной системе в целом. В идеальном случае каждый из серверов уровня 3 должен быть готов взять на себя роль главного диспетчера. Основная задача ГД – получить информацию о конфигурации каждого сервера уровня 3 и растиражировать ее по всем серверам.

Таким образом, общая схема распределенной информационной системы состоит из четырех логических уровней.

Интеграция диспетчеров БД на 4 уровне



Основные задачи клиентских и серверных сценариев

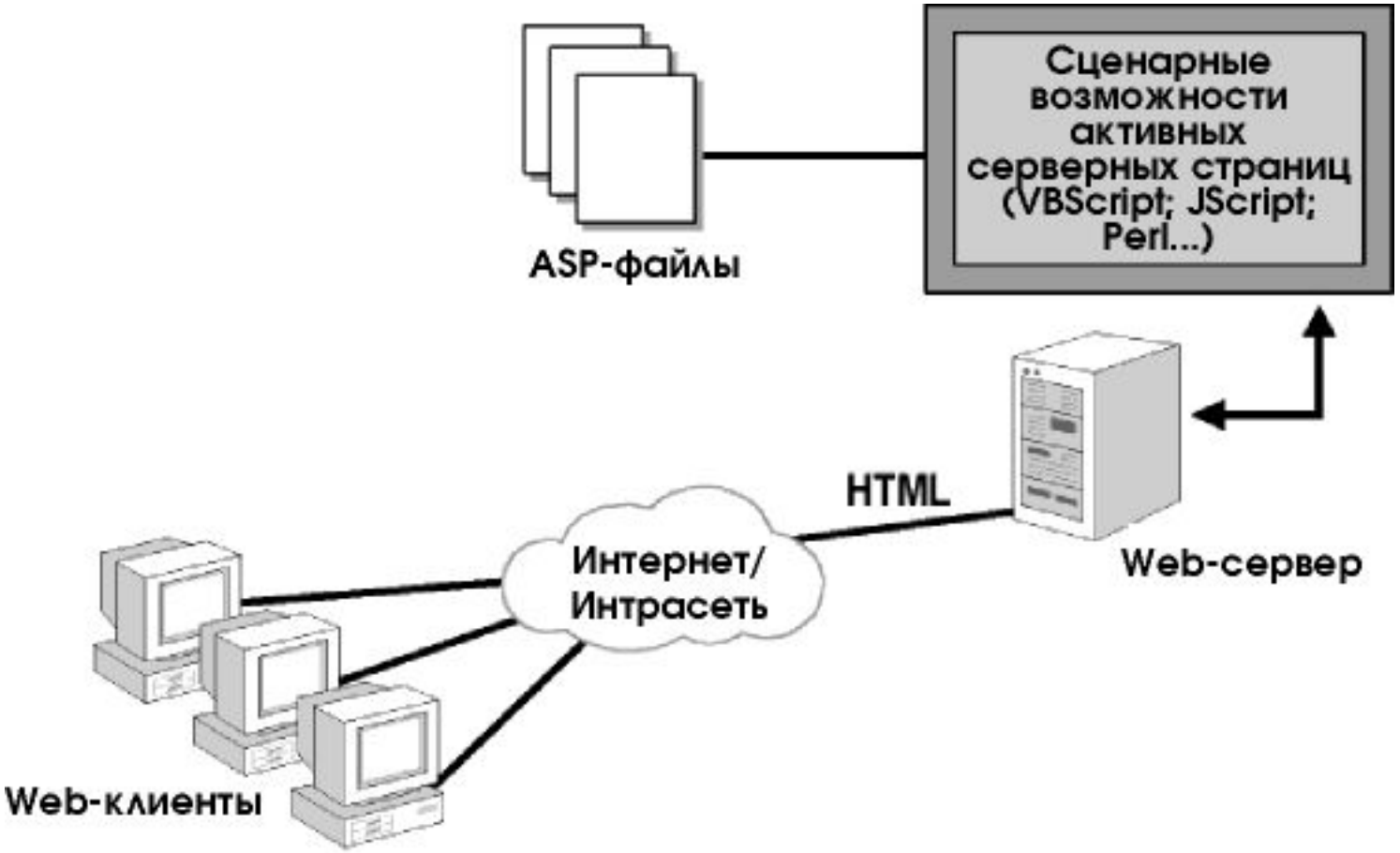
- Клиентский сценарий выполняется на компьютере пользователя в процессе взаимодействии с Web-страницей и позволяет решать следующие задачи:

1. верифицировать значения элементов управления формы;
2. реализовать событийные процедуры для элементов управления.

- Серверный сценарий выполняется на Web-сервере до передачи страницы пользователю и позволяет:

1. обеспечить доступ к базе данных и возврат данных пользователю;
2. хранить информацию о состоянии пользователя или сеанса.

Серверные и клиентские сценарии



Клиентские сценарии

- Клиентский сценарий выполняется на компьютере-клиенте. Программы просмотра снабжены встроенным интерпретатором, который может считывать и выполнять сценарии.
- Основная цель добавления клиентского сценария к Web-странице — создание событийных процедур для элементов управления. Например, событийная процедура будет запускать определенную функцию, когда пользователь нажмет соответствующую кнопку.
- Клиентские сценарии в HTML-странице не компилируются и не шифруются. Поэтому, если посмотреть исходный HTML-код Web-страницы, можно увидеть текст встроенного сценария.
- Чтобы сценарий клиентской части функционировал, программа просмотра должна поддерживать язык, на котором он написан. В противном случае пользователь не получит полного доступа к сценарным средствам Web-страницы.

Серверные сценарии

- Серверный сценарий выполняется в рамках активной страницы на Web-сервере до того, как тот вернет пользователю готовую HTML-страницу. Когда пользователь запрашивает активную серверную страницу, сервер выполняет сценарии и создает HTML-код, который и передается пользователю. В результате пользователь не видит серверного сценария на полученной Web-странице.
- Поскольку серверный сценарий выполняется на Web-сервере, ему доступны все ресурсы сервера – например, базы данных и исполняемые файлы.
- Для работы серверных сценариев Web-сервер должен поддерживать технологию активных страниц; к программе просмотра же не предъявляется никаких дополнительных требований, поскольку Web-клиент в данном случае получает стандартную HTML-страницу. Таким образом, сценарии серверной части не зависят от клиентов.

Реализация клиентских сценариев

- Чтобы расширить функциональные возможности Web-страницы средствами клиентских сценариев, исходный текст сценария надо встроить в HTML-страницу в виде ASCll-текста. Встретив ее в тексте страницы, программа просмотра вызывает интерпретатор сценария, который анализирует и выполняет код. Программа просмотра должна поддерживать выполнение сценариев и их интеграцию с элементами управления ActiveX или Java-апплетами, встроенными в HTML-страницу.

- **Языки разработки сценариев:**

1. **Visual Basic Scripting Edition (VBScript)** – не зависит от регистра символов и совместимо снизу вверх с Visual Basic for Applications. Microsoft Internet Explorer поддерживает VBScript средствами VBScript Interpreter — быстрого кросс-платформенного интерпретатора; лицензию на него бесплатно выдает компания Microsoft.
2. **JavaScript (JScript)** – реализован Microsoft и подобен C: в его основе лежит Java – язык программирования, разработанный компаниями Sun Microsystems и Netscape. JavaScript поддерживают как Netscape Navigator, так и Internet Explorer.

VBScript и JavaScript похожи – как в одном, так и в другом можно определять переменные, создавать процедуры и обращаться к свойствам и методам объектов.

Разница между ними – небольшие отличия в синтаксисе. Ни один из них не компилируется, и оба работают на всех аппаратных платформах. Это интерпретируемые языки, поэтому скорость исполнения определяется возможностями программы просмотра, а не характеристиками самого языка.

Механизмы, реализующие серверную часть обработки данных

1. **Internet Server Application Programming Interface (ISAPI)** – интерфейс программирования приложений сервера Интернета реализуется через механизм библиотек DLL.

Приложения ISAPI являются динамически подключаемыми библиотеками. Такая библиотека с интерфейсными функциями загружается WEB-сервером один раз и остается в памяти, после чего она будет готова отвечать на любое количество запросов. Каждый клиентский запрос обслуживается в отдельном потоке.

Библиотеки DLL работают как часть процесса WEB-сервера, выполняясь в том же пространстве адресов памяти, в котором работает и сам WEB-сервер. Вместо передачи информации в обе стороны в виде файлов, теперь расширения WEB-серверов передают информацию в пределах одного и того же адресного пространства, без необходимости записи в файл. Благодаря этому WEB-приложения стали работать быстрее, с большей эффективностью и с меньшим потреблением ресурсов.

С помощью ISAPI Internet connector возможно взаимодействие с базами данных через драйверы ODBC, также возможна реализация других расширенных функций (создание различных фильтров запросов). Основным средством разработки приложений является Microsoft Visual C++ (также VB, Delphi), который поддерживается Microsoft Internet Information Server.

2. **Server Sides Includes (SSI/SSI+)** – технология динамического формирования документов.

Скрипт (серверные инструкции) находится в HTML файле обычно имеющем расширение sht или shtm, при этом серверные инструкции размещаются между специальными разделителями (tokens), а сами инструкции записаны на языке Cscript. При пересылке такой файл сканируется сервером на наличие SSI инструкций и результат динамически подставляется в посылаемый документ.

SSI реализуется через специальные компоненты (DLL), которые входят в состав сервера. Данная технология опирается на использование разнообразных объектов и компонент (COM, ActiveX и т. п.), работа с которыми ведётся средствами языков VBScript или JavaScript.

Механизмы, реализующие серверную часть обработки данных

3. **Common Gateway Interface (CGI) –** интерфейс общего шлюза реализуется через дополнительные программы (скрипты) на любом из языков программирования высокого уровня (C++, Perl, VisualBasic, Pascal, Java).

По сути CGI – способ взаимодействия Web-программ с браузером пользователя Основа – спецификация набора переменных. С помощью CGI приложений возможно взаимодействие с любыми базами данных через формирование SQL запросов, или другие механизмы; также возможна реализация счетчиков посещений, гостевых книг и других расширений.

CGI обеспечивает способ, посредством которого Web-браузер осуществляет запуск Web-приложения на стороне сервера, результатом работы которого является HTML-страница, посылаемая клиенту. Всякий раз, когда клиент инициирует выполнение CGI-приложения, Web-сервер выполняет отдельную его копию (instance).

Недостатки CGI-приложений:

- Для каждого запроса клиента запускается копия Web-приложения на сервере, что резко сокращает производительность сервера при больших и средних нагрузках
- Каждый запрос должен запускать на сервере свой собственный процесс, выделенная ему на сервере область памяти не пересекается с областью памяти приложения web-сервера. И поэтому несколько запросов могут существенно замедлить работу даже умеренно загруженного сервера - ведь ему приходится выполнять такие относительно медленные задачи, как создание файла, запуск отдельного процесса, его выполнение, запись и возвращение другого файла.

Большинство CGI-программ пишется на языке Perl (Practical Extraction and Report Language), который является одним из наиболее гибких языковых средств, служащих для программирования интерфейсов CGI. Изначально Perl предназначался для обработки больших объемов данных и генерации отчетов по обработке этих данных, но за последние несколько лет Perl превратился в полнофункциональный язык программирования.

Технология Java

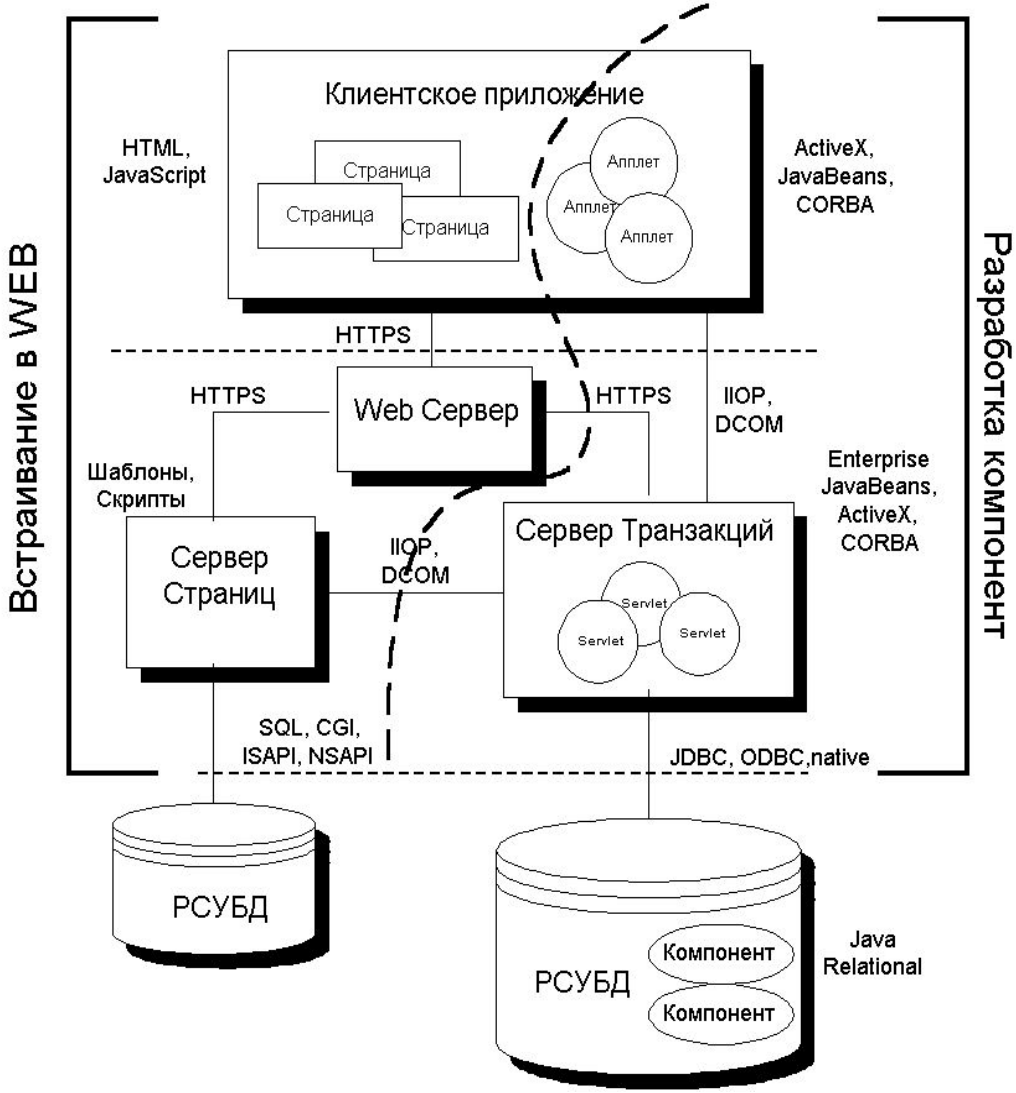
Технология Java – позволяет строить универсальные системы со смешанной архитектурой:

1. апплетами (applets) – приложения, выполняемые на стороне клиента,
2. сервлеты (servlets) – приложения, выполняемые на стороне сервера.

Аплеты пишутся на Java и посылаются по Web как HTML-файлы браузеру, где выполняются как HTML-документы. Существенным преимуществом Java является независимость программ от платформ, на которых программы выполняются. Хотя Java не обязательно выполняется в окне браузера, возможно создание независимых (stand-alone) Java-приложений, которые могут выполняться на компьютере независимо от Интернета.

Фактически программа на языке Java транслируется компилятором в специальный код, называемый байтовым (bytecode), а затем выполняется уже с помощью интерпретатора языка Java. Такое «разделение обязанностей» и позволяет обеспечивать полную независимость Java-кода от конечной платформы, на которой он будет выполняться. Для каждой конкретной платформы имеется свой интерпретатор языка, называемый виртуальной машиной Java (Java Virtual Machine).

Архитектура распределенного приложения



Активные серверные страницы (ASP)

- ASP комбинирует сценарий ActiveX и команды HTML для того, чтобы получить динамическую страницу HTML. Сценарии ASP отличаются от сценариев, базирующихся на браузерах.

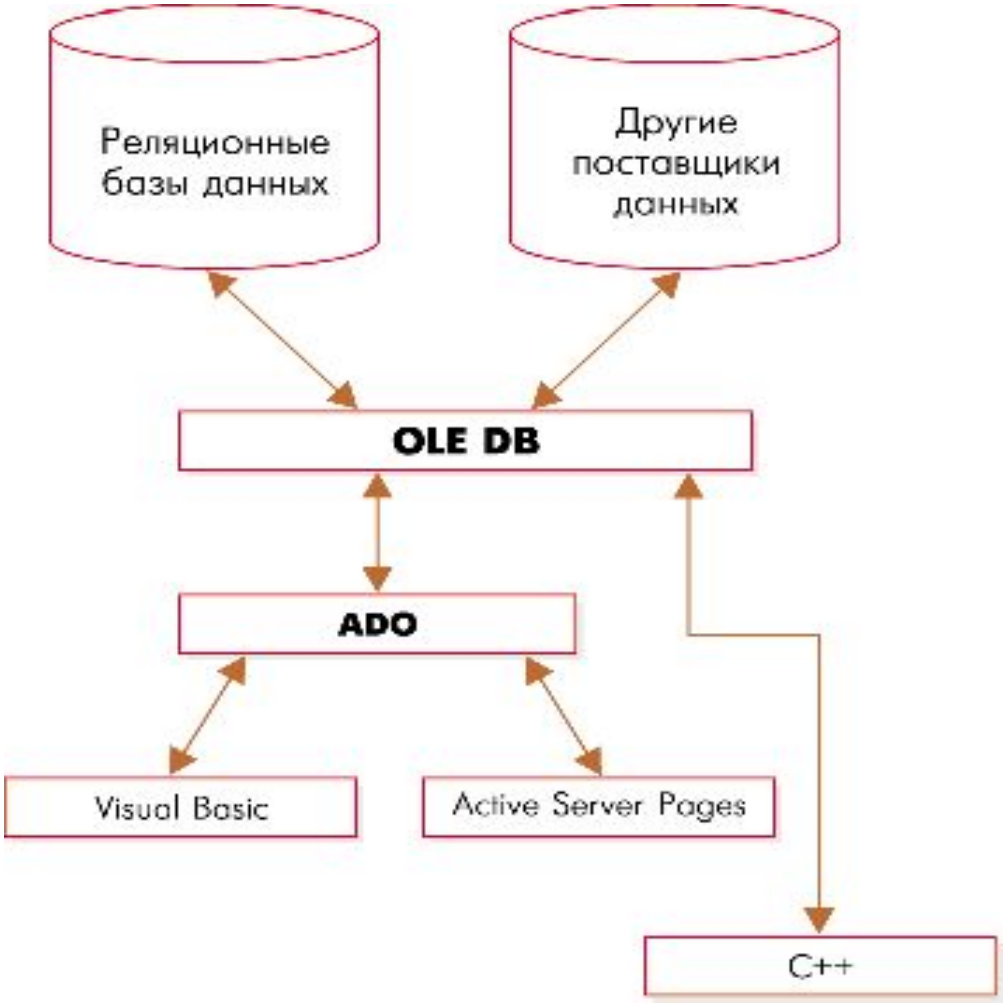
1. В традиционных сценариях, основывающихся на браузерах, WEB-сервер посылает страницу HTML, содержащую сценарий ActiveX в браузер клиента, который и отвечает за выполнение сценария. Подход, при котором основной акцент делается на клиентской части приложения, возлагает на нее дополнительный груз обязанностей, что может привести к возникновению проблем, если клиентский браузер не будет в состоянии выполнить сценарий.
2. Напротив, страницы ASP исполняются на WEB-сервере IIS. В ходе исполнения страницы сервер напрямую посылает клиенту команды HTML и все клиентские сценарии, содержащиеся на странице ASP. Но как только сервер доходит до команды серверного сценария ASP, то он исполняет этот сценарий и передает клиенту в форме HTML только полученные в качестве результата выходные данные.
3. Клиент, действия которого сводятся к использованию браузера, не видит разницы между потоком страниц HTML, порождаемым сценарием ASP, и потоком HTML, посылаемым статичными WEB-страницами.

- Таким образом, написание сценариев для серверной стороны с помощью ASP создает WEB-страницы, которые выступают в качестве исполнителей сценариев. Тот факт, что ASP генерирует только поток страниц HTML, обеспечивает независимость от типа браузера клиента.
- В силу того, что сервер IIS интерпретирует страницы ASP «на лету», ASP служит идеальным средством для встраивания результатов обработки интерактивных запросов к базе данных в WEB-страницы. Эти возможности обеспечиваются доступом к базе данных через ADO непосредственно со страниц ASP.

Использование объектов ADO на страницах ASP

- При использовании ADO, приложение первым делом пытается применить объекты Соединение (Connection), Команда (Command) или Набор записей (Recordset) для установления соединения с сервером баз данных. Объект Соединение следует употреблять для того, чтобы открыть соединение ADO явным образом. Объекты Команда и Набор записей позволяют сделать то же самое динамически.
- После установления соединения приложение ASP может выполнять команды ADO такого же типа, что и стандартное приложение, написанное на Visual Basic. Эти команды включают исполнение хранимых процедур, открытие и просмотр набора записей, вставку, обновление и удаление данных.
- Поставщик OLE DB для ODBC позволяет использовать структуру объекта ADO с большинством существующих драйверов ODBC. Но поставщик OLE DB для SQL Server дает возможность подключиться только к SQL Server. Однако с объектами ADO Соединение, Команда и Набор записей возможно применять любой из упомянутых поставщиков.

Использование объектов ADO на страницах ASP



Изменение данных средствами ADO

- ASP и ADO можно применять не только для динамической выдачи WEB-страниц, но и в целях создания WEB-страниц для ввода данных. Такая возможность позволяет создавать основанные на WEB приложения с использованием баз данных, обладающие таким же набором функций работы с базами данных, что и стандартные приложения, разработанные в соответствии с архитектурой клиент-сервер.
- Объекты ADO Набор записей (Recordset), которые становятся доступными на страницах ASP, предоставляют тот же перечень услуг, что и приложения, написанные на Visual Basic. Их можно применять для ввода данных, изменения или удаления данных.
- Все остальные возможности ADO, такие как способность запускать подготовленные заранее операторы SQL или хранимые процедуры, также имеют место.

Модель объектов ASP

- Активные серверные страницы, в качестве автоматического сервера OLE, обладают иерархической структурой.
- Первичным объектом в программной модели ASP является объект **Контекст сценария (ScriptingContext)**, который обеспечивает взаимодействие с браузером клиента. Поскольку объект Контекст сценария всегда доступен приложениям ASP, то нет необходимости в явном виде делать на него ссылку.
- Объект Контекст сценария содержит шесть основных объектов ASP, среди которых 5 встроенных объектов, позволяющих расширить функциональные возможности Web-приложения.
- Средствами встроенных объектов можно обеспечить совместное использование информации Web-приложения, сохранить данные о конкретном пользователе, получить сведения, передаваемые серверу, отправить сообщение адресату и манипулировать свойствами и методами серверных компонентов.

Модель объектов ASP

- Основные объекты ASP:

1. объект Приложение (Application) – обеспечивает совместное использование данных всеми клиентами Web-приложения.
2. объект Запрос (Request) – получает информацию, переданную пользователем Web-серверу при HTTP-запросе.
3. объект Сервер (Server) – предоставляет доступ к ресурсам Web-сервера.
4. объект Сессия (Session) – сохраняет сведения о сеансе конкретного пользователя.
5. объект Отклик (Response) – управляет передачей пользователю информации в ответном HTTP-сообщении.
6. объект Контекст объекта (ObjectContext).

- Все активные WEB-сессии применяют объект Приложение (Application) для того, чтобы все пользователи могли одновременно обращаться к информации приложения ASP.
- Объект Приложение (Application) включает две коллекции:

1. Содержание (Context) - каждый объект Содержание соответствует какому-либо пункту, для включения которого в WEB-приложение были использованы команды ActiveX.
2. Статические объекты (StaticObjects) - коллекция Статические объекты содержит все объекты, для включения которых в WEB-приложение применялись ярлыки HTML.