РЕФЕРАТ

по дисциплине «Информатика»

на тему «Архитектуры реализации корпоративных информационных систем»

**1. Архитектура КЛИЕНТ-СЕРВЕР**

При построении корпоративных информационных сетей, как правило, используются две базовые архитектуры: Клиент-сервер и Интернет/Интранет. В чем же преимущества и недостатки использования каждой из данных архитектур и когда их применение оправдано? Найти ответы на эти вопросы мы постараемся в данном разделе.

Одной из самых распространенных на сегодня архитектур построения корпоративных информационных систем является архитектура КЛИЕНТ-СЕРВЕР.

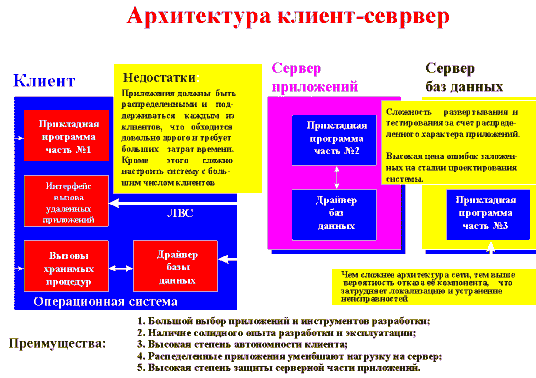


Рис.1. Компоненты архитектуры Клиент-сервер и их свойства.

В реализованной по данной архитектуре информационной сети клиенту предоставлен широкий спектр приложений и инструментов разработки, которые ориентированы на максимальное использование вычислительных возможностей клиентских рабочих мест, используя ресурсы сервера в основном для хранения и обмена документами, а также для выхода во внешнюю среду. Для тех программных систем, которые имеют разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, при этом, предоставляя возможность приложениям либо непосредственно адресоваться к другим серверным приложениям, либо маршрутизировать запросы к ним. Средством (инструментарием) для реализации клиентских модулей для ОС Windows в данном случае является, как правило, Delpfi.

Однако при этом частые обращения клиента к серверу снижают производительность работы сети, кроме этого приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределенный характер построения системы обуславливает сложность ее настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре КЛИЕНТ-СЕРВЕР, тем выше вероятность отказа любого из ее компонентов.

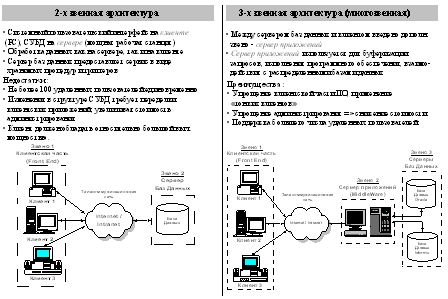


Рис.2. Сравнительные характеристики двух- и трехзвенной архитектуры клиент-сервер.

В последнее время все большее развитие получает архитектура Интернет/Интранет. В основе реализации корпоративных информационных систем на базе данной архитектуры лежит принцип "открытой архитектуры", что во многом определяет независимость реализации корпоративной системы от конкретного производителя. Все программное обеспечение таких систем реализуется в виде аплетов или сервлетов (программ написанных на языке JAVA) или в виде cgi-модулей (программ написанных как правило на Perl или С).

Основными экономическими преимуществами данной архитектуры являются:

* относительно низкие затраты на внедрение и эксплуатацию;
* высокая способность к интеграции существующих гетерогенных информационных ресурсов корпораций;
* повышение уровня эффективности использования оборудования (сохранение инвестиций).
* прикладные программные средства доступны с любого рабочего места, имеющего соответствующие права доступа;
* минимальный состав программно-технических средств на клиентском рабочем месте (теоретически необходима лишь программа просмотра - броузер и общесистемное ПО);
* минимальные затраты на настройку и сопровождение клиентских рабочих мест, что позволяет реализовывать системы с тысячами пользователей (причем многие из которых могут работать за удаленными терминалами).

**2. Архитектура Интернет/Интранет**

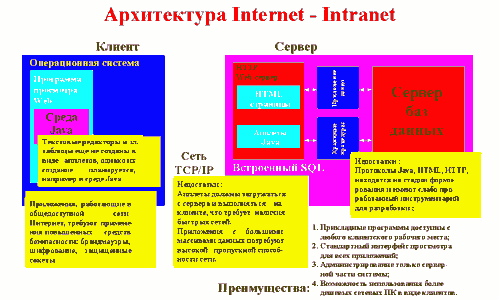


Рис.3. Компоненты архитектуры Интернет-Интранет и их свойства.

В общем случае АИС, реализованная с использованием данной архитектуры включает Web-узлы с интерактивным информационным наполнением, реализованных при помощи технологий Java, JavaBeans и JavaScript, взаимодействующих с предметной базой данных, с одной стороны, и с клиентским местом с другой. База данных, в свою очередь, является источником информации для интерактивных приложений реального времени.

По запросу клиента WEB узел осуществляет следующие операции (рис.7):

* Отправляет ASCII коды HTML страниц (или VRML документов), включающие при необходимости элементы javaScript;
* Отсылает двоичный код запрошенного ресурса (изображения, адио-, видеофайла, архива и т.п.);
* Отсылает байт коды JAVA апплетов.
* Принимает конкретную информацию от пользователя (результат заполнения активной формы, или статистическую информацию запрошенную CGI скриптом);
* Осуществляет заполнение базы данных;
* Принимает сообщения от пользователя и регламентирует доступ к ресурсам Web узла на основе анализа принятой информации (проверка паролей и т.п.);
* Принимает информацию от пользователя и в зависимости от нее динамически формирует HTML страницы, либо VRML документы, обращаясь, при необходимости, к базам данных и существующим на WEB узле HTML страницам и VRML документам.

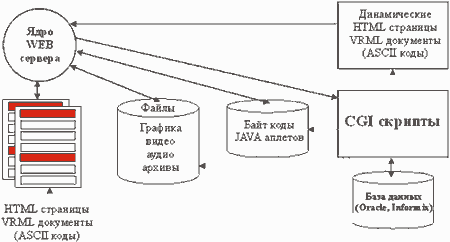


Рис.4. Информационные взаимосвязи компонентов WEB узла

После того, как клиент получил ответ WEB сервера, он осуществляет следующие операции:

* визуализирует HTML страницу либо VRML документ в окне броузера;
* интерпретирует команды JavaScript, модифицирует образ HTML страницы и т.п.;
* интерпретируя байт коды JAVA апплетов, позволяет загружать и выполнять активные приложения;
* ведет диалог с пользователем, заполняющим формы, и создает новые запросы к WEB серверу;
* с помощью утилит воспроизводит коды аудио и видео файлов, поддерживает мультимедийные средства;
* обеспечивает моделирование виртуальной реальности просматривая VRML документы.

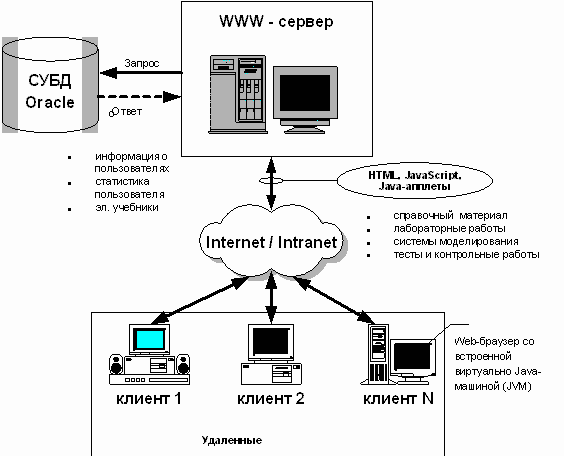


Рис.5. Функциональная схема интерактивного взаимодействия пользователей в архитектуре интернет/интранет.

Перечисленные задачи WEB клиента обеспечиваются возможностями броузера и специализированным программным обеспечением (утилитами), размещенными на рабочей станции клиента. Следует отметить и тот факт, что жестких стандартов на построение WEB клиента пока нет и его компонентный состав может различаться.

На сегодняшний день известны и широко применяются три основных технологии создания интерактивного взаимодействия с пользователем в Web. Первый путь заключается в использовании Стандартного Интерфейса Шлюза (Commom Gateway Interface) - CGI. Второй - включение JavaScript - сценариев в тело Web-страниц. И наконец самый мощный, предоставляющий практически неограниченные возможности способ - применение технологии Java (ипользование Java-аплетов).

CGI - это механизм для выбора, обработки и форматирования информации. Возможность взаимодействия, обеспечиваемая CGI, предоставляется во многих формах, но в основном это динамический доступ к информации, содержащейся в базах данных. Например, многие узлы применяют CGI для того, чтобы пользователи могли запрашивать базы данных и получать ответы в виде динамически сформированных Web-страниц (рис.6).

Имеются в виду узлы, предоставляющие доступ к базам данных, средствам поиска, и даже информационные системы, предающие сообщения в ответ на ввод пользователя. Все эти узлы используют CGI, чтобы принять ввод пользователя и передать его с сервера Web-базе данных. База данных обрабатывает запрос и возвращает ответ серверу, который в свою очередь пересылает его опять браузеру для отображения. Без СGI база данных этого не смогла бы. Данный интерфейс можно считать посредником между браузером, сервером и любой информацией которая должна передаваться между ними.

В отличии от HTML, CGI не является языком описания документов. Собственно, это и не язык вообще; это стандарт. Он просто определяет, как серверы Web передают информацию, используя приложения, исполняемые на сервере. Это способ расширения возможностей сервера Web без преобразования при этом его самого. Подобно тому как броузер Web обращается к вспомогательным приложениям для обработки информации, которую он не понимает, CGI предоставляет серверу Web возможность преложить работу на другие приложения, такие как базы данных и средства поиска.

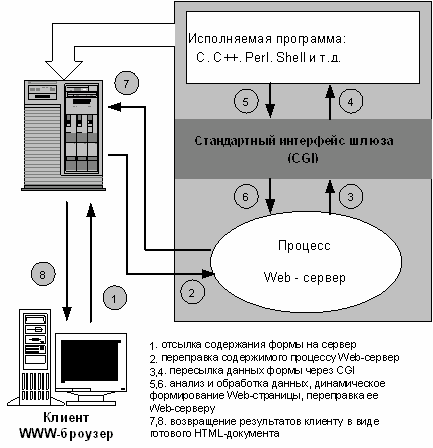


Рис.6. Схема взаимодействия между браузером, сервером и сценарием CGI

При написании программы шлюза (которая может конвертировать ввод из одной системы в другую) CGI позволяет использовать почти любой язык программирования. Способность использовать при написании программы шлюза любой язык, даже язык сценариев, чрезвычайно важна. Самыми популярными языками являются shell, Perl, C и С++. Сценарием традиционно называют программу, которая выполняется с помощью интерпретатора, выполняющего каждую строку программы по мере ее считывания.

Последовательность действий при взаимодействии клиента с программой запущенной на Web-сервере можно сформулировать как следующая последовательность шагов.

1. Браузер принимает введенную пользователем информацию, как правило с помощью форы.
2. Браузер помещает введенную пользователем информацию в URL, указывающий имя и местоположение сценария CGI, который требуется ввести в действие.

архитектура клиент сервер интранет

1. Браузер подключается к серверу Web и запрашивает URL. Сервер определяет, что URL должен ввести в действие сценарий CGI, и запускает указанный сценарий.
2. Сценарий CGI выполняется, обрабатывая все передаваемые ему данные.
3. Сценарий CGI динамически формирует Web-страницу и возвращает результат серверу.
4. Сервер возвращает результат клиенту.
5. Браузер отображает результат пользователю

Это является упрощенной схемой взаимодействия между браузером, сервером и сценарием CGI. Наибольшую популярность CGI - сценарии нашли при использовании в качестве обработчиков форм, средства доступа к базам данных, средства осуществления локального и глобального поиска, шлюзовых протоколов.

Наибольшей мощью в реализации клиентского программного обеспечения обладают аплеты - программы написанные на языке JAVA. В узком смысле слова Java - это объектно-ориентированный язык, напоминающий C++, но более простой для освоения и использования. В более широком смысле Java - это целая технология программирования, изначально рассчитанная на интеграцию с Web-сервисом, то есть на использование в сетевой среде. Поскольку Web-навигаторы существуют практически для всех аппаратно-программных платформ, Java-среда должна быть как можно более мобильной, в идеале полностью независимой от платформы.

С целью решения перечисленных проблем были приняты, помимо интеграции с Web-навигатором, два других важнейших постулата.

1. Была специфицирована виртуальная Java-машина (JVM), на которой должны выполняться (интерпретироваться) Java-программы. Определены архитектура, представление элементов данных и система команд Java-машины. Исходные Java-тексты транслируются в коды этой машины. Тем самым, при появлении новой аппаратно-программной платформы в портировании будет нуждаться только Java-машина; все программы, написанные на Java, пойдут без изменений.
2. Определено, что при редактировании внешних связей Java-программы и при работе Web-навигатора прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных по сети (в частности, на WWW-сервере). Найденные объекты загружаются, а их методы выполняются затем на машине пользователя.

Несомненно, между двумя сформулированными положениями существует тесная связь. В компилируемой среде трудно абстрагироваться от аппаратных особенностей компьютера, как трудно (хотя и можно) реализовать прозрачную динамическую загрузку по сети. С другой стороны, прием объектов извне требует повышенной осторожности при работе с ними, а, значит, и со всеми Java-программами. Принимать необходимые меры безопасности проще всего в интерпретируемой, а не компилируемой среде. Вообще, мобильность, динамизм и безопасность - спутники интерпретатора, а не компилятора.

Принятые решения делают Java-среду идеальным средством разработки интерактивных клиентских компонентов (апплетов) Web-систем. Особо отметим прозрачную для пользователя динамическую загрузку объектов по сети. Из этого вытекает такое важнейшее достоинство, как нулевая стоимость администрирования клиентских систем, написанных на Java. Достаточно обновить версию объекта на сервере, после чего клиент автоматически получит именно ее, а не старый вариант. Без этого реальная работа с развитой сетевой инфраструктурой практически невозможна.

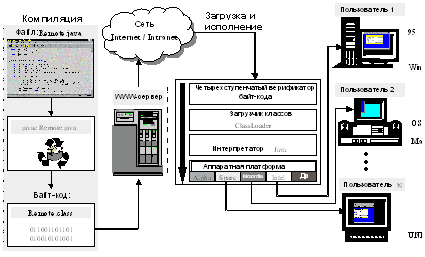


Рис.7. Java технологии при реализации АИС.

Стандартный реляционный доступ к данным очень важен для программ на Java, потому что Java-апплеты по природе своей не являются монолитными, самодостаточными программами. Будучи модульными, апплеты должны получать информацию из хранилищ данных, обрабатывать ее и записывать обратно для последующей обработки другими апплетами. Монолитные программы могут себе позволить иметь собственные схемы обработки данных, но Java-апплеты, пересекающие границы операционных систем и компьютерных сетей, нуждаются в опубликовании открытых схем доступа к данным.

Интерфейс JDBC (Java Database Connectivity - связанность баз данных Java) является первой попыткой реализации доступа к данным из программ Java, не зависящего от платформы и базы данных. В версии JDK 1.1 JDBC является составной частью основного Java API.

JDBC - это набор реляционных объектов и методов взаимодействия с источниками данных. Программа на языке Java открывает связь с таблицей, создает объект оператор, передает через него операторы SQL системе управления базой данных получает результаты и служебную информацию о них. В типичном случае файлы .class JDBC и апплет/приложение на языке Java находятся на компьютере клиенте. Хотя они могут быть загружены из сети, для минимизации задержек во время выполнения лучше иметь классы JDBC у клиента. Система управления базой данных (CУБД) и источник данных обычно расположены на удаленном сервере.

На рисунке 8 показаны различные варианты реализаций связи JDBC с базой данных. Апплет/приложение взаимодействует с JDBC в системе клиента, драйвер отвечает за обмен информацией с базой данных через сеть.

Классы JDBC находятся в пакете java.sql.\*. Все программы Java используют объекты и методы из этого пакета для чтения и записи в источник данных. Программе, использующей JDBC, требуется драйвер к источнику данных, с которым она будет взаимодействовать. Этот драйвер может быть написан на другом (не Java) языке программирования, или он может являться программой на языке Java, которая общается с сервером, используя RPC (Remote Procedure Call) - удаленный вызов процедур или HTTP. Обе схемы приведены на рис.19. Драйвер JDBC может быть библиотекой на другом (не Java), как программа сопряжения ODBC - JDBC, или классом Java, который общается через сеть с сервером базы данных, используя RPC или HTTP. Допускается, что приложение будет иметь дело с несколькими источниками данных, возможно, с неоднородными. По этой причине у JDBC есть диспетчер драйверов, чьи обязанность заключаются в управлении драйверами и предоставлении программе списка загруженных драйверов.

Хотя словосочетание "База данных" входит в расшифровку аббревиатуры JDBC, форма, содержание и расположение данных не интересуют программу Java, использующую JDBC, поскольку существует драйвер к этим данным.

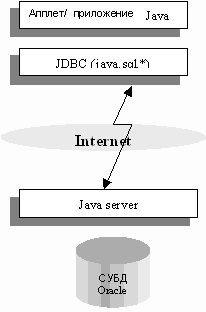


Рис.8 Варианты реализации связи JDBC с базой данных

**Сопряжение JDBC - ODBC**

В качестве составной части JDBC поставляется драйвер для доступа из JDBC к источникам данных ODBC (Open Database Connectivity), и называется "программа сопряжения JDBC - ODBC". Эта программа сопряжения реализована в виде JdbcOdbc.class и является библиотекой для доступа к драйверу ODBC.

Поскольку JDBC конструктивно близок к ODBC, программа сопряжения является несложной надстройкой над JDBC. На внутреннем уровне этот драйвер отображает методы Java в вызовы ODBC и тем самым взаимодействует с любым ODBC - драйвером. Достоинство такой программы сопряжения состоит в том, что JDBC имеет доступ к любым базам данных, поскольку ODBC - драйверы распространены очень широко.

В соответствии с правилами Internet JDBC идентифицирует базу данных при помощи URL, который имеет форму:

jdbc:<субпротокол>:<имя, связанное с СУБД или Протоколом>

У баз данных в Internet/intranet "имя" может содержать сетевой URL

//<имя хоста>:<порт>/..

<субпротокол> может быть любым именем, которое понимает база данных. Имя субпротокола "odbc" зарезервированно для источников данных формата ODBC. Типичный JDBC URL для базы данных ODBC выглядит следующим образом:

jdbc:odbc:<DNS - имя ODBC>;User=<имя пользователя>; PW=<пароль>

Внутреннее устройство JDBC – приложения

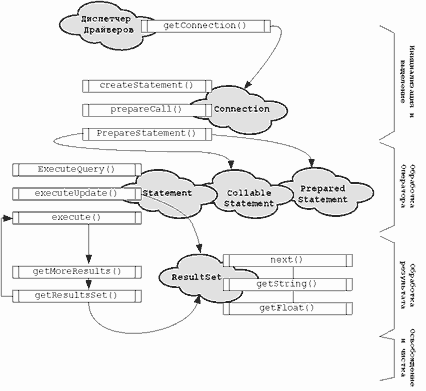


Рис.9. Иерархия классов JDBC и поток API JDBC

Чтобы обработать информацию из базы данных, информационно-обучающая система на языке Java выполняет ряд шагов. На рис.9 показаны основные объекты JDBC, методы и последовательность выполнения, Во-первых, программа вызывает метод getConnection (), чтобы получить объект Connection.Затем она создает объект Statement и подготавливает оператор SQL.

Оператор SQL может быть выполнен немедленно (объект Statement), а может быть откомпилирован (объект PreparedStatement) или представлен в виде вызова процедуры (объект CallableStatement). Когда выполняется метод executeQuery(), возвращается объект ResultSet. Операторы SQL, такие как updatе или delete не возвращают ResultSet. Для таких операторов используется метод executeUpdate(). Он возвращает целое, указывающее количество рядов, затронутых оператором SQL.

ResultSet содержит ряды данных и анализируетcя методом next(). Если приложение обрабатывает транзакции, можно пользоваться методами rollback() и commit() для отмены или подтверждения изменений, внесенных оператором SQL.

Примеры запроса и модификации базы данных с использованием JDBC

Данный пример иллюстрирует как при помощи SQL - опрератора SELECT составляется список всех студентов из базы данных. Ниже приводятся шаги, которые необходимы для выполнения этого задания при помощи API JDBC. Каждый шаг имеет форму текста на языке Java с комментариями.

// описать методы и переменные

public void ListStudents () throws SQLException

{

int i, noOfColumns;

String stNo, stFName, stLName;

// инициализировать и загрузить драйвер JDBC-ODBC

Class.forName ("jdbc.odbc.JdbcOdbcDriver");

// создать объект Connection

Connection ex1Con = DriverManager.getConnection (

"jdbc:odbc:StudentDB;uid="admin";pw="sa"");

// создать простой объект Statement

Statement ex1Stmt = ex1Con.createStatement ();

// Создать строку SQL, передать ее СУБД и

// выполнить SQL-оператор

ResultSet ex1rs = ex1Stmt.executeQuery (

"SELECT StudentNumber, FirstName, LastName FROM Students");

// Обработать каждый ряд и вывести результат на консоль

System.out.println ("Student Number First Name Last Name");

while (ex1rs.next())

{

stNo = ex1rs.getString (1);

stFName = ex1rs.getString (2);

stLName = ex1rs.getString (3);

System.out.println (stNo, stFName, stLName);

}

}

В следующем примере поле firstName таблицы Students изменяется. Доступ осуществляется через поле StudentNumber.

// описать методы, переменные и параметры

public void UpdateStudentName (String stFName, String stLName, String stNo)

throws SQLException

{

int retValue;

//инициализировать и загрузить драйвер JDBC-ODBC

Class.forName ("jdbc.odbc.JdbcOdbcDriver");

// создать объект Connection

Connection ex1Con = DriverManager.getConnection (

"jdbc:odbc:StudentDB;uid="admin";pw="sa"");

// создать простой объект Statement

Statement ex1Stmt = ex1Con.createStatement ();

// Создать строку SQL, передать ее СУБД и

// выполнить SQL-оператор

String SQLBuffer = "UPDATE Students SET FirstName =" +

stFName + ", lastName =" + stLName +

"WHERE StudentNumber = " + stNo;

retValue = ex1Stmt.executeUpdate (SQLBuffer);

System.out.println ("Модифицированно " + retValue +

" строк в базе данных.")

}

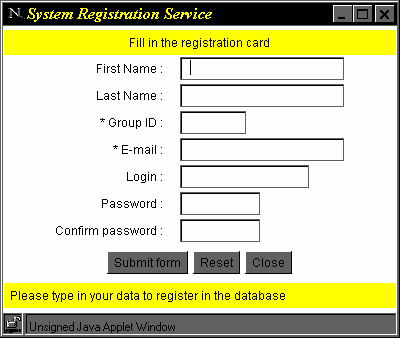


Рис.11. Интерфейс для регистрации пользователя в АИС.

Таким образом, взаимодействие с базами данных из Java также отличается простотой и гибкостью, связанной с эффективной реализацией JDBC API. В сочетании со своей природной платформо-независимостью, Java предоставляет уникальный инструмент для создания интерактивных распределенных информационно-обучающих систем на база Internet/Intranet - технологий.

Основными сложностями при реализации корпоративных систем на базе данной архитектуры являются:

* отсутствие многих популярных приложений и средств разработки реализованных в виде JAVA аплетов;
* относительное высокое время компиляции аплетов на клиентских местах (временно);
* вопросы безопасной работы в сети.

**Список литературы**

1. Попов И.Г., Мамонов С.Г. Информационные системы. М.: Инфра, 2007.

2. Абросимов А.Г. Бородинова М.А. Теория экономических информационных систем. Учебное пособие - Самара. Изд-во Самарск.гос. экон. акад., 2007.

3. Информационные системы. Учебник /Петров В.Н. – СПб.: Питер, 2008.

4. Информационное обеспечение систем управления. Учебное пособие/Голенищев Э.П., Клименко И.В. - Ростов н/Д: Феникс, 2009.

5. Интеллектуальные информационные системы в экономике. Учебное пособие/Тельнов Ю.Ф. Издание третье, расширенное и доработанное. Серия «Экономика и бизнес». – Москва.: СИНТЕГ, 2009.