# ТЕЗИСЫ

## Основные понятия информационных систем как комплекса программных приложений.

Информационные системы являются программным продуктом, как правило представляющим комплекс программных приложений и имеют ряд существенных отличий от стандартных прикладных программ и систем. В зависимости от предметной области информационные системы могут очень сильно различаться по своим функциям, архитектуре, реализации. Можно выделить ряд свойств, которые являются общими:

- информационные системы предназначены для сбора, хранения и обработки информации, с учетом этого, в основе любой из них лежит среда хранения и доступа к данным;
- информационные системы ориентируются на конечного пользователя, не обладающего высокой квалификацией в области применения вычислительной техники. Поэтому клиентские приложения информационной системы должны обладать простым, удобным, легко осваиваемым интерфейсом, который предоставляет конечному пользователю все необходимые для работы функции, но в то же время не дает ему возможность выполнять какие-либо лишние действия.

В большинстве случаев, при разработке информационной системы приходится решать две основные задачи:

- задачу разработки базы данных, предназначенной для хранения информации;
- задачу разработки графического интерфейса пользователя клиентских приложений.

Важным компонентом информационной системы является система управления базой данных (СУБД). Тип используемой СУБД обычно определяется масштабом информационной системы, например малые информационные системы могут использовать локальные СУБД, в корпоративных же информационных системах потребуется мощная клиент-серверная СУБД, поддерживающая многопользовательскую работу.

В настоящее время наиболее широко распространены реляционные СУБД. Несмотря на очевидную привлекательность и растущую популярность объектно-ориентированных СУБД, пока все же преобладают реляционные базы данных, являющиеся хорошо отлаженными, развитыми, сопровождаемыми системами, поддерживающими стандарт ANSI SQL-92 (к таким системам относятся, например, Oracle, Informix, Sybase, DB2, MS SQL Server и т.п.).

Традиционным методом организации информационных систем является двухзвенная архитектура клиент-сервер. В этом случае вся прикладная часть информационной системы размещается на рабочих станциях, а на стороне сервера осуществляется только доступ к базе данных. Чтобы разгрузить клиентскую рабочую станцию и уменьшить загрузку сети, применяются трехзвенные архитектуры клиент-сервер. В этой архитектуре кроме клиентской части системы и сервера базы данных вводится промежуточный сервер приложений. На стороне клиента выполняются только интерфейсные действия, а вся логика обработки информации поддерживается в сервере приложений.

При разработке базы данных необходимо учитывать специфику той СУБД, для которой эта разработка проводится. Несмотря на существование стандарта ANSI SQL 92, практически все SQL-серверы используют свои реализации SQL, содержащие расширения стандарта. Тем не менее на начальном этапе, при разработке общей структуры базы данных (на уровне концептуальной модели), особенности используемой СУБД можно не учитывать.

Первым шагом в проектировании информационной системы является получение формального описания предметной области, построение полных и непротиворечивых функциональных и информационных моделей информационной системы. Это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Следует также учитывать, что в процессе создания и функционирования информационной системы потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

Указанные сложности способствовали появлению программно-технологических средств специального класса, так называемых CASE-средств, призванных повысить эффективность разработки программного обеспечения. Термин CASE (Computer Aided Software/System Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения, в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных информационных систем в целом. В настоящее время под CASE-средствами понимаются программные средства, поддерживающие процессы создания и

сопровождения информационных систем, включая анализ и формулировку требований, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

Еще один класс задач, решаемых при проектировании информационных систем, относится к созданию удобного и соответствующего целям информационной системы пользовательского интерфейса. Следует понимать, что задача эргономичности интерфейса не формализуется, но в то же время она является очень существенной. Пользователи часто судят о качестве системы в целом, исходя из качества ее интерфейса. Более того, от качества интерфейса зависит эффективность использования системы.

Разработка интерфейса всегда являлась трудоемкой задачей, отнимающей много времени у разработчиков. Однако в последние годы появились так называемые *средства визуальной разработки приложений,* в значительной мере упростившие задачу разработки графического интерфейса пользователя. Сейчас на рынке программных продуктов предлагается довольно много разнообразных средств визуальной разработки приложений, ориентированных на разработку информационных систем. Все их можно условно разделить на два класса:

- специализированные средства - ориентированные исключительно на работу с вполне определенной СУБД и не предназначенные для разработки обычных приложений, не использующих базы данных. Примером средств такого рода может служить система Power Builder фирмы Sybase;
- универсальные средства, которые могут использоваться как для разработки информационных приложений, взаимодействующих с базами данных, так и для разработки любых других приложений, не использующих базы данных. Из таких средств наибольшей известностью пользуются системы Borland Delphi фирмы Borland и Visual Basic фирмы Microsoft. Каждый из указанных классов имеет свои достоинства и недостатки, поэтому в общем случае трудно отдать предпочтение одному из них.

Например, Delphi базируется на объектно-ориентированном языке Object Pascal, который наилучшим образом подходит для учебных целей вследствие своей строгости и простоты. Кроме того, в Object Pascal в полной мере реализованы все основные концепции объектно-ориентированного программирования. Объектно-ориентированное программирование позволяет сделать любую систему более гибкой и динамичной, исключив необходимость в постоянном переписывании структуры базы данных и приложений.

Главное достоинство объектно-ориентированного проектирования заключается в возможности повторно использовать ранее написанный код. Кроме того, объектные системы несут в себе возможность модификации и развития. Применительно к базам данных это положение позволяет начать проектирование будущей системы, не имея исчерпывающего представления о предметной области. Поскольку получение детальной информации о предметной области - процесс весьма трудоемкий, то применение объектно-ориентированного подхода позволит сократить сроки и уменьшить стоимость разработки системы. Под информационной системой обычно понимается прикладная программная подсистема, ориентированная на сбор, хранение, поиск и обработку текстовой и/или фактографической информации. Подавляющее большинство информационных систем работает в режиме диалога с пользователем.

В наиболее общем случае типовые программные компоненты, входящие в состав информационной системы, включают:

- диалоговый ввод-вывод;
- логику диалога;
- прикладную логику обработки данных;
- логику управления данными;
- операции манипулирования файлами и (или) базами данных.

Корпоративной информационной системой (КИС) мы будем называть совокупность специализированного программного обеспечения и вычислительной аппаратной платформы, на которой установлено и настроено программное обеспечение.

В последнее время все больше руководителей начинают отчетливо осознавать важность построения на предприятии корпоративной информационной системы как необходимого инструментария для успешного управления бизнесом в современных условиях.

Можно выделить три наиболее важных фактора, существенно влияющих на развитие корпоративных информационных систем:

- развитие методик управления предприятием;
- развитие общих возможностей и производительности компьютерных систем;

- развитие подходов к технической и программной реализации элементов информационной системы.

Рассмотрим эти факторы более подробно.

*Развитие методик управления предприятием*

Теория управления предприятием представляет собой довольно обширный предмет для изучения и совершенствования. Это обусловлено широким спектром постоянных изменений ситуации на мировом рынке. Все время растущий уровень конкуренции вынуждает руководителей компаний искать новые методы сохранения своего присутствия на рынке и поддержания рентабельности своей деятельности. Такими методами могут быть диверсификация, децентрализация, управление качеством и многое другое. Современная информационная система должна отвечать всем нововведениям в теории и практике менеджмента. Несомненно, это самый главный фактор, так как построение продвинутой в техническом отношении системы, которая не отвечает требованиям по функциональности не имеет смысла.

*Развитие общих возможностей и производительности компьютерных систем*

Прогресс в области наращивания мощности и производительности компьютерных систем, развитие сетевых технологий и систем передачи данных, широкие возможности интеграции компьютерной техники с самым разнообразным оборудованием позволяют постоянно наращивать производительность информационных систем и их функциональность.

*Развитие подходов к технической и программной реализации элементов информационных систем*

Параллельно с развитием аппаратной части информационных систем на протяжении последних лет происходит постоянный поиск новых, более удобных и универсальных, методов программно-технологической реализации информационных систем. Можно выделить три наиболее существенных новшества, оказавших большое влияние на развитие информационных систем в последние годы:

- *новый подход к программированию:* объектно-ориентированное программирование фактически вытеснило модульное; до настоящего времени непрерывно совершенствуются методы построения объектных моделей. Благодаря внедрению объектно-ориентированных технологий программирования существенно сокращаются сроки разработки сложных информационных систем, упрощаются их поддержка и развитие;
- благодаря *развитию сетевых технологий* локальные информационные системы повсеместно вытесняются клиент-серверными и многоуровневыми реализациями;
- *развитие сети Интернет* принесло большие возможности работы с удаленными подразделениями, открыло широкие перспективы электронной коммерции, обслуживания покупателей через Интернет и многое другое. Более того, определенные преимущества дает использование Интернет-технологий в интрасетях предприятия (так называемые интранет-технологии).

Следует иметь в виду, что использование определенных технологий при построении информационных систем не является самоцелью разработчика. Выбор технологий должен производиться в зависимости от реальных потребностей.

*Основные составляющие корпоративных информационных систем*

В составе корпоративных информационных систем можно выделить две относительно независимых составляющих:

- *компьютерную инфраструктуру* организации, представляющую собой совокупность сетевой, телекоммуникационной, программной, информационной и организационной инфраструктур. Данная составляющая обычно называется *корпоративной сетью.*
- *взаимосвязанные функциональные подсистемы,* обеспечивающие решение задач организации и достижение ее целей.

Первая составляющая отражает системно-техническую, структурную сторону любой информационной системы. По сути, это основа для интеграции функциональных подсистем, полностью определяющая свойства информационной системы, определяющие ее успешную эксплуатацию. Требования к компьютерной инфраструктуре едины и стандартизованы, а методы ее построения хорошо известны и многократно проверены на практике.

Вторая составляющая корпоративной информационной системы полностью относится к прикладной области и сильно зависит от специфики задач и целей предприятия. Данная составляющая полностью базируется на компьютерной инфраструктуре предприятия и определяет прикладную функциональность информационной системы. Требования к функциональным подсистемам сложны и зачастую противоречивы, так как выдвигаются специалистами из различных прикладных областей. Однако, в конечном счете именно эта

составляющая более важна для функционирования организации, так как для нее, собственно, и строится компьютерная инфраструктура.

*Соотношение между составляющими информационной системы*

Взаимосвязи между двумя указанными составляющими информационной системы достаточно сложны. С одной стороны, эти две составляющие в определенном смысле независимы. Например, организация сети и протоколы, используемые для обмена данными между компьютерами, абсолютно не зависят от того, какие методы и программы планируется использовать на предприятии для организации бухгалтерского учета.

С другой стороны, указанные составляющие в определенном смысле все же зависят друг от друга. Функциональные подсистемы в принципе не могут существовать без компьютерной инфраструктуры. В то же время компьютерная инфраструктура сама по себе достаточно ограничена, поскольку не обладает необходимой функциональностью. Невозможно эксплуатировать распределенную информационную систему при отсутствии сетевой инфраструктуры. Хотя, имея развитую инфраструктуру, можно предоставить сотрудникам организации ряд полезных общесистемных служб (например, электронную почту и доступ в Интернет), улучшающих работу и делающих ее более эффективной (в частности, за счет использования более развитых средств связи).

Таким образом, разработку информационной системы целесообразно начинать с построения компьютерной инфраструктуры (корпоративной сети) как наиболее важной составляющей, опирающейся на апробированные промышленные технологии и гарантированно реализуемой в разумные сроки в силу высокой степени определенности как в постановке задачи, так и в предлагаемых решениях. Бессмысленно строить корпоративную сеть как некую самодостаточную систему принимая во внимание прикладную функциональность. Если в процессе создания системно-технической инфраструктуры не проводить анализ и автоматизацию управленческих задач, то средства, инвестированные в разработку корпоративной сети не дадут впоследствии реальной отдачи.

Корпоративная сеть создается на многие годы вперед, капитальные затраты на разработку и внедрение настолько велики, что практически исключают возможность полной или частичной переделки существующей сети.

Функциональные подсистемы, в отличие от корпоративной сети, изменчивы по своей природе, так как в предметной области деятельности организации посто происходят более или менее существенные изменения. Функциональность информационных систем сильно зависит от организационно-управленческой струк организации, ее функциональности, распределения функций, принятых в организации финансовых технологий и схем, существующей технологии документооборота и множества других факторов.

Разработку и внедрение функциональных подсистем можно выполнять постепенно. Например, сначала на наиболее важных и ответственных участках выполнять разработки, обеспечивающие прикладную функциональность системы (внедрение системы финансового учета, управления кадрами и т. п.), а затем распространять прикладные программные системы и на другие, первоначально менее значимые области управления предприятием.

**Классификация информационных систем**

Информационные системы классифицируются по разным признакам. Рассмот наиболее часто используемые способы классификации.

Кла**ссификация по масштабу**

По масштабу информационные системы подразделяются на следующие группы :

- одиночные;
- групповые;
- корпоративные.

**Одиночные информационные системы**

*Одиночные информационные системы* реализуются, как правило, на автономном локальном компьютере (сеть не используется). Такая система может содержать несколько простых приложений, связанных общим информационным фондом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место. Подобные приложения создаются с помощью так называемых *настольных* или локальных систем управления базами данных (СУБД), Среди локальных СУБД наиболее известными являются Clarion, Clipper, FoxPro, Paradox, dBase и Microsoft Access.

**Групповые информационные системы**

*Групповые информационные системы* ориентированы на коллективное использование информации членами рабочей группы и чаще всего строятся на базе локальной вычислительной сети. При разработке таких приложений используются серверы баз данных (называемые также

SQL-серверами) для рабочих групп. Существует довольно большое количество различных SQL-серверов, как коммерческих, так и свободнораспространяемых. Среди них наиболее известны такие серверы баз данных, как Oracle, DB2, Microsoft SQL Server, InterBase, Sybase, Informix.

**Корпоративные информационные системы**

*Корпоративные информационные системы* являются развитием систем для рабочих групп, они ориентированы на крупные компании и могут поддерживать территориально разнесенные узлы или сети. В основном они имеют иерархическую структуру из нескольких уровней. Для таких систем характерна архитектура клиент--сервер со специализацией серверов или же многоуровневая архитектура. При разработке таких систем могут использоваться те же серверы баз данных, что и при разработке групповых информационных систем. Однако в крупных информационных системах наибольшее распространение получили серверы Oracle, DB2 и Microsoft SQL Server.

Для групповых и корпоративных систем существенно повышаются требования к надежности функционирования и сохранности данных. Эти свойства обеспечиваются поддержкой целостности данных, ссылок и транзакций в серверах баз данных.

**Классификация по сфере применения**


Рис. 1. Деление информационных систем по сфере применения

По сфере применения информационные системы обычно подразделяются на четыре группы :
- системы обработки транзакций;
- системы принятия решений;
- информационно-справочные системы;
- офисные информационные системы.

*Системы обработки транзакций,* в свою очередь, по оперативности обработки данных, разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций — OLTP (OnLine Transaction Processing), для отражения актуального состояния предметной области в любой момент времени, а пакетная обработка занимает весьма ограниченную часть. Для систем OLTP характерен регулярный (возможно, интенсивный) поток довольно простых транзакций, играющих роль заказов, платежей, запросов и т. п. Важными требованиями для них являются:
-высокая производительность обработки транзакций;
- гарантированная доставка информации при удаленном доступе к БД по телекоммуникациям.

*Системы, поддержки принятия решений* - DSS (Decision Support Systeq) - представляют собой другой тип информационных систем, в которых с помощью довольно сложных запросов

производится отбор и анализ данных в различных разрезах: временных, географических и по другим показателям.
Обширный класс *информационно-справочных систем* основан на гипертекстовых документах и мультимедиа. Наибольшее развитие такие информационные системы получили в сети Интернет.
Класс *офисных информационных систем* нацелен на перевод бумажных документов в электронный вид, автоматизацию делопроизводства и управление документооборотом.
Следует отметить, что приводимая классификация по сфере применения в достаточной степени условна. Крупные информационные системы очень часто обладают признаками всех перечисленных выше классов. Кроме того, корпоративные информационные системы масштаба предприятия обычно состоят из ряда подсистем, относящихся к различным сферам применения.

**Классификация по способу организации**

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы (рис. 2):
- системы на основе архитектуры файл-сервер;
- системы на основе архитектуры клиент-сервер;
- системы на основе многоуровневой архитектуры;
- системы на основе Интернет/интранет-технологий.

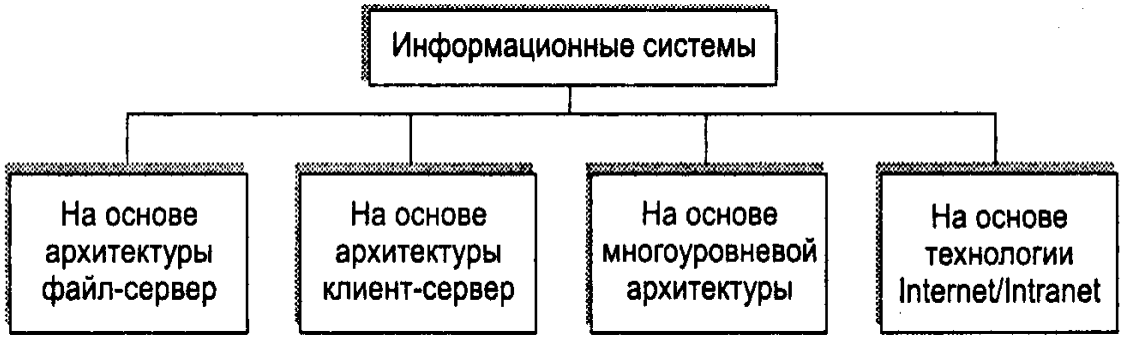


Рис. 2. Деление информационных систем по способу организации

В любой информационной системе можно выделить необходимые функциональные компоненты (табл. 1), которые помогают понять ограничения различных архитектур информационных систем. Рассмотрим более подробно особенности вариантов построения информационных приложений.

**Таблица 1.** Типовые функциональные компоненты информационной системы

| **Обозначение** | **Наименование** | **Характеристика** |
|---|---|---|
| **PS** | Presentation Services (средства представления) | Обеспечиваются устройствами, принимающими ввод от пользователя и отображающими то, что сообщает ему компонент логики представления PL,с использованием соответствующей программной поддержки |
| PL | Presentation Logic (логика представления) | Управляет взаимодействием между пользователем и ЭВМ. Обрабатывает действия пользователя при выборе команды в меню, нажатии кнопки или выборе элемента из списка |
| BL | Business | Набор правил для принятия |

| | | |
|---|---|---|
| | or Application Logic (прикладная логика) | решений, вычислений и операций, которые должно выполнить приложение |
| DL | Data Logic (логика управления данными) | Операции с базой данных (SQL-операторы), которые нужно выполнить для реализации прикладной логики управления данными |
| DS | Data Services (операции с базой данных) | Действия СУБД, вызываемые для выполнения логики управления данными, такие как манипулирование данными, определения данных, фиксация или откат транзакций и т. п. СУБД обычно компилирует SQL-предложения |
| FS | File Services (файловые операции) | Дисковые операции чтения и записи данных для СУБД и других компонентов. Обычно являются функциями операционной системы (ОС) |

## Архитектуры компонент приложений информационных систем

### Архитектура файл-сервер

*Архитектура файл-сервер* не имеет сетевого разделения компонентов диалога PS и PL и использует компьютер для функций отображения, что облегчает построение графического интерфейса. Файл-сервер только извлекает данные из файлов, так что дополнительные пользователи и приложения добавляют лишь незначительную нагрузку на центральный процессор. Каждый новый клиент добавляет вычислительную мощность к сети.

Объектами разработки в файл-серверном приложении являются компоненты приложения, определяющие логику диалога PL, а также логику обработки BL и управления данными DL. Разработанное приложение реализуется либо в виде законченного загрузочного модуля, либо в виде специального кода для интерпретации. Однако такая архитектура имеет существенный недостаток: при выполнении некоторых запросов к базе данных клиенту могут передаваться большие объемы данных, загружая сеть и приводя к непредсказуемости времени реакции. Значительный сетевой трафик особенно сильно сказывается при организации удаленного доступа к базам данных на файл-сервере через низкоскоростные каналы связи. Одним из вариантов устранения данного недостатка является удаленное управление файл-серверным приложением в сети. При этом в локальной сети размещается сервер приложений, совмещенный с телекоммуникационным сервером (обычно называемым сервером доступа), в среде которого выполняются обычные файл-серверные приложения. Особенность состоит в том, что диалоговый ввод-вывод поступает от удаленных клиентов через телекоммуникации. Приложения не должны быть слишком сложными, иначе велика вероятность перегрузки сервера, или же нужна очень мощная платформа для сервера приложений. Одним из традиционных средств, на основе которых создаются файл-серверные системы, являются локальные СУБД. Однако такие системы, как правило, не отвечают требованиям обеспечения целостности данных (в частности, они не поддерживают транзакции). Поэтому при их использовании задача обеспечения целостности данных возлагается на программы клиентов, что приводит к усложнению клиентских приложений Однако эти инструменты привлекают своей простотой, удобством использования и доступностью. Поэтому файл-серверные информационные системы до сих пор представляют интерес для малых рабочих групп и, более того, нередко используются в качестве информационных систем масштаба предприятия.

*Архитектура клиент-сервер*

Архитектура клиент-сервер предназначена для разрешения проблем файл-серверных приложений путем разделения компонентов приложения и размещения их там, где они будут функционировать наиболее эффективно. Особенностью архитектуры клиент-сервер является использование выделенных серверов баз данных, понимающих запросы на языке структурированных запросов SQL (Structured Query Language) и выполняющих поиск, сортировку и агрегирование информации. Отличительная черта серверов БД - наличие справочника данных, в котором записана структура БД, ограничения целостности данных, форматы и даже серверные процедуры обработки[3] данных по вызову или по событиям в программе. Объектами разработки в таких приложениях помимо диалога и логики обработки являются, прежде всего, реляционная модель данных и связанный с ней набор SQL-операторов для типовых запросов к базе данных. Большинство конфигураций клиент-сервер использует двухуровневую модель, в которой клиент обращается к услугам сервера. Предполагается, что диалоговые компоненты PS и PL размещаются на клиенте, что позволяет обеспечить графический интерфейс. Компоненты управления данными DS и FS размещаются на сервере, а диалог (PS, PL), логика BL и DL - на клиенте. Двухуровневое определение архитектуры клиент-сервер использует именно этот вариант: приложение работает у клиента, СУБД - на сервере (рис. 3).
Поскольку эта схема предъявляет наименьшие требования к серверу, она обладает наилучшей масштабируемостью. Однако сложные приложения, вызывающие большое взаимодействие с БД, могут жестко загрузить как клиента, так и сеть. Результаты SQL-запроса должны вернуться клиенту для обработки, потому что там находится логика принятия решения. Такая схема приводит к дополнительному усложнению администрирования приложений, разбросанных по различным клиентским узлам.

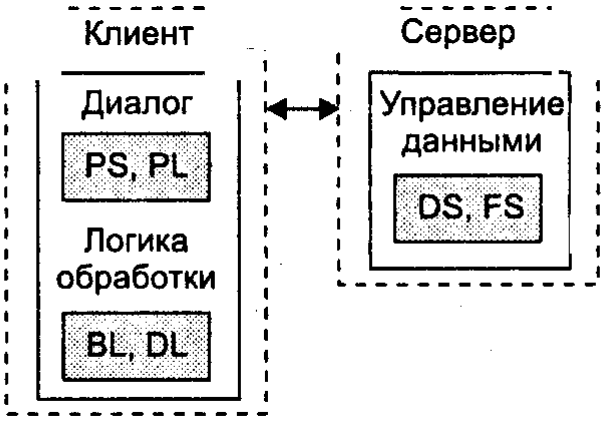


Рис. 3. Классический вариант клиент-серверной информационной системы

Для сокращения нагрузки на сеть и упрощения администрирования приложений компонент BL можно разместить на сервере. При этом вся логика принятия решений оформляется в виде *хранимых процедур* и выполняется на сервере БД. Хранимая процедура - процедура с операторами SQL для доступа к БД, вызываемая по имени с передачей требуемых параметров и выполняемая на сервере БД. Хранимые процедуры могут компилироваться, что повышает скорость их выполнения и сокращает нагрузку на сервер. Хранимые процедуры улучшают целостность приложений и БД, гарантируют актуальность коллективно используемых операций и вычислений. Улучшается сопровождение таких процедур, а также безопасность (нет прямого доступа к данным). Следует помнить, что перегрузка хранимых процедур прикладной логикой может перегрузить сервер, что приведет к потере производительности. Эта проблема особенно актуальна при разработке крупных информационных систем, в которых к серверу может одновременно обращаться большое количество клиентов. Поэтому в большинстве случаев следует принимать компромиссные решения: часть логики приложения размещать на стороне сервера, часть - на стороне клиента. Такие клиент-серверные системы называются системами с разделенной логикой. Данная схема при удачном разделении логики позволяет получить более сбалансированную загрузку клиентов и сервера, но при этом затрудняется сопровождение приложений.

Создание архитектуры клиент-сервер возможно и на основе многотерминальной системы. В этом случае в многозадачной среде сервера приложений выполняются программы пользователей, а клиентские узлы вырождены и представлены терминалами. Подобная схема информационной системы характерна для UNIX. В настоящее время архитектура клиент-сервер получила признание и широкое распространение как способ организации приложений для рабочих групп и информационных систем корпоративного уровня. Подобная организация работы повышает эффективность выполнения приложений за счет использования возможностей сервера БД, разгрузки сети и обеспечения контроля целостности данных.

Двухуровневые схемы архитектуры клиент-сервер могут привести к некоторым проблемам в сложных информационных приложениях с множеством пользователей и запутанной логикой. Решением этих проблем может стать использование многоуровневой архитектуры.

*Многоуровневая архитектура*

Многоуровневая архитектура стала развитием архитектуры клиент-сервер и в своей классической форме состоит из трех уровней:

-нижний уровень представляет собой приложения клиентов, выделенные для выполнения функций и логики представлений PS и PL и имеющие программный интерфейс для вызова приложения на среднем уровне;

- средний уровень представляет собой сервер приложений, на котором выполняется прикладная логика BL и с которого логика обработки данных DL вызывает операции с базой данных DS;

- верхний уровень представляет собой удаленный специализированный сервер базы данных, выделенный для услуг обработки данных DS и файловых операций FS (без риска использования хранимых процедур).

Подобную концепцию обработки данных используют, в частности, фирмы Oracle, Sun, Borland и др.

Трехуровневая архитектура позволяет еще больше сбалансировать нагрузку на разные узлы и сеть, а также способствует специализации инструментов для разработки приложений и устраняет недостатки двухуровневой модели клиент-сервер.

Централизация логики приложения упрощает администрирование и сопровождение. Четко разделяются платформы и инструменты для реализации интерфейса и прикладной логики, что позволяет с наибольшей отдачей реализовывать их специалистам узкого профиля. Наконец, изменения прикладной логики не затрагивают интерфейса, и наоборот. Но поскольку границы между компонентами PL, BL и DL размыты, прикладная логика может появиться на всех трех уровнях. Сервер приложений с помощью монитора транзакций обеспечивает интерфейс с клиентами и другими серверами, может управлять транзакциями и гарантировать целостность распределенной базы данных. Средства удаленного вызова процедур наиболее соответствуют идее распределенных вычислений: они обеспечивают из любого узла сети вызов прикладной процедуры, расположенной на другом узле, передачу параметров, удаленную обработку и возврат результатов.

С ростом систем клиент-сервер необходимость трех уровней становится все более очевидной. Продукты для трехзвенной архитектуры, так называемые мониторы транзакций, являются относительно новыми. Эти инструменты в основном ориентированы на среду UNIX, однако прикладные серверы можно строить на базе Microsoft Windows с использованием вызова удаленных процедур для организации связи клиентов с сервером приложений. На практике в локальной сети могут использоваться смешанные архитектуры (двухуровневые и трехуровневые) с одним и тем же сервером базы данных. С учетом глобальных связей архитектура может иметь больше трех звеньев. В настоящее время появились новые инструментальные средства для гибкой сегментации приложений клиент-сервер по различным узлам сети.

Таким образом, многоуровневая архитектура распределенных приложений позволяет повысить эффективность работы корпоративной информационной системы и оптимизировать распределение ее программно-аппаратных ресурсов. Слежует отметить, что пока на рынке программных продуктов по-прежнему доминирует архитектура клиент-сервер.

*Интернет/интранет-технологии*

В развитии технологии Интернет/интранет основной акцент пока что делается на разработке инструментальных программных средств. В то же время наблюдается отсутствие развитых средств разработки приложений, работающих с базами данных. Компромиссным решением для создания удобных и простых в использовании и сопровождении информационных систем, эффективно работающих с базами данных, стало объединение Интернет/интранет-технологии с многоуровневой архитектурой. При этом структура информационного приложения приобретает

следующий вид: браузер - сервер приложений - сервер баз данных - сервер динамических страниц - web-сервер.
Благодаря интеграции Интернет/интранет-технологий и архитектуры клиент-сервер процесс внедрения и сопровождения корпоративной информационной системы существенно упрощается при сохранении достаточно высокой эффективности и простоты совместного использования информации.

## Методология разработки информационных систем

Методология создания информационных систем заключается в организации процесса построения информационной системы и обеспечении управления этим процессом для того, чтобы гарантировать выполнение требований как к самой системе, так и к характеристикам процесса разработки. Основными задачами, решение которых должна обеспечивать методология создания корпоративных информационных систем (с помощью соответствующего набора инструментальных средств), являются следующие:
- обеспечение создания информационных систем, отвечающих целям и задачам предприятия и соответствующих предъявляемым к ним требованиям по автоматизации деловых процессов;
- гарантия создания системы с заданными параметрами в течение заданного времени в рамках оговоренного заранее бюджета;
- простота сопровождения, модификации и расширения системы с целью обеспечения ее соответствия изменяющимся условиям работы предприятия;
- обеспечение создания корпоративных информационных систем, отвечающих требованиям открытости, переносимости и масштабируемости;
- возможность использования в создаваемой системе разработанных ранее и применяемых на предприятии средств информационных технологий (программного обеспечения, баз данных, средств вычислительной техники, телекоммуникаций).
Методологии, технологии и инструментальные средства проектирования (CASE-средства) составляют основу проекта любой информационной системы. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов жизненного цикла информационных систем.
Основное содержание технологии проектирования составляют технологические инструкции, состоящие из описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция, и описаний самих операций.
Технология проектирования может быть представлена как совокупность трех составляющих:
- заданной последовательности выполнения технологических операций проектирования;
- критериев и правил, используемых для оценки результатов выполнения технологических операций;
- графических и текстовых средств (нотаций), используемых для описания проектируемой системы.
Каждая технологическая операция должна обеспечиваться следующими материальными и информационными ресурсами:
-данными, полученными на предыдущей операции (или исходными данными), представленными в стандартном виде;
-методическими материалами, инструкциями, нормативами и стандартами; Q программными и техническими средствами;
- исполнителями.
Результаты выполнения операции должны представляться в некотором стандартном виде, обеспечивающем их адекватное восприятие при выполнении следующей технологической операции (на которой они будут использоваться в качестве исходных данных).
Можно сформулировать следующий ряд общих требований, которым должна удовлетворять технология проектирования, разработки и сопровождения информационных систем:
-поддерживать полный жизненный цикл информационной системы;
-обеспечивать гарантированное достижение целей разработки системы с заданным качеством и в установленное время;
- обеспечивать возможность разделения крупных проектов на ряд подсистем - декомпозицию проекта на составные части, разрабатываемые группами исполнителей ограниченной численности, с последующей интеграцией составных частей;
- технология должна обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами (3-7 человек). Это обусловлено принципами управляемости коллектива и повышения производительности за счет минимизации числа внешних связей;
- обеспечивать минимальное время получения работоспособной системы;

- предусматривать возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта;
- обеспечивать независимость выполняемых проектных решений от средств реализации системы - системы управления базами данных, операционной системы, языка и системы программирования.

*Методология RAD ( Rapid Application Development)*

На начальном этапе существования компьютерных информационных систем их разработка велась на традиционных языках программирования. Однако по мере возрастания сложности разрабатываемых систем и увеличения запросов пользователей (чему в значительной степени способствовал прогресс в области вычислительной техники, а также появление удобного графического интерфейса пользователя в системном программном обеспечении) потребовались новые средства, обеспечивающие значительное сокращение сроков разработки. Это послужило предпосылкой к созданию целого направления в области программного обеспечения — инструментальных средств для быстрой разработки приложений. Развитие этого направления привело к появлению на рынке программного обеспечения средств автоматизации практически всех этапов жизненного цикла информационных систем.

*Основные особенности методологии RAD*

Методология разработки информационных систем, основанная на использовании средств быстрой разработки приложений, получила в последнее время широкое распространение и приобрела название *методологии быстрой разработки приложений* — RAD (Rapid Application Development). Данная методология охватывает все этапы жизненного цикла современных информационных систем.

RAD - это комплекс специальных инструментальных средств быстрой разработки прикладных информационных систем, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений.

Под методологией быстрой разработки приложений обычно понимается процесс разработки информационных систем, основанный на трех основных элементах:
- небольшой команде программистов (обычно от 2 до 10 человек);
- тщательно проработанный производственный график работ, рассчитанный на сравнительно короткий срок разработки (от 2 до 6 мес.);
- итерационная модель разработки, основанная на тесном взаимодействии с заказчиком - по мере выполнения проекта разработчики уточняют и реализуют в продукте требования, выдвигаемые заказчиком.

При использовании методологии RAD большое значение имеют опыт и профессионализм разработчиков. Группа разработчиков должна состоять из профессионалов, имеющих опыт в анализе, проектировании, программировании и тестировании программного обеспечения.

Основные принципы методологии RAD можно свести к следующему:
- используется итерационная (спиральная) модель разработки; G полное завершение работ на каждом из этапов жизненного цикла не обязательно;
- в процессе разработки информационной системы необходимо тесное взаимодействие с заказчиком и будущими пользователями;
- необходимо применение CASE-средств и средств быстрой разработки приложений;
- необходимо применение средств управления конфигурацией, облегчающих
внесение изменений в проект и сопровождение готовой системы;
- необходимо использование прототипов, позволяющее полнее выяснить и реализовать потребности конечного пользователя;
- тестирование и развитие проекта осуществляются одновременно с разработкой;
- разработка ведется немногочисленной и хорошо управляемой командой профессионалов;
- необходимы грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

## Методология разработки информационных систем. Объектно-ориентированный подход.

*Объектно-ориентированный подход*

Средства RAD дали возможность реализовывать совершенно иную по сравнению с традиционной технологию создания приложений: информационные объекты формируются как некие действующие модели (прототипы), чье функционирование согласовывается с

пользователем, а затем разработчик может переходить непосредственно к формированию законченных приложений, не теряя из виду общей картины проектируемой системы.

Возможность использования подобного подхода в значительной степени является результатом применения принципов объектно-ориентированного проектирования. Применение объектно-ориентированных методов позволяет преодолеть одну из главных трудностей, возникающих при разработке сложных систем - колоссальный разрыв между реальным миром (предметной областью описываемой проблемы) и имитирующей средой.

Использование объектно-ориентированных методов позволяет создать описание (модель) предметной области в виде совокупности объектов - сущностей, объединяющих данные и методы обработки этих данных (процедуры). Каждый объект обладает своим собственным поведением и моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует определенное поведение.

В объектном подходе акцент переносится на конкретные характеристики физической или абстрактной системы, являющейся предметом программного моделирования. Объекты обладают целостностью, которая не может быть нарушена. Таким образом, свойства, характеризующие объект и его поведение, остаются неизменными. Объект может только менять состояние, управляться или становиться в определенное отношение к другим объектам.

Широкую известность объектно-ориентированное программирование получило с появлением визуальных средств проектирования, когда было обеспечено слияние (инкапсуляция) данных с процедурами, описывающими поведение реальных объектов, в объекты программ, которые могут быть отображены определенным образом в графической пользовательской среде. Это позволило приступить к созданию программных систем, максимально йохожих на реальные, и добиваться наивысшего уровня абстракции. В свою очередь, объектно-ориентированное программирование позволяет создавать более надежные коды, так как у объектов программ существует точно определенный и жестко контролируемый интерфейс.

При разработке приложений с помощью инструментов RAD используется множество готовых объектов, сохраняемых в общедоступном хранилище. Однако обеспечивается и возможность разработки новых объектов. При этом новые объекты могут разрабатываться как на основе существующих, так и «с нуля».

Инструментальные средства RAD обладают удобным графическим интерфейсом пользователя и позволяют на основе стандартных объектов формировать простые приложения без написания кода программы. Это является большим преимуществом RAD, так как в значительной степени сокращает рутинную работу по разработке интерфейсов пользователя (при использовании обычных средств разработка интерфейсов представляет собой достаточно трудоемкую задачу, отнимающую много времени). Высокая скорость разработки интерфейсной части приложений позволяет быстро создавать прототипы и упрощает взаимодействие с конечными пользователями.

Таким образом, инструменты RAD позволяют разработчикам сконцентрировать усилия на сущности реальных деловых процессов предприятия, для которого создается информационная система. В итоге это приводит к повышению качества разрабатываемой системы.

*Визуальное программирование*

Применение принципов объектно-ориентированного программирования позволило создать принципиально новые средства проектирования приложений, называемые средствами *визуального программирования.* Визуальные инструменты RAD позволяют создавать сложные графические интерфейсы пользователя вообще без написания кода программы. При этом разработчик может на любом этапе наблюдать то, что закладывается в основу принимаемых решений. Визуальные средства разработки оперируют в первую очередь со стандартными интерфейсными объектами - окнами, списками, текстами, которые легко можно связать с данными из базы данных и отобразить на экране монитора. Другая группа объектов представляет собой стандартные элементы управления - кнопки, переключатели, флажки, меню и т. п., с помощью которых осуществляется управление отображаемыми данными. Все эти объекты могут быть стандартным образом описаны средствами языка, а сами описания сохранены для дальнейшего повторного использования.

В настоящее время существует довольно много различных визуальных средств разработки приложений. Но все они могут быть разделены на две группы - универсальные и специализированные.

Среди универсальных систем визуального программирования сейчас наиболее распространены такие, как Borland Delphi и Visual Basic. Универсальными мы их называем потому, что они не ориентированы на разработку только приложений баз данных - с их помощью могут быть разработаны приложения почти любого типа, в том числе и информационные приложения.

Причем программы, разрабатываемые с помощью универсальных систем, могут взаимодействовать практически с любыми системами управления базами данных. Это обеспечивается как использованием драйверов ODBC или OLE DB, так и применением специализированных средств (компонентов).

Специализированные средства разработки ориентированы только на создание приложений баз данных. Причем, как правило, они привязаны к вполне определенным системам управления базами данных. В качестве примера таких систем можно привести Power Builder фирмы Sybase (естественно, предназначенный для работы с СУБД Sybase Anywhere Server) и Visual FoxPro фирмы Microsoft. Поскольку задачи создания прототипов и разработки пользовательского интерфейса, по существу, слились, программист получил непрерывную обратную связь с конечными пользователями, которые могут не только наблюдать за созданием приложения, но и активно участвовать в нем, корректировать результаты и свои требования. Это также способствует сокращению сроков разработки и является важным психологическим аспектом, который привлекает к RAD все большее число пользователей.

Визуальные инструменты RAD позволяют максимально сблизить этапы создания информационных систем: анализ исходных условий, проектирование системы, разработка прототипов и окончательное формирование приложений становятся сходными, так как на каждом этапе разработчики оперируют визуальными объектами.

*Событийное программирование*

Логика приложения, построенного с помощью RAD, является событийно-ориентированной. Это означает следующее: каждый объект, входящий в состав приложения, может генерировать события и реагировать на события, генерируемые другими объектами. Примерами событий могут быть: открытие и закрытие окон, нажатие кнопки, нажатие клавиши клавиатуры, движение мыщи, изменение данных в базе данных и т. п.

Разработчик реализует логику приложения путем определения обработчика каждого события - процедуры, выполняемой объектом при наступлении соответствующего события. Например, обработчик события «нажатие кнопки» может открыть диалоговое окно. Таким образом, управление объектами осуществляется с помощью событий.

Обработчики событий, связанных с управлением базой данных (DELETE, INSERT, UPDATE), могут реализовываться в виде триггеров на клиентском или серверном узле. Такие обработчики позволяют обеспечить ссылочную целостность базы данных при операциях удаления, вставки и обновления, а также автоматическую генерацию первичных ключей.

## Методология разработки информационных систем. Этап проектирования.

Рассмотрим более подробно этап проектирования приложений информационных систем на основе вышеупомянутой технологии RAD. На этапе проектирования необходимым инструментом являются CASE-средства, используемые для быстрого получения работающих прототипов приложений.

Термин CASE (Computer Aided Software/System Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE ограничивалось лишь вопросами автоматизации разработки программного обеспечения. Однако в дальнейшем значение этого термина расширилось и приобрело новый смысл, охватывающий процесс разработки сложных информационных систем в целом. Теперь под термином «CASE-средства» понимаются программные средства, поддерживающие процессы создания и сопровождения информационных систем, включая анализ и формулировку требований, проектирование прикладного программного обеспечения и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. Прототипы, созданные с помощью CASE-средств, анализируются пользователями, которые уточняют и дополняют те требования к системе, которые не были выявлены на предыдущей фазе. Таким образом, на данной фазе также необходимо участие будущих пользователей в техническом проектировании системы.

Для построения всех моделей и прототипов должны быть использованы именно те CASE-средства, которые будут затем применяться при построении системы. Данное требование связано с тем, что при передаче информации о проекте с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог или отчет (это позволяет устранить неясности или неоднозначности). Затем определяются требования разграничения доступа к данным.
После детального рассмотрения процессов определяется количество функциональных элементов разрабатываемой системы. Это позволяет разделить информационную систему на ряд подсистем, каждая из которых реализуется одной командой разработчиков за приемлемое для RAD-проектов время (порядка полутора месяцев). С использованием CASE-средств проект распределяется между различными командами - делится функциональная модель.
На этой же фазе происходит определение набора необходимой документации. Результатами данной фазы являются:
- общая информационная модель системы;
- функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
- точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, диалогов и отчетов.
Одной из особенностей применения методологии RAD на данной фазе является то, что каждый созданный прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация. (При традиционном подходе использовались средства прототипирования, не предназначенные для построения реальных приложений, поэтому разработанные прототипы не могли быть использованы на последующих фазах и просто «выбрасывались» после того, как выполняли задачу устранения неясностей в проекте.)
*Фаза построения*
На фазе построения выполняется собственно быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных ранее моделей, а также требований нефункционального характера. Разработка приложения ведется с использованием визуальных средств программирования. Формирование программного кода частично выполняется с помощью автоматических генераторов кода, входящих в состав CASE-средств. Код генерируется на основе разработанных моделей. На фазе построения также требуется участие пользователей системы, которые оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом.
Завершается физическое проектирование системы, а именно:
- определяется необходимость распределения данных;
- производится анализ использования данных;
- производится физическое проектирование базы данных;
- определяются требования к аппаратным ресурсам;
- определяются способы увеличения производительности;
- завершается разработка документации проекта.
Результатом данной фазы является готовая информационная система, удовлетворяющая всем требованиям пользователей.
*Фаза внедрения*
Фаза внедрения в основном сводится к обучению пользователей разработанной информационной системы.
Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, еще на этапе проектирования системы.
Приведенная схема разработки информационной системы не является универсальной. Вполне возможны различные отклонения от нее. Это связано с зависимостью схемы выполнения проекта от начальных условий, при которых начинается разработка (например, разрабатывается совершенно новая система или на предприятии уже существует некоторая информационная система). Во втором случае существующая система может либо использоваться в качестве прототипа новой системы, либо интегрироваться в новую разработку в качестве одной из подсистем.
Несмотря на все свои достоинства, методология RAD тем не менее (как, впрочем, и любая другая методология) не может претендовать на универсальность. Ее применение наиболее

эффективно при выполнении сравнительно небольших систем, разрабатываемых для вполне определенного предприятия.
При разработке же типовых систем, не являющихся законченным продуктом, а представляющих собой совокупность типовых элементов информационной системы, большое значение имеют такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Это связано с тем, что типовые системы обычно централизованно сопровождаются и могут быть адаптированы к различным программно-аппаратным платформам, системам управления базами данных, коммуникационным средствам, а также интегрироваться с существующими разработками. Поэтому для такого рода проектов необходим высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

## Методология разработки информационных систем. Стандарты и методики.

Одним из важных условий эффективного использования информационных технологий является внедрение корпоративных стандартов. Корпоративные стандарты представляет собой соглашение о единых правилах организации технологии или управления. При этом за основу корпоративных могут приниматься отраслевые, национальные и даже международные стандарты.
Однако высокая динамика развития информационных технологий приводит к быстрому устареванию существующих стандартов и методик разработки информационных систем. Так, например, в связи со значительным прогрессом в области программного обеспечения и средств вычислительной техники наблюдается рост размеров и сложности информационных систем. При этом существенно меняются требования как к основным функциям и сервисным возможностям систем, так и к динамике изменения этих функций. В этих условиях применение классических способов разработки и обеспечения качества информационных систем становиться малоэффективным и не приводит к уровню качества, адекватному реальным требованиям.
Полезны в этом отношении стандарты открытых систем (в первую очередь стандарты на интерфейсы различных видов, включая лингвистические, и на протоколы взаимодействия). Однако разработка систем в новых условиях требует также новых методов проектирования и новой организации проектных работ. Проектирование и методическая поддержка организации разработки информационных систем (включая программное обеспечение (ПО), и базы данных (БД)) традиционно поддерживаются многими стандартами и фирменными методиками. Вместе с тем известно, что требуется адаптивное планирование разработки, в том числе в динамике процесса ее выполнения. Одним из способов адаптивного проектирования является разработка и применение профилей жизненного цикла информационных систем и программного обеспечения. Корпоративные стандарты образуют целостную систему, которая включает три вида стандартов:
Стандарты и методики
- стандарты на продукты и услуги;
- стандарты на процессы и технологии;
- стандарты на формы коллективной деятельности, или управленческие стандарты.

*Виды стандартов*

Существующие на сегодняшний день стандарты можно несколько условно разделить на несколько групп по следующим признакам:
- *по предмету стандартизации.* К этой группе можно отнести функциональные стандарты (стандарты на языки программирования, интерфейсы, протоколы) и стандарты на организацию жизненного цикла создания и использования информационных систем и программного обеспечения;
- *по утверждающей организации.* Здесь можно выделить официальные международные, официальные национальные или национальные ведомственные стандарты (например, ГОСТы, ANSI, IDEF0/1), стандарты международных консорциумов и комитетов по стандартизации (например, консорциума OMG), стандарты «де-факто» - официально никем не утвержденные, но фактически действующие (например, стандартом «де-факто» долгое время были язык взаимодействия с реляционными базами данных SQL и язык программирования C), фирменные стандарты (например, Microsoft ODBC);

- *по методическому источнику.* К этой группе относятся различного рода методические материалы ведущих фирм-разработчиков программного обеспечения, фирм-консультантов, научных центров, консорциумов по стандартизации.

Необходимо иметь в виду, что, хотя это и не очевидно, в каждую из указанных выше групп и подгрупп входят стандарты, существенно различающиеся по степени обязательности для различных организаций; конкретности и детализации содержащихся требований; открытости и гибкости, а также адаптируемости к конкретным условиям. Рассмотрим в качестве примера методику Oracle CDM (Custom Development Method) по разработке прикладных информационных систем. Одним из сложившихся направлений деятельности фирмы Oracle стала разработка методологических основ и производство инструментальных средств для автоматизации процессов разработки сложных прикладных систем, ориентированных на интенсивное использование баз данных. Методика Oracle CDM является развитием давно разработанной версии Oracle CASE-Method, применяемой в CASE-средстве Oracle CASE.

Основу CASE-технологии и инструментальной среды фирмы ORACLE составляют:

- методология структурного нисходящего проектирования, при которой разработка прикладной системы представляется в виде последовательности четко определенных этапов;
- поддержка всех этапов жизненного цикла прикладной системы, начиная с самых общих описаний предметной области до получения и сопровождения готового программного продукта;
- ориентация на реализацию приложений в архитектуре клиент-сервер с использованием всех особенностей современных серверов баз данных, включая декларативные ограничения целостности, хранимые процедуры, триггеры баз данных, и с поддержкой в клиентской части всех современных стандартов и требований к графическому интерфейсу конечного пользователя;
- наличие централизованной базы данных, репозитария, для хранения спецификаций проекта прикладной системы на всех этапах ее разработки. Такой репозитарий представляет собой базу данных специальной структуры, работающую под управлением СУБД ORACLE;
- возможность одновременной работы с репозитарием многих пользователей. Такой многопользовательский режим почти автоматически обеспечивается стандартными средствами СУБД ORACLE. Централизованное хранение проекта системы и управление одновременным доступом к нему всех участников разработки поддерживают согласованность действий разработчиков и не допускают ситуацию, когда каждый проектировщик или программист работает со своей версией проекта и модифицирует ее независимо от других;
- автоматизация последовательного перехода от одного этапа разработки к следующему. Для этого предусмотрены специальные утилиты, с помощью которых можно по спецификациям концептуального уровня (модели предметной области) автоматически получать первоначальный вариант спецификации уровня проектирования (описание структуры базы данных и состава программных модулей), чтобы на его основе после всех необходимых уточнений и дополнений автоматически генерировать готовые к выполнению программы;
- автоматизация различных стандартных действий по проектированию и реализации приложения: предусматривается генерация многочисленных отчетов по содержимому репозитария, обеспечивающих полное документирование текущей версии системы на всех этапах ее разработки; с помощью специальных процедур предоставляется возможность проверки спецификаций на полноту и непротиворечивость.

Жизненный цикл информационной системы формируется из определенных этапов (фаз) проекта и процессов, каждый из которых выполняется в течение нескольких этапов.

Методика Oracle CDM определяет следующие фазы жизненного цикла информационной системы:

- стратегия;
- анализ (формулирование детальных требований к прикладной системе);
- проектирование (преобразование требований в детальные спецификации системы);
- реализация (написание и тестирование приложений);
- внедрение (установка новой прикладной системы, подготовка к началу эксплуатации);
- эксплуатация (поддержка приложения и слежение за ним, планирование будущих функциональных расширений).

Первый этап связан с моделированием и анализом процессов, описывающих деятельность организации, технологические особенности работы. Целью является построение моделей существующих процессов, выявление их недостатков и возможных источников усовершенствования. Этот этап не является обязательным в случае, когда существующая технология и организационные структуры четко определены, хорошо понятны и не требуют дополнительного изучения и реорганизации.

На втором этапе разрабатываются детальные концептуальные модели предметной области, описывающие информационные потребности организации, особенности функционирования и т. п. Результатом являются модели двух типов:
- информационные, отражающие структуру и общие закономерности предметной области;
- функциональные, описывающие особенности решаемых задач.

На третьей стадии (этапе проектирования) на основании концептуальных моделей вырабатываются технические спецификации будущей прикладной системы - определяются структура и состав базы данных, специфицируется набор программных модулей. Первоначальный вариант проектных спецификаций может быть получен автоматически с помощью специальных утилит на основании данных концептуальных моделей.

Использование генераторов приложений, позволяет полностью автоматизировать этот этап, существенно сократить сроки разработки системы и повысить ее качество и надежность.

Методика Oracle CDM выделяет следующие процессы, протекающие на протяжении жизненного цикла информационной системы:
- определение производственных требований;
- исследование существующих систем;
- определение технической архитектуры;
- проектирование и построение базы данных;
- проектирование и реализация модулей;
- конвертирование данных;
- документирование;
- тестирование;
- обучение;
- переход к новой системе;
- поддержка и сопровождение.

Процессы состоят из последовательностей задач, задачи разных процессов взаимосвязаны с помощью явных ссылок.

Отметим основные особенности методики Oracle CDM, определяющие область ее применения и присущие ей ограничения.

1 .Степень адаптивности CDM ограничивается тремя моделями жизненного цикла:
- классическая - предусматривает все этапы;
- быстрая разработка - ориентированна на использование инструментов моделирования и программирования Oracle;
- облегченный подход - рекомендуется в случае малых проектов и возможности быстро прототипировать приложения.

2. Методика не предусматривает включение дополнительных задач, которые не оговорены в CDM, и их привязку к остальным. Также исключено удаление задачи (и порождаемых ею документов), не предусмотренное ни одной из трех моделей жизненного цикла, и изменение последовательности выполнения задач по сравнению с предложенной.

3. Все модели жизненного цикла являются по сути каскадными. Даже «облегченный подход», несмотря на итерационность выполнения действий по прототипированию, сохраняет общий последовательный и детерминированный порядок выполнения задач.

4. Методика не является обязательной, но может считаться фирменным стандартом. При формальном применении степень обязательности полностью соответствует ограничениям возможностей адаптации.

5. Прикладная система рассматривается в основном как программно-техническая система — например, возможность выполнения организационно-структурных преобразований, практически всегда происходящих при переходе к новой информационной системе, в этой методике отсутствуют.

6. CDM теснейшим образом опирается на использование инструментария Oracle, несмотря на утверждения о простом приспособлении CDM к проектам, в которых используется другой комплект инструментальных средств.

7. Методика Oracle CDM представляет собой вполне конкретный материал, детализированный до уровня заготовок проектных документов, рассчитанных на прямое использование в проектах информационных систем с опорой на инструментальные средства и СУБД фирмы Oracle.

# Методология разработки информационных систем. Профили открытых информационных систем.

Создание, сопровождение и развитие современных сложных информационных систем базируется на методологии построения таких систем как *открытых.* Открытые информационные системы создаются в процессе информатизации всех основных сфер современного общества: органов государственного управления, финансово-кредитной сферы, информационного обслуживания предпринимательской деятельности, производственной сферы, науки, образования. Развитие и использование открытых информационных систем неразрывно связаны с применением стандартов на основе методологии функциональной стандартизации информационных технологий.

*Понятие профиля информационной системы*

При создании и развитии сложных, распределенных, тиражируемых информационных систем требуется гибкое формирование и применение гармонизированных совокупностей базовых стандартов и нормативных документов разного уровня, выделение в них требований и рекомендаций, необходимых для реализации заданных функций системы. Для унификации и регламентирования такие совокупности базовых стандартов должны адаптироваться и конкретизироваться применительно к определенным классам проектов, функций, процессов и компонентов системы. В связи с этим выделилось и сформировалось понятие *профиля* информационной системы как основного инструмента функциональной стандартизации.

*Профиль* - это совокупность нескольких (или подмножество одного) базовых стандартов с четко определенными и гармонизированными подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций.

Профиль формируется исходя из функциональных характеристик объекта стандартизации. В профиле выделяются и устанавливаются допустимые возможности и значения параметров каждого базового стандарта и/или нормативного документа, входящего в профиль.

Профиль не должен противоречить использованным в нем базовым стандартам и нормативным документам. Он должен применять выбранные из альтернативных вариантов необязательные возможности и значения параметров в пределах допустимых.

На базе одной совокупности базовых стандартов могут формироваться и утверждаться различные профили для разных проектов информационных систем. Ограничения базовых документов профиля и их согласованность, проведенная разработчиками профиля, должны обеспечивать качество, совместимость и корректное взаимодействие отдельных компонентов системы, соответствующих профилю, в заданной области его применения.

Базовые стандарты и профили в зависимости от проблемно-ориентированной области применения информационных систем могут использоваться как непосредственные директивные, руководящие или рекомендательные документы, а также как нормативная база, необходимая при выборе или разработке средств автоматизации технологических этапов или процессов создания, сопровождения и развития информационных систем.

Обычно рассматривают две группы профилей:

- регламентирующие архитектуру и структуру информационной системы;
- регламентирующие процессы проектирования, разработки, применения, сопровождения и развития системы.

В зависимости от области применения профили могут иметь разные категории и соответственно разные статусы утверждения:

- профили конкретной информационной системы, определяющие стандартизованные проектные решения в пределах данного проекта;
- профили информационной системы, предназначенные для решения некоторого класса прикладных задач.

Профили информационных систем унифицируют и регламентируют только часть требований, характеристик, показателей качества объектов и процессов, выделенных и формализованных на базе стандартов и нормативных документов. Другая часть функциональных и технических характеристик системы определяется заказчиками и разработчиками творчески, без учета положений нормативных документов.

*Принципы формирования профиля информационной системы*

Использование профилей информационных систем призвано решить следующие задачи:

- снижение трудоемкости проектов;
- повышение качества компонентов информационной системы;
- обеспечение расширяемости и масштабируемости разрабатываемых систем;
- обеспечение возможности функциональной интеграции в информационную

систему задач, которые раньше решались раздельно;
- обеспечение переносимости прикладного программного обеспечения.

В зависимости от того, какие из указанных задач являются наиболее приоритетными, производится выбор стандартов и документов для формирования профиля. Актуальность использования профилей информационных систем обусловлена современным состоянием стандартизации информационных технологий, которое характеризуется следующими особенностями:
- существует множество международных и национальных стандартов, которые не полностью и неравномерно удовлетворяют потребности в стандартизации объектов и процессов создания и применения сложных информационных систем;
- длительные сроки разработки, согласования и утверждения международных и национальных стандартов приводят к их консерватизму и хроническому отставанию от современных информационных технологий;
- функциональными стандартами поддержаны и регламентированы только самые простые объекты и рутинные, массовые процессы: телекоммуникации, программирование, документирование программ и данных. Наиболее сложные и творческие процессы создания и развития крупных распределенных информационных систем - системный анализ и проектирование, интеграция компонентов и систем, испытания и сертификация - почти не поддержаны требованиями и рекомендациями стандартов из-за трудности их формализации и унификации;
- совершенствование и согласование нормативных и методических документов в ряде случаев позволяют создать на их основе национальные и международные стандарты.

Подходы к формированию профилей информационных систем могут быть различными. В международной функциональной стандартизации информационных технологий принято довольно жесткое понятие профиля. Считается, что его основой могут быть только международные и национальные, утвержденные стандарты. Использование стандартов де-факто и нормативных документов фирм не допускается. При таком подходе затруднены унификация, регламентирование и параметризация множества конкретных функций и характеристик сложных объектов архитектуры и структуры современных информационных систем. Другой подход к разработке и применению профилей информационных систем состоит в использовании совокупности адаптированных и параметризованных базовых международных и национальных стандартов и открытых спецификаций, отвечающих стандартам де-факто и рекомендациям международных консорциумов.

Эталонная модель среды открытых систем (OSE/RM) определяет разделение любой информационной системы на две составляющие: *приложения* (прикладные программы и программные комплексы) и *среду,* в которой эти приложения функционируют.

Между приложениями и средой определяются стандартизованные интерфейсы - Application Program Interface (API), которые являются необходимой частью профилей любой открытой системы. Кроме того, в профилях могут быть определены унифицированные интерфейсы взаимодействия функциональных частей друг с другом и интерфейсы взаимодействия между компонентами среды системы. Спецификации выполняемых функций и интерфейсов взаимодействия могут быть оформлены в виде профилей компонентов системы. Таким образом, профили информационной системы с иерархической структурой могут включать в) себя:
- стандартизованные описания функций, выполняемых данной системой;
- функции взаимодействия системы с внешней для нее средой;
- стандартизованные интерфейсы между приложениями и средой информационной системы;
- профили отдельных функциональных компонентов, входящих в систему.

Для эффективного использования конкретного профиля необходимо:
- выделить объединенные логической связью проблемно-ориентированные области функционирования, где могут применяться стандарты, общие для одной организации или группы организаций;
- идентифицировать стандарты и нормативные документы, варианты их использования и параметры, которые необходимо включить в профиль;
- документально зафиксировать участки конкретного профиля, где требуется создание новых стандартов или нормативных документов, и идентифицировать характеристики, которые могут оказаться важными для разработки недостающих стандартов и нормативных документов этого профиля;
- формализовать профиль в соответствии с его категорией, включая стандарты, различные варианты нормативных документов и дополнительные параметры, которые непосредственно связаны с профилем;

- опубликовать профиль и/или продвигать его по формальным инстанциям для дальнейшего распространения.

При использовании профилей важное значение имеет обеспечение проверки корректности их применения путем тестирования, испытаний и сертификации. Для этого требуется создание технологии контроля и тестирования в процессе применения профиля. Данная технология должна поддерживаться совокупностью методик, инструментальных средств, составом и содержанием оформляемых документов на каждом этапе выполнения проекта.

Использование профилей способствует унификации при разработке тестов, проверяющих качество и взаимодействие компонентов проектируемой информационной системы. Профили должны определяться таким образом, чтобы тестирование их реализации можно было проводить по возможности наиболее полно по стандартизованной методике. При этом возможно применение ранее разработанных методик, так как международные стандарты и профили являются основой для создания общепризнанных аттестационных тестов.

*Структура профилей информационных систем*

Разработка и применение профилей являются органической частью процессов проектирования, разработки и сопровождения информационных систем. Профили характеризуют каждую конкретную информационную систему на всех стадиях ее жизненного цикла, задавая согласованный набор базовых стандартов, которым должна соответствовать система и ее компоненты.

Стандарты, важные с точки зрения заказчика, должны задаваться в ТЗ на проектирование системы и составлять ее первичный профиль. То, что не задано в ТЗ, первоначально остается на усмотрение разработчика системы, который, руководствуясь требованиями ТЗ, может дополнять и развивать профили системы и впоследствии согласовывать их с заказчиком. Таким образом, профиль конкретной системы не является статичным, он развивается и конкретизируется в процессе проектирования информационной системы и оформляется в составе документации проекта системы.

В профиль конкретной системы включаются спецификации компонентов, разработанных в составе данного проекта, и спецификации использованных готовых программных и аппаратных средств, если эти средства не специфицированы соответствующими стандартами. После завершения проектирования и испытаний системы, в ходе которых проверяется ее соответствие профилю, профиль применяется как основной инструмент сопровождения системы при эксплуатации, модернизации и развитии.

*Общая структура профиля информационной системы*

Формирование и применение профилей конкретных информационных систем выполняется на основе использования международных и национальных стандартов, ведомственных нормативных документов, а также стандартов де-факто при условии доступности соответствующих им спецификаций. Для обеспечения корректного применения профилей их описания должны содержать: - определение целей использования данного профиля;

- точное перечисление функций объекта или процесса стандартизации, определяемого данным профилем;
- формализованные сценарии применения базовых стандартов и спецификаций, включенных в данный профиль;
- сводку требований к информационной системе или ее компонентам, определяющих их соответствие профилю, и требований к методам тестирования соответствия;
- нормативные ссылки на конкретный набор стандартов и других нормативных документов, составляющих профиль, с точным указанием применяемых редакций и ограничений, способных повлиять на достижение корректного взаимодействия объектов стандартизации при использовании данного профиля;
- информационные ссылки на все исходные документы.

На стадиях жизненного цикла информационной системы выбираются и затем применяются основные функциональные профили:

- профиль прикладного программного обеспечения;
- профиль среды информационной системы;
- профиль защиты информации в информационной системе;
- профиль инструментальных средств, встроенных в информационную систему.

*Профиль прикладного программного обеспечения*

Прикладное программное обеспечение всегда является проблемно-ориентированным и определяет основные функции информационной системы. Функциональные профили системы должны включать в себя согласованные базовые стандарты. При использовании функциональных профилей информационных систем следует еще иметь в виду согласование

этих профилей между собой. Необходимость такого согласования возникает, в частности, при использовании стандартизованных API, в том числе интерфейсов приложений со средой их функционирования и со средствами защиты информации. При согласовании функциональных профилей возможны также уточнения профиля среды системы и профиля встраиваемых инструментальных средств создания, сопровождения и развития прикладного программного обеспечения.

*Профиль среды информационной системы*

Профиль среды информационной системы должен определять ее архитектуру в соответствии с выбранной моделью обработки данных.

Стандарты интерфейсов приложений со средой (API) должны быть определены по функциональным областям профилей информационной системы. Декомпозиция структуры среды функционирования системы на составные части, выполняемая на стадии эскизного проектирования, позволяет детализировать профиль среды информационной системы по функциональным областям эталонной модели OSE/RM:

- область графического пользовательского интерфейса;
- область реляционных или объектно-ориентированных СУБД (например, стандарт языка SQL-92 и спецификации доступа к разным базам данных);
- область операционных систем с учетом сетевых функций, выполняемых на уровне операционной системы;
- область телекоммуникационной среды в части услуг и служб прикладного уровня: электронной почты, доступа к удаленным базам данных, передачи файлов, доступа к файлам и управления файлами.

Профиль среды распределенной системы должен включать стандарты протоколов транспортного уровня, стандарты локальных сетей (например, стандарт Ethernet IEEE 802.3 или стандарт Fast Ethernet IEEE 802.3 и), а также стандарты средств сопряжения проектируемой информационной системы с сетями передачи данных общего назначения.

Выбор аппаратных платформ информационной системы связан с определением их параметров: вычислительной мощности серверов и рабочих станций в соответствии с проектными решениями по разделению функций между клиентами и серверами; степени масштабируемости аппаратных платформ; надежности. Профиль среды должен содержать стандарты, определяющие параметры технических средств и способы их измерения (например, стандартные тесты измерения производительности).

*Профиль защиты информации*

Профиль защиты информации должен обеспечивать реализацию политики информационной безопасности, разрабатываемой в соответствии с требуемой категорией безопасности и критериями безопасности, заданными в ТЗ на систему. Построение профиля защиты информации в распределенных системах клиент-сервер методически связано с точным определением компонентов системы, ответственных за те или иные функции, службы и услуги, и средств защиты информации, встроенных в эти компоненты. Функциональная область защиты информации включает в себя следующие функции защиты, реализуемые разными компонентами системы:

- функции, реализуемые операционной системой;
- функции защиты от несанкционированного доступа, реализуемые на уровне программного обеспечения промежуточного слоя;
- функции управления данными, реализуемые СУБД;
- функции защиты программных средств, включая средства защиты от вирусов;
- функции защиты информации при обмене данными в распределенных системах, включая криптографические функции;
- функции администрирования средств безопасности.

Профиль защиты информации должен включать указания на методы и средства обнаружения в применяемых аппаратных и программных средствах недекларированных возможностей. Профиль должен также включать указания на методы и средства резервного копирования информации и восстановления информации при отказах и сбоях аппаратуры системы.

*Профиль инструментальных средств*

Профиль инструментальных средств, встроенных в информационную систему, должен отражать решения по выбору методологии и технологии создания, сопровождения и развития информационной системы. В этом профиле должны содержаться ссылки на описание выбранных методологии и технологии, выполненное на стадии эскизного проектирования системы.

Состав инструментальных средств определяется на основании решений и нормативных документов об организации сопровождения и развития информационной системы. При этом должны быть учтены правила и порядок, регламентирующие внесение изменений в действующие системы. Функциональная область профиля
инструментальных средств, встроенных в систему, охватывает функции централизованного управления и администрирования, связанные с:
- контролем производительности и корректности функционирования системы в целом;
- управлением конфигурацией прикладного программного обеспечения, тиражированием версий;
- управлением доступом пользователей к ресурсам системы и конфигурацией ресурсов;
- перенастройкой приложений в связи с изменениями прикладных функций информационной системы;
- настройкой пользовательских интерфейсов (генерацией экранных форм и отчетов);
- ведением баз данных системы;
- восстановлением работоспособности системы после сбоев и аварий.
Дополнительные ресурсы, необходимые для функционирования встроенных инструментальных средств, такие как минимальный и рекомендуемый объем оперативной памяти, размеры требуемого дискового пространства и т. п., должны быть учтены в разделе проекта, относящемся к среде информационной системы. Выбор инструментальных средств, встроенных в систему, должен производиться в соответствии с требованиями профиля среды. Ссылки на соответствующие стандарты, входящие в профиль среды, должны содержаться и в профиле инструментальных средств.
В этом профиле должны также содержаться ссылки на требования к средствам тестирования, которые необходимы для процессов сопровождения и развития системы и должны быть в нее встроены. В число встроенных в информационную систему средств тестирования должны входить средства функционального тестирования приложений, тестирования интерфейсов, системного тестирования и тестирования серверов/клиентов при максимальной нагрузке.

## Разработка приложений информационных систем. Клиент-серверные архитектуры.

Развитие архитектуры "клиент-сервер" привело к появлению трехуровневой архитектуры, в которой кроме сервера и приложений-клиентов (клиентов) дополнительно присутствует сервер приложений. *Сервер приложений* является промежуточным уровнем, обеспечивающим организацию взаимодействия ("тонких" клиентов) и сервера, например, выполнение соединения с сервером, разграничение доступа к данным и реализацию бизнес-правил. Сервер приложений реализует работу с клиентами, расположенными на различных платформах, т. е. функционирующими на компьютерах различных типов и под управлением различных операционных систем. Сервер приложений также называют *брокером данных* (broker — посредник).
Основные достоинства трехуровневой архитектуры "клиент-сервер":
- снижение нагрузки на сервер;
- упрощение клиентских приложений;
- единое поведение всех клиентов;
- упрощение настройки клиентов;
- независимость от платформы.
Информационные системы, основанные на трехуровневой сетевой архитектуре называют также *распределенными.*
*Принципы построения трехуровневых приложений*
При создании многоуровневых приложений могут быть использованы различные технологии межпрограммного и межкомпьютерного заимодействия, перечислим наиболее часто применяемые:
- Модель DCOM (Distributed Component Object Model - модель распределенных компонентных объектов) позволяет использовать объекты, расположенные на другом компьютере.
- Сервер MTS (Microsoft Transaction Server - сервер транзакций Microsoft) является дополнением к технологии COM, разработанной фирмой Microsoft, и предназначен для управления транзакциями.

- Модель COM+ (Component Object Model+ - усовершенствованная объектная модель компонентов) фирмы Microsoft введена в Windows 2000 и интегрирует технологии MTS в стандартные службы COM.
- Сокеты TCP/IP (Transport Control Protocol/Protocol Internet - транспортный протокол/протокол Интернета) используются для соединения компьютеров в различных сетях, в том числе в Интернете.
- CORBA (Common Object Request Broker Architecture - общедоступная архитектура с брокером при запросе объекта) позволяет организовать взаимодействие между объектами, расположенными на различных платформах.
- SOAP (Simple Object Acces Protocol - простой протокол доступа к объектам) служит универсальным средством обеспечения взаимодействия с клиентами и серверами Web-служб на основе кодирования XML и передачи данных по протоколу HTTP.

При создании трехуровневого приложения разработка БД и использование сервера принципиально не отличаются от уже рассмотренного случая двухуровневых приложений. Главные особенности трехуровневого приложения связаны с созданием сервера приложений и клиентского приложения, а также с организацией взаимодействия между ними.
Соответствующая трехуровневая архитектура схематично представлена на рис.4. В частном случае два или все три уровня Могут располагаться на одном компьютере, что широко используется при отладке приложений.
Взаимодействие между сервером приложений и клиентом организуется через интерфейс провайдера, называемый *интерфейсом оператора* или просто *провайдером.* Этот интерфейс обеспечивает передачу информации в виде *пакетов данных.* Физически пакеты данных представляют собой совокупности двоичных кодов, образующих блоки. Логически пакет данных является подмножеством набора данных, которое содержит данные записей, а также метаданные (информация об именах и типах полей) и ограничения. Провайдер обеспечивает разбиение данных на пакеты, а также кодирование пакетов в зависимости от используемого сетевого протокола. Данные, отредактированные клиентом, пересылаются обратно, в пакете содержится информация о старых и новых значениях записей. Перед внесением в записи БД требуемых изменений (т. е. перед пересылкой записей серверу БД) сервер приложений выполняет проверку корректности и допустимости изменений. Изменения могут быть отвергнуты, например, если запись уже была изменена другим пользователем. В этом случае сервер приложений выполняет функции распознавания и обработки конфликтов между клиентами. Сервер приложений создается на основе удаленного модуля данных, который служит для размещения компонентов, а также для обеспечения взаимодействия с сервером и клиентами.

*Проектирование приложений информационных систем на основе использования WEB-технологий*

Рассмотрим особенности проектирования приложений информационных систем на основе использования WEB-технологий, в этой технологии, как правило, используется рассмотренная выше трехуровневая архитектура. Отметим, что размещение информации из в Интернете, и в частности широкое использование различного рода информационных систем с использованием СУБД, представляет собой новую информационную технологию, получившую широкое распространение в связи с ростом популярности и доступности "всемирной паутины". При использовании WEB-технологий решаются следующие типичные задачи, встающие перед разработчиками современного программного обеспечения:

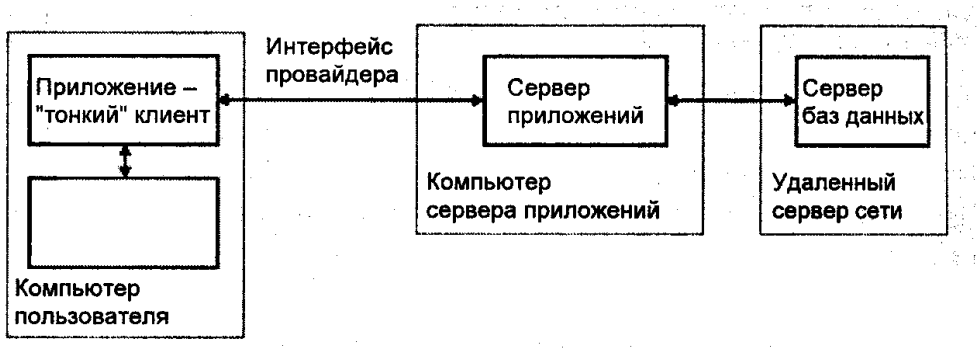


Рис.4 Трехуровневая архитектура типа клиент-сервер.

- организация взаимосвязи СУБД, работающих на различных платформах;
- построение информационных систем в Интернете на основе многоуровневой архитектуры БД (архитектура таких систем включает дополнительный уровень - Web-сервер с модулями расширения серверной части, который и реализует возможность информационного обмена с БД в глобальной сети);
- построение локальных интранет-сетей на основе технологии использования БД в Интернете (локальные сети строятся на принципах Интернету с выходом при необходимости в глобальную сеть);
- использование в Интернете информации из существующих локальных сетевых баз данных (при необходимости опубликования в глобальной сети информации из локальных сетей);
- применение БД для упорядочивания (каталогизирования) информации (огромный объем данных, представленных в Интернете, не обладает требуемой степенью структурированности, что делает процесс поиска необходимой информации весьма сложным и долгим);
- применение языка SQL для поиска необходимой информации в БД;
- использование средств СУБД для обеспечения безопасности данных, разграничения доступа и управления транзакциями при создании интернет-магазинов, защищенных информационных систем и т. д.;
- стандартизация пользовательского интерфейса на основе применения Web-обозревателей с типовым внешним видом пользовательского интерфейса и его типовой реакцией на действия пользователя;
- использование Web-обозревателя в качестве дешевой клиентской программы для доступа к БД.

Рассмотрим подробнее интерфейсы доступа к источникам данных применяемых при проектировании приложений информационных систем функционирующих в WEB - это:

*Интерфейс ODBC* (Open Database Connectivity - совместимость открытых баз данных) применяется операционной системой для доступа к источникам данных, как правило, к реляционным БД, использующим язык структурированных запросов SQL для организации управления данными.

*Интерфейс OLE DB* (Object Linking and Embedding DataBase - связывание и внедрение объектов баз данных) является более универсальной технологией для доступа к любым источникам данных через стандартный интерфейс COM (Component Object Model — объектная модель компонентов). Данные могут быть представлены в любом виде и формате (например, реляционные БД, электронные таблицы, документы в RTF-формате и т. д.). В интерфейсе OLE DB используется механизм провайдеров, под которыми понимаются поставщики данных, составляющих надстройку над физическим форматом данных. Такие провайдеры еще называются сервис-провайдерами, благодаря им можно объединять в однотипную совокупность объекты, связанные с разными источниками данных.

Кроме того, есть OLE DB-провайдер, который реализует интерфейс доступа OLE DB поверх конкретного сервис-провайдера данных. При этом поддерживается возможность многоуровневой системы OLE DB-провайдеров, когда OLE DB-провайдер может находиться поверх группы OLE DB-провайдеров или сервис-провайдеров.

Интерфейс OLE DB может использовать для доступа к источникам данных интерфейс ODBC. В этом случае применяется OLE DB-провайдер для доступа к ODBC-данным. Таким образом, интерфейс OLE DB не заменяет интерфейс ODBC, а позволяет организовывать доступ к источникам данных через различные интерфейсы, в том числе и через ODBC.

Интерфейс ADO (ActiveX Data Objects - объекты данных ActiveX) предоставляет иерархическую модель объектов для доступа к различным OLE DB-провайдерам данных. Он характеризуется еще более высоким уровнем абстракции и базируется на интерфейсе OLE DB. Объектная модель ADO включает небольшое количество объектов, которые обеспечивают соединение с провайдером данных, создание SQL-запроса к данным, создание набора записей на основе запроса и др. Разрабатывая интерфейс ADO, фирма Microsoft предназначала его для использования в сетях Интернет/интранет для доступа к различным источникам данных.

Размещение информации из БД в Интернете представляет собой новую информационную технологию, получившую широкое распространение в последнее время в связи с ростом популярности и доступности "всемирной паутины".

# Разработка приложений информационных систем. Распределенные системы.

Информационных системы в сетях Интернет/интранет, как было отмечено, обычно строятся как многозвенные (многоуровневые) приложения.

При большом числе клиентских компьютеров многоуровневая архитектура позволяет уменьшить нагрузку на сервер баз данных и линии связи.

*Принципы функционирования Web-приложений*

Развитие сетей Интернет/интранет привело к появлению новой категории программ - Web-приложений. К Web-приложениям относят набор Web-страниц, сценариев и других программных средств, расположенных на одном или нескольких компьютерах (клиентских и серверных) и объединенных для выполнения прикладной задачи. При этом Web-приложения, публикующие БД в Интернете, представляют собой отдельный класс Web-приложений.

Современные информационные системы, построенные на основе Web-приложений, использующих БД, базируются на многоуровневой клиент-серверной архитектуре и принципах функционирования Интернета. Web-приложения выполняются на стороне Web-сервера, который находится на Web-узлах сети Интернет. Web-сервер обрабатывает запросы обозревателя на получение Web-страниц и отсылает ему требуемые данные в формате Web-документов.

Обмен данными в Интернете осуществляется на основе коммуникационного протокола TCP/IP и протокола более высокого уровня (приложений) HTTP. Упрощенная схема функционирования Web-приложения приведена на рис. 5.

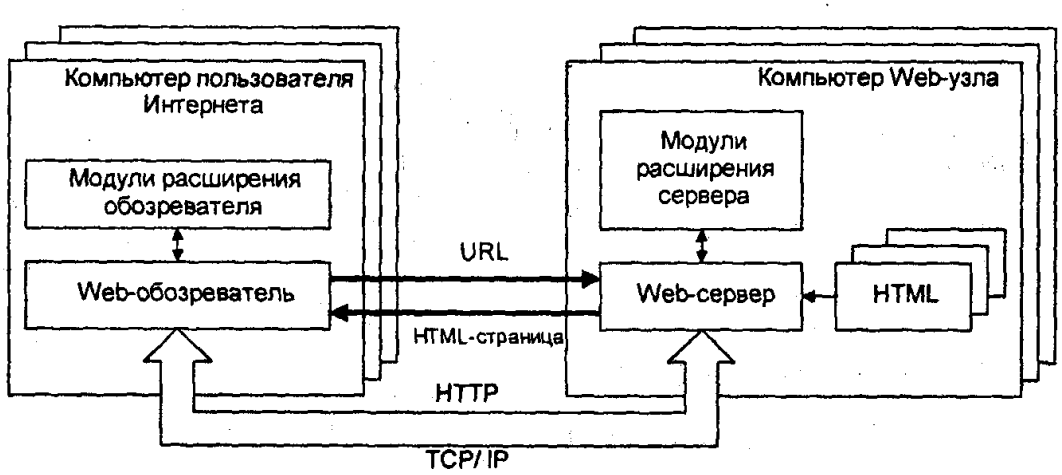


Рис.5. Схема функционирования Web-приложения с модулями расширения

Напомним, что под Web-документом (Web-страницей) понимаются используемые в Интернете документы в форматах HTML, XML, шаблоны в форматах ASP, HTX и т. д.

Для доступа к Web-страницам используются специальные клиентские программы - Web-обозреватели, находящиеся на компьютерах пользователей Интернета. Обозреватель формирует запрос на получение требуемой страницы или другого ресурса с помощью адреса URL.

Функции обозревателя заключаются в отображении Web-страниц, сгенерированных сервером или модулями расширения, и отправке запросов пользователя Web-приложению, т. е. обозреватель является связующим звеном между пользователем и Web-приложением. При этом Web-обозреватель устанавливает соединение с требуемым Web-узлом, используя различные протоколы передачи данных, в частности уже рассмотренный протокол HTTP.

*Web-приложения в сетях интранет*

Развитие технологий глобальной сети Интернет привело к появлению архитектурных и технологических решений для локальных корпоративных сетей, называемых также сетями *интранет.* В общем случае под сетью интранет понимают выделенную часть Интернета, в которой выполняется Web-приложение (информационная система).

Применение интернет-технологий в корпоративных интранет-сетях позволяет повысить эффективность функционирования сетей и используемых в них информационных систем. Приложения, построенные на основе интернет-технологий, характеризуются надежностью и масштабируемостью, открытостью архитектуры, простотой освоения и использования.

Кроме того, использование сетей интранет характеризуется значительным снижением затрат на обслуживание, модернизацию и наращивание сети по сравнению с традиционными корпоративными сетями, построенными на клиент-серверных технологиях. Важным достоинством сетей интранет является возможность развертывания корпоративных локальных и глобальных сетей на существующей инфраструктуре. При использовании Web-приложений в сети интранет может использоваться архитектура, показанная на рис. 6. Сети интранет в общем случае имеют различные внутренние структуры, причем они могут и не иметь выхода в Интернет.

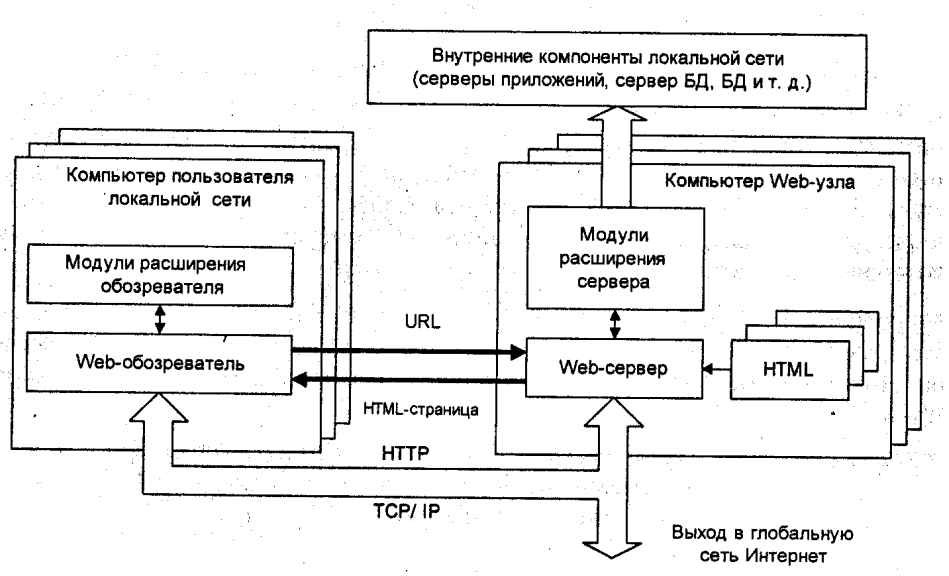


Рис. 6. Схема функционирования Web-приложения с модулями расширения в Интранет

В качестве клиентских приложений в этой архитектуре выступают Web-обозреватели, которые обращаются с запросами к серверу БД или к серверу приложений через Web-сервер. В зависимости от используемой конфигурации Web-сервер может находиться на сервере БД или на сервере приложений.

В функции Web-сервера в сети интранет входит обработка запросов Web-обозревателей на получение информации из разделяемых БД, преобразование этих запросов (может выполняться модулями расширения Web-сервера) в SQL-запросы или другие форматы, понятные для сервера БД или сервера приложений.

Кроме того, интранет-приложение предоставляет следующие дополнительные возможности.

- Удаленные доступ и управление. Концепция удаленного доступа подразумевает возможность подключения к интранет-сети извне, т. е. с любого компьютера из Интернета. Под удаленным управлением понимается подключение к локальной сети и выполнение функциональных операций по управлению ее ресурсами с удаленного компьютера. Для реализации удаленного управления необходимо наличие специального сервера удаленного доступа и специального программного обеспечения на удаленном компьютере.
- Выход в Интернет клиентов сети. При этом становятся доступными услуги, предоставляемые глобальной сетью: получение актуальной информации в различных сферах, электронная почта, обмен данными с внешними источниками, использование приложений, находящихся в Интернете, и т. д.

Сеть интранет может быть построена на основе использования Web-сервера в локальной сети или на основе услуг, предоставляемых внешним (интернетовским) Web-сервером. Такие услуги, как правило, предоставляет интернет-провайдер, обеспечивающий возможность использования

функций своего Web-сервера. Многие компании используют совмещенную структуру Web-серверов, при которой сама локальная сеть строится с использованием собственного Web-сервера, а выход в глобальную сеть осуществляется через Web-сервер провайдера.
Применение архитектуры Интернета в сетях интранет по сравнению с традиционными архитектурами локальных сетей дает следующие преимущества: стандартизация пользовательского интерфейса; более удобное администрирование и конфигурирование; удешевление установки и лицензирования клиентских компьютеров пользователей.
Для расширения возможностей клиентской части (обозревателя) и серверной части разрабатывают модули расширения обозревателя и сервера, используемые для динамического управления интерфейсными объектами (компонентами) Web-документа.
В следующих разделах мы рассмотрим взаимодействие Web-приложения с дополнительными компонентами: модулем расширения клиентской части и модулем расширения серверной части (см. рис. 5).
*Web-приложения с модулями расширения серверной части*
Архитектура Web-приложений с модулями расширения сервера (рис. 7) может включать в себя стандартные модули расширения - DLL-библиотеки, реализующие, например, технологии ASP, IDC/HTX, объекты ActiveX и пр. Кроме того, допускается подключение дополнительных модулей, разработанных с использованием интерфейсов CGI, ISAPI и др.
В этом случае в функции Web-сервера входят обработка запросов Web-обозревателей пользователей сети, вызов (загрузка) соответствующего модуля расширения сервера и передача ему параметров запроса. В результате обработки запроса модулями расширения сервера формируется Web-документ с использованием различных HTML-шаблонов. Готовый Web-документ Web-сервера отсылается обратно Web-обозревателю в формате протокола HTTP.
Как отмечалось, различают пассивное и активное состояние Web-сервера. Так, Web-сервер находится в пассивном состоянии, если формируемый им документ содержит статическую информацию, т. е. на Web-странице отсутствуют средства ввода и обработки запросов к серверу.
Если Web-документ динамически создается в ответ на запрос пользователя (рис. 7) или если в обозреватель загружены различные интерактивные визуальные элементы управления, то сервер является активным. Для публикации БД основной интерес представляет как раз активный Web-сервер, который реализуется с помощью модулей расширения.

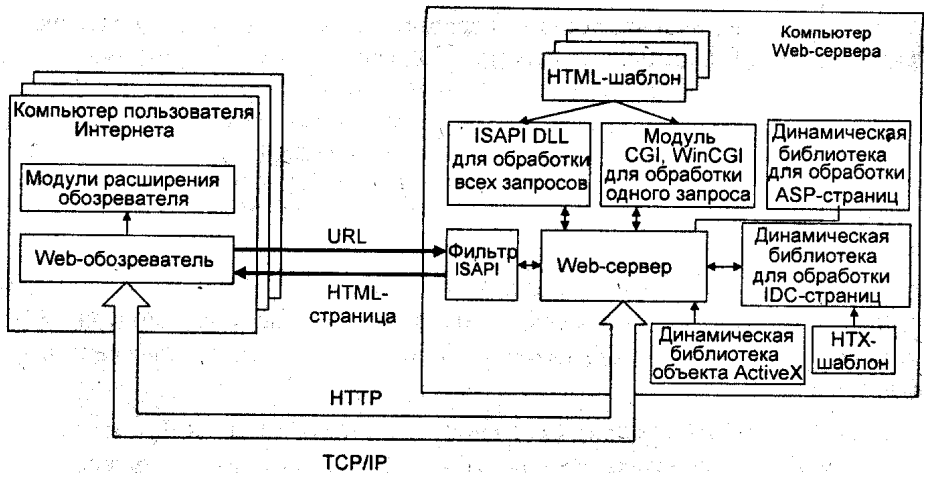


Рис.7. Архитектура Web-приложений с модулями расширения сервера

Для создания модулей расширения Web-сервера используются интерфейсы CGI, WinCGI или API.
Интерфейс *CGI* является стандартным протоколом взаимодействия между Web-сервером и модулями расширения, которые могут применяться для выполнения дополнительных функций, не поддерживаемых сервером. Напомним, что интерфейсу CGI соответствуют обычные консольные приложения операционной системы DOS. Для написания CGI-программ подходит практически любой язык программирования, обеспечивающий доступ к переменным среды и ввод/вывод через стандартные потоки STDIN и STDOUT. В частности, для написания CGI-программ подходят среды разработки Delphi и C++ Builder.

Для запуска CGI-модуля обозреватель должен сформировать запрос к серверу указанием адреса URL этого модуля. Это можно сделать следующими способами:
- задать U RL модуля CGI в адресной строке обозревателя;
- послать серверу запрос на выполнение CGI-модуля путем выбора ссылки, атрибут HREF которой содержит адрес этого модуля;
- нажать в форме КНОПКУ типа submit, у которой атрибут Action содержит URL-*адрес* CGI-модуля

После передачи запроса обозревателя CGI-приложению сервер передает ему также данные из командной строки запроса. CGI-приложение формирует ответ и помещает его в выходной поток (на стандартном устройстве вывода), затем сервер посылае этот ответ с использованием протокола HTTP обратно обозревателю.

В случае параллельной обработки нескольких запросов сервер запуска отдельный процесс для каждого запроса, причем для каждого процесса создается копия модуля расширения в памяти компьютера, на котором находится Web-сервер. При большом размере исполняемого CGI-файла сильно увеличивается время отклика сервера на запрос обозревателя, поскольку потребуется время для загрузки модуля CGI с диска в память. Причем, если CGI-программа является интерпретируемой (например, написана на языке PHP или Perl), она будет выполняться еще медленнее, т к в этом случае, кроме загрузки модуля CGI, загружается интерпретатор PHP или Perl и производится интерпретация команд При наличии сотен или тысяч обращений к серверу произойдет запуск сотен или тысяч CGI-программ, каждая из которых будет обрабатывать соответствующий запрос

Отметим, что в некоторых операционных системах, в частности, в UNIX, могут повторно использоваться уже загруженные программные коды модуля CGI первого процесса с созданием для каждого нового обращения только своего экземпляра данных

Интерфейс *WinCGI* (протокол) отличается от интерфейса CGI тем, что управляющие параметры передаются через INI-файл, а входной и выходной поток данных перенаправлены в специальные файлы. Этот интерфейс является реализацией интерфейса CGI для операционных систем Windows ранних версий.

Более перспективными интерфейсами для разработки дополнительных модулей расширения Web-сервера являются интерфейсы ISAPI/NSAPI. При использовании этих интерфейсов модули расширения реализуются в виде библиотек DLL. Такой механизм обеспечивает экономию ресурсов сервера и увеличение скорости обработки запросов.

Интерфейс ISAPI может применяться также для создания ISAPI-фильтров, которые, в отличие от модулей ISAPI, используются для контроля всего потока данных между сервером и обозревателем на уровне протокола HTTP ISAPI-фильтры можно применять для динамической перекодировки, шифрования, сбора статистической информации о работе сервера.

*Web-приложения с модулями расширения клиентской части*

Для создания динамических эффектов при просмотре Web-страниц в модулях расширения клиентской части (активность на стороне клиента) используют ап-плеты, подключаемые программы, объекты ActiveX и сценарии Архитектура Web-приложения с модулями расширения клиентской части показана на рис. 8.

Исполняемые программы являются первым поколением клиентских расширений Для организации работы такой программы в Web-приложении необходимо ее предварительно установить и сконфигурировать обозреватель Поэтому при использовании исполняемых программ очень трудно обеспечить совместимость Web-приложения с различными обозревателями Кроме того, использование такой технологии нарушает принцип стандартности клиентского интерфейса

Апплеты Java применяются для создания динамически формируемого интерфейса пользователя. Апплет представляет собой байтовый код, который интерпретируется виртуальной Java-машиной, входящей в состав обозревателя. Загрузка апплета производится при загрузке Web-документа. Принцип функционирования апплета делает возможным подключение классов Java, хранящихся на сервере. Код апплета может начать выполняться сразу после загрузки в компьютер клиента или активизироваться с помощью специальных команд.

Используя библиотеку классов Java, можно создавать мультимедийные страницы и организовывать распределенные процедуры обработки данных с использованием различных серверов и протоколов.

Апплет может быть специализирован для работы с внешними базами данных. С этой целью в Java включены наборы классов для поддержки графического пользовательского интерфейса и классы для доступа к БД.

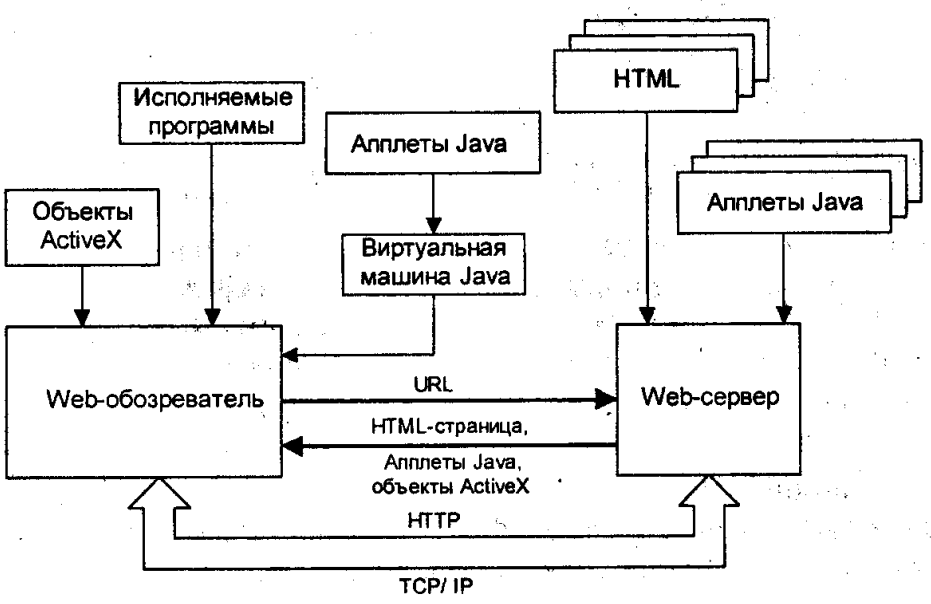


Рис. 8. Схема Web-приложения с модулями расширения обозревателя

Для включения дополнительного действия в Web-приложение достаточно включить тег апплета в Web-документ и поместить апплет-класс в библиотеку апплетов на сервере. При этом изменения в конфигурацию Web-сервера вносить не нужно.

Для взаимодействия апплета Java с внешним сервером баз данных разработан специализированный протокол JDBC (Java DataBase Connectivity - совместимость Java с базами данных), построенный на принципах интерфейса ODBC и применяется для стандартизации кода апплета Java при организации доступа к различным СУБД.

Сравним достоинства и недостатки использования технологии Java и наиболее распространенного в настоящее время интерфейса CGI. Технология апплетов Java является более гибкой. Апплет выполняется локально на машине пользователя, поэтому он может обеспечивать динамическое взаимодействие с пользователем гораздо быстрее. Кроме того, апплет может использоваться для выполнения функций, недоступных CGI-модулю. С точки зрения безопасности данных более целесообразно применять CGI-модули, т. к. они выполняются на стороне сервера и не могут получить доступ к ресурсам компьютера клиента.

## Разработка приложений информационных систем. Web-приложения, использующие базы данных.

*Архитектура Web-приложений, использующих БД*

Для использования БД в архитектуру приложений вводятся дополнительные уровни, включающие сервер БД, сервер приложений и источник БД.

При такой архитектуре Web-сервер передает запрос на генерацию Web-страниц своей программе-расширению, которая на основе информации БД формирует требуемый документ. Затем Web-сервер отсылает готовые страницы обратно обозревателю Как мы знаем, для формирования динамических страниц используются различные средства и технологии: ASP и IDC/HTX-страницы, программы-расширения сервера на основе интерфейсов CGI и ISAPI.

При использовании ASP и IDC/HTX-страниц запрос на получение динамически формируемой Web-страницы передается специальным динамическим библиотекам, входящим в состав Web-сервера. Например, если в качестве Web-сервера используется Personal Web Server и публикация осуществляется средствами IDC/HTX, то применяется динамическая библиотека httpodbc.dll.

Такие библиотеки анализируют файл ASP или IDC и HTX-файлы, которые используются в качестве шаблонов.

Напомним, что публикация БД на Web-страницах бывает статической и динамической. Программы-расширения Web-сервера, публикующие базы данных, включают в себя элементы, характерные для обычных приложений БД. При статической или динамической публикации должна использоваться архитектура приложений БД, дополненная характерными компонентами архитектуры Web-приложений. Рассмотрим подробнее архитектуру Web-приложений, использующих БД.

*Двухуровневые Web-приложения*

При двухуровневой архитектуре Web-приложений источник БД хранится на том же компьютере, где находится Web-сервер. Для доступа к источнику БД используются модули расширения. В простейшем случае в архитектуру Web-приложений добавляется источник БД (рис. 9).

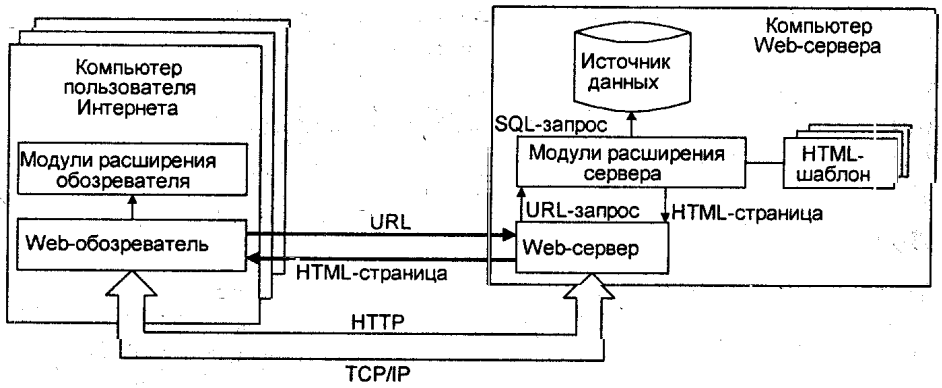


Рис. 9. Архитектура Web-приложения, использующего БД

Функционирование Web-приложения при такой архитектуре заключается в следующем. Обозреватель для начала работы с Web-приложением отсылает URL-адрес главной страницы приложения Web-серверу. Последний, обработав запрос URL, высылает требуемую страницу в формате HTML обратно обозревателю. Эта страница несет общую информацию о Web-приложении и позволяет пользователю выбрать нужную ему функцию из предоставляемых приложением. Далее возможны несколько вариантов работы Web-приложения. Если пользователю нужна определенная информация из БД, то обозреватель по ссылке, находящейся в загруженной HTML-странице, формирует URL-запрос к модулю расширения сервера. Используемые при этом технологии различаются в зависимости от типа Web-сервера и других особенностей Web-приложения. Например, если на Web-узле установлен Web-сервер Microsoft Internet Information Server, то это может быть технология ASP- или IDC/HTX-страниц, интерфейсы CGI или ISAPI, а если установлен сервер Apache, то интерфейс CGI.

Если необходимо сформировать запрос URL с параметрами, то на уровне обозревателя могут использоваться сценарии JavaScript для проверки правильности ввода параметров запроса.

После того как пользователь выбрал ссылку, обозреватель отсылает URL Web-серверу. Для обработки запроса сервер вызывает требуемый модуль расширения и передает ему параметры URL. Модуль расширения сервера формирует SQL-запрос к БД.

Из модуля расширения сервера доступ к БД может осуществляться различными способами и на основе различных интерфейсов. Например, в случае использования технологии ASP-страниц применяются объектная модель ADO, объектный интерфейс OLE DB, интерфейс ODBC. Также возможен вариант непосредственного доступа к БД. Например, в случае модуля ISAPI, разработанного в среде Delphi, для доступа к БД может использоваться один посредник - драйвер BDE (Borland DataBase Engine), входящий в состав программных средств модуля расширения сервера.

Недостатки рассмотренной двухуровневой архитектуры:

- повышенная нагрузка на Web-сервер, связанная с тем, что все действия по обработке URL-запросов, извлечению информации из БД и формированию HTML-страниц выполняются Web-сервером и модулями расширения Web-сервера;
- низкий уровень безопасности из-за невозможности обеспечить требуемый уровень защиты информации в БД от сбоев во время обращения к базе данных из модуля расширения сервера или конфиденциальности информации БД от администратора Web-узла.

Для преодоления указанных недостатков применяются Web-приложения с большим числом уровней.

*Трехуровневые Web-приложения*

При включении в Web-приложение промежуточного уровня, основанного на технологии "клиент-сервер", его архитектура расширяется до трехуровневой. При такой архитектуре клиентский уровень занимает обозреватель, на уровне сервера находится сервер БД, а на промежуточном уровне размещаются Web-сервер и модули расширения сервера. Модуль расширения сервера выступает преобразователем протоколов между клиент-серверным приложением БД и Web-сервером (рис. 10).

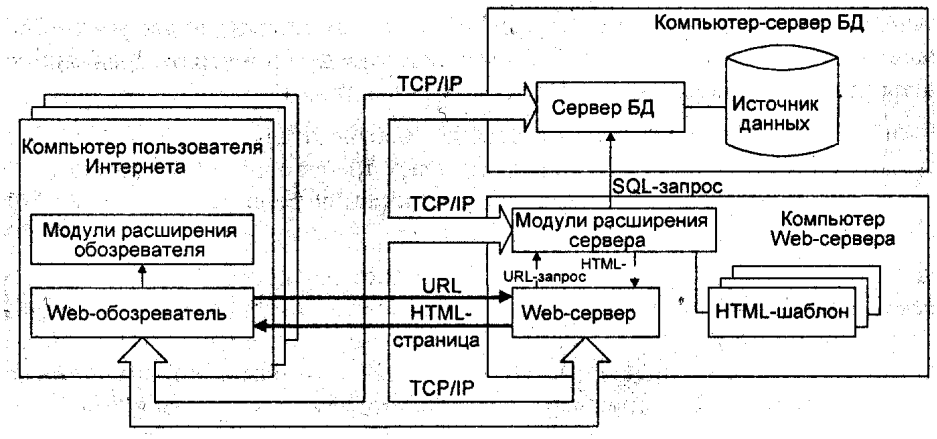


Рис. 10. Архитектура трехуровневого Web-приложения, использующего БД

Введение уровня Web-сервера в клиент-серверные приложения БД расширяет возможность их применения как межплатформенного приложения. Принципы взаимосвязи обозревателя и Web-сервера остаются те же, что и в предыдущей архитектуре. Отличия этой архитектуры заключаются в организации взаимодействия модуля расширения сервера и источника БД. Для получения данных модуль расширения Web-сервера формирует и отсылает SQL-запрос удаленному серверу БД. На компьютере, где установлен удаленный сервер БД, содержится и сама база данных. После получения SQL-запроса удаленный сервер направляет его SQL-серверу (серверу баз данных). SQL-сервер обеспечивает выполнение запроса и выдачу модулю расширения Web-сервера результатов запроса.

Таким образом, в трехуровневой архитектуре вся обработка SQL-запроса выполняется на удаленном сервере. Достоинства такой архитектуры по сравнению с предыдущей состоят в следующем:
- уменьшение сетевого трафика - в сети циркулирует минимальный объем информации;
- увеличение уровня безопасности информации, поскольку обработка запросов к базе данных выполняется сервером БД, который запрещает одновременное изменение одной записи различными пользователями и реализует механизм транзакций;
- повышение устойчивости Web-приложения к сбоям;
- взаимозаменяемость компонентов архитектуры приложения;
- снижение сложности модулей расширения Web-сервера, в которых отсутствует программный код, связанный с контролем БД и разграничением доступа к ней.

Недостатком рассмотренной архитектуры является увеличение времени обработки запросов, связанное с дополнительным обращением по сети к серверу БД. Для устранения этого недостатка между сервером БД и Web-сервером должны использоваться высокоскоростные надежные линии связи.

# Разработка приложений информационных систем. Многоуровневые Web-приложения.

*Многоуровневые Web-приложения*

Дальнейшее развитие архитектуры Web-приложений и технологии "клиент-сервер" привело к появлению многоуровневой архитектуры, в которой между модулем расширения Web-сервера и базой данных, кроме сервера БД, дополнительно вводится сервер приложений. Сервер приложений является промежуточным уровнем, который обеспечивает организацию взаимодействия клиентов ("тонких" клиентов) и сервера БД (рис. 11).

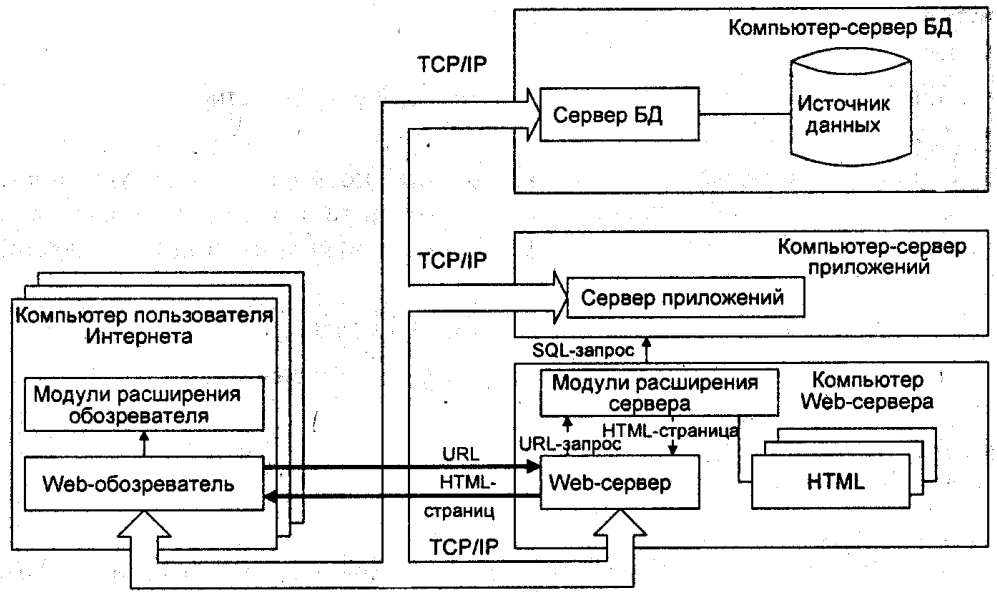


Рис. 11. Архитектура многоуровневого Web-приложения

Напомним, что сервер приложений может использоваться для выполнения различных функций, которые в предыдущей архитектуре выполнялись сервером БД или модулем расширения Web-сервера.

В качестве "тонкого" клиента в этой архитектуре выступает программа-модуль расширения Web-сервера. Сервер приложений может обеспечивать взаимодействие с Web-серверами и серверами БД, функционирующими на различных аппаратно-программных платформах (на компьютерах различных типов, под управлением различных операционных систем). Такая архитектура является основой для интранет-сетей, создаваемых на основе существующих локальных сетей.

Введение дополнительного уровня Web-сервера позволяет публиковать информацию из БД локальных сетей в Интернете, получать информацию от других интранет-сетей или Web-узлов. Кроме того, при частичной или полной реорганизации внутренней архитектуры локальных сетей появляется возможность использовать преимущества сетей интранет, касающиеся упрощения дополнительного подключения новых пользователей и администрирования локальной сети Отметим, что в некоторых архитектурах информационных систем Web-сервер может структурно объединяться с сервером приложений. В этом случае программные средства, входящие в состав модуля расширения, выполняют роль сервера приложений.

Основные достоинства многоуровневой архитектуры Web-приложений:

- разгрузка Web-сервера от выполнения части операций, перенесенных на сервер приложений, и уменьшение размера модулей расширения сервера вследствие их освобождения от лишнего кода;
- обеспечение более гибкого межплатформенного управления между Web-сервером и сервером БД;
- упрощение администрирования и настройки параметров сети — при внесении изменений в программное обеспечение или конфигурацию сервера БД не нужно вносить изменения в программное обеспечение Web-сервера.

При функционировании Web-приложений с использованием многоуровневой архитектуры сохраняется возможность параллельной работы Web-обозревателей и клиентских приложений БД (рис. 12).

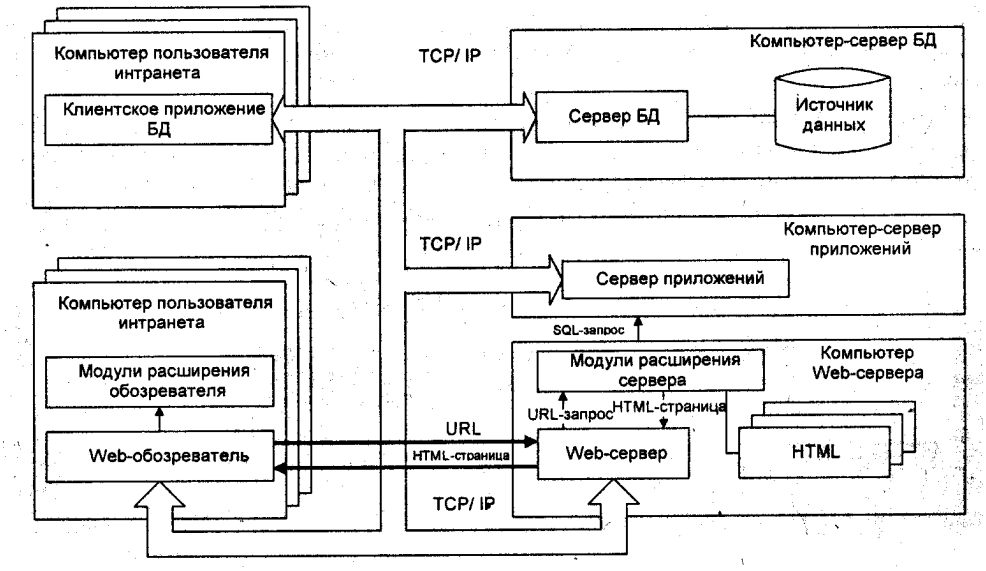


Рис. 12. Архитектура смешанного Web-приложения

В этом случае говорят о *смешанном* Web-приложении. При такой архитектуре Web-приложения и клиентские приложения БД могут параллельно получать доступ к источнику БД.

*Web-приложения на основе CORBA*

В настоящее время одним из перспективных направлений развития интранет-сетей является использование технологии CORBA (Common Object Request Broker Architectur - общая архитектура с брокером при запросе объекта) в сочетании с апплетами Java. CORBA представляет собой шину (интерфейс) распределенных объектов с открытыми стандартами, используемую в клиент-серверных системах. Единственной конкурентноспособной технологией, обладающей аналогичными возможностями, является технология DCOM (Distributed Component Object Model — распределенная объектная модель компонентов) фирмы Microsoft.

Стандарт CORBA не зависит от платформ и операционных систем. Поскольку технология CORBA хорошо интегрирована с Java, мы можем получить преимущества от совместного использования в трехуровневой архитектуре продуктов Delphi и JBuilder (интегрированной среды разработки на языке Java фирмы Borland).

Технология Java предполагает большую гибкость при разработке распределенных приложений, но не поддерживает в полной мере технологию "клиент-сервер". Интерфейс CORBA позволяет обеспечить связь переносимых приложений Java и объектов CORBA. Технология объектов CORBA предназначена для использования в Web-приложениях вместо CGI-интерфейса.

В результате объединения Java-апплетов и CORBA-интерфейса появилось новое понятие - *объектная модель Web,* означающее использование объектных моделей различных интерфейсов (модели CORBA, ADO и др.) при построении Web-приложений Архитектура такого многоуровневого клиент-серверного Web-приложения, построенного на основе технологии CORBA и Java, приведена на рис. 13.

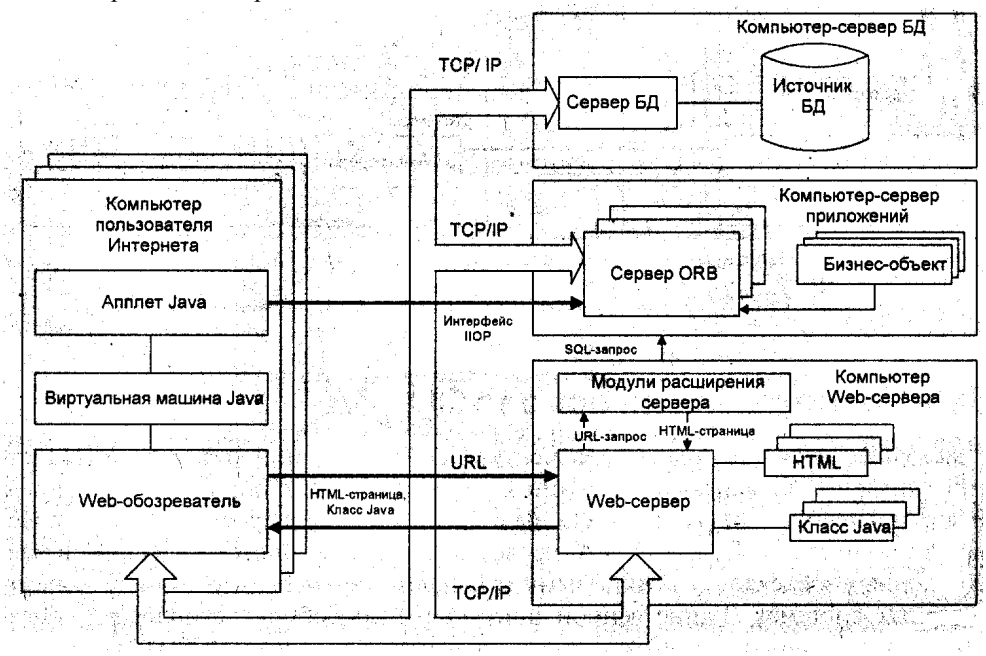


Рис. 13. Архитектура многоуровневого Web-приложения на основе технологии CORBA

На первом уровне находится клиентское приложение - обозреватель. В нем выполняется клиентский апплет Java, из которого может осуществляться обращение к объектам CORBA.
На втором уровне находится Web-сервер, обрабатывающий HTTP-запросы и CORBA-вызовы клиентских приложений. Третий уровень - это уровень сервера приложений. В его роли могут выступать серверы ORB (Object Request Broker - посредник запросов объектов) или распределенные объекты CORBA, функционирующие как серверы приложений промежуточного звена и выполняющие прикладные функции и набор компонентных сервисов (услуг). Серверы ORB являются унифицированными фрагментами программы, используемыми в распределенных приложениях в качестве связующего звена между клиентскими приложениями и сервером.
Объекты CORBA взаимодействуют с серверами БД последнего уровня, используя, например, SQL в случае реляционных БД. Кроме того, объекты CORBA на сервере могут взаимодействовать и друг с другом. В основе механизма взаимодействия между объектами CORBA лежит протокол IIOP (Internet Inter-ORB Protocol - интернет-протокол взаимодействия ORB). Протокол IIOP основан на протоколе TCP/IP с добавленными компонентами обмена сообщениями и функционирует как общий опорный протокол при организации взаимодействия серверов ORB и объектов CORBA. В дополнение к IIOP, в технологии CORBA используются ESIOP-протоколы (Environment-Specific Inter-ORB Protocols - зависящие от среды протоколы взаимодействия ORB), применяемые в специализированных сетевых средах.
Java-клиент может непосредственно взаимодействовать с объектом CORBA, используя Java ORB. При этом серверы CORBA замещают уровень HTTP-сервера и выступают в качестве программного обеспечения промежуточного уровня, обеспечивая взаимодействие между объектами (object-to-object). Интерфейс CORBA IIOP функционирует в Интернете так же, как и протокол HTTP.
Протокол HTTP в этом случае используется для загрузки Web-документов, апплетов и графики, а CORBA - для организации клиент-серверных приложений с помощью апплетов Java.
Серверный компонент CORBA предоставляет "настраиваемый" интерфейс, который можно конфигурировать с помощью визуальных средств. Объект CORBA обладает определенными функциональными возможностями, реализует инкапсуляцию свойств и методов, генерируемых объектами событий. Можно создавать ансамбли объектов, "стыкуя" выходные события с входными методами. Разработка таких визуальных объектов поддерживается средствами

быстрой разработки приложений (RAD). В частности, объекты CORBA поддерживаются в Delphi.

На последнем, четвертом, уровне размещается сервер баз данных или другой источник данных, т. е. практически любой источник информации, к которому CORBA может получить доступ. Сюда входят мониторы процедур транзакций (TP Monitors), MOM (Message-Oriented Middleware - промежуточное программное обеспечение, ориентированное на обмен сообщениями), ODBMS (объектные СУБД), электронная почта и т. д.

Таким образом, CORBA обеспечивает инфраструктуру распределенных объектов, что позволяет приложениям распространяться через сети, языки, границы компонентов и операционные системы. В свою очередь, Java обеспечивает инфраструктуру переносимых объектов, которые работают на всех основных операционных системах. То есть CORBA дает независимость от сетей, а Java - независимость от реализации.

Для организации связи программных расширений Web-сервера с БД используются современные интерфейсы доступа к данным: OLE DB, ADO и ODBC. Эти интерфейсы являются промежуточным уровнем между источником данных и приложением, в качестве которого выступают программные расширения Web-сервера. Рассмотрим особенности архитектуры Web-приложений, использующих интерфейсы доступа к данным OLE DB, ADO и ODBC.

*Web-приложения на основе OLE DB, ADO и ODBC*

Современные интерфейсы OLE DB, ADO и ODBC, внедряемые фирмой Microsoft, позволяют осуществлять доступ к различным источникам данных единообразными способами. Доступ к данным основан на интерфейсе OLE DB, позволяющем связывать и внедрять объекты из любых источников. (Напомним, что интерфейс OLE DB является универсальной технологией доступа к данным через стандартный интерфейс COM и основан на механизме сервис-провайдеров (поставщиков услуг).)

Архитектура Web-приложений, использующих интерфейсы OLE DB, ADO и ODBC, приведена на рис. 14.

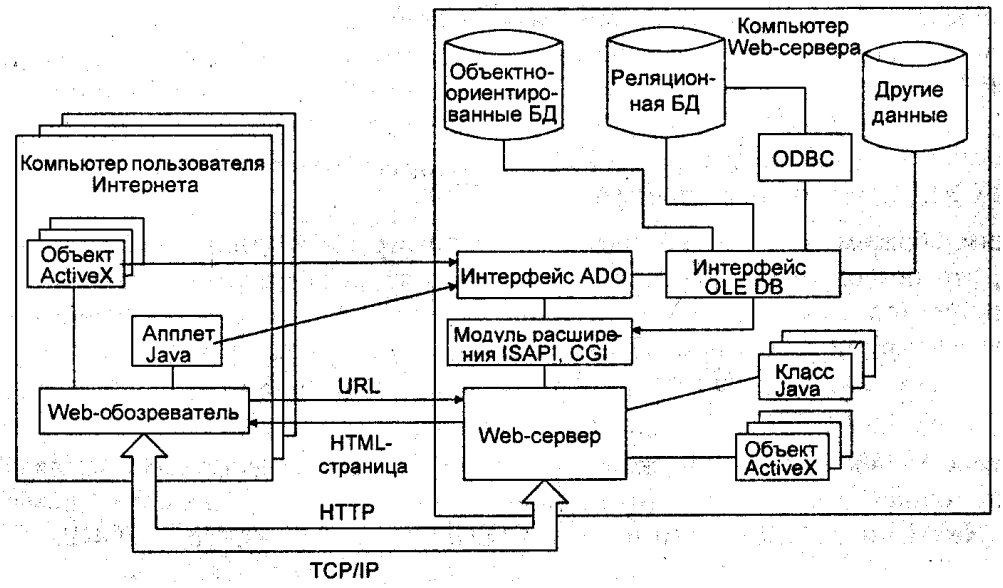


Рис.14. Архитектура Web-приложений с интерфейсами OLE DB, ADO и ODBC

Интерфейс ADO представляет собой еще более высокий уровень абстракции, чем интерфейс OLE DB. Он реализован в виде иерархической модели объектов для доступа к различным OLE DB-провайдерам данных. В модель ADO входит набор объектов, которые обеспечивают соединение с провайдером данных, создание SQL-запроса к данным, создание набора записей на основе запроса и др.

Особенность функционирования Web-приложений, использующих интерфейс ADO, заключается в том, что обозреватель может извлекать информацию из любого источника данных, находящегося в Интернете, заранее не имея представления о логической структуре, типе и физическом формате источника данных. То есть появляется возможность публиковать требуемую информацию в Интернете, не показывая внутреннюю структуру данных.

Таким образом, интерфейс ADO позволяет стандартизировать существующие интерфейсы для доступа к данным в сетях Интернет/интранет. Он объединяет все существующие интерфейсы и предоставляет единообразный способ доступа к любому источнику данных через любой интерфейс.

## Реализация примера разработки приложений информационной системы. (является самостоятельной работой магистранта)