Раздел 2 Архитектура клиент – сервер кис. и ее разновидности

Значение ПО промежуточного слоя

В любой ИС можно выделить 3 слоя

Представление данных,
Бизнес – функции (решение деловых проблем) и
Данные.

- . модель сервера приложений (Application Server - AS).

Прежде чем перейти к рассмотрению этих рахновидностей приведем простейшую структуру двухслойной системы КС

Рис.1

Различия в реализациях технологии "клиент-сервер" определяются четырьмя факторами. Во-первых, тем, в какие виды программного обеспечения интегрированы каждый из этих компонентов. Во-вторых, тем, какие механизмы программного обеспечения используются для реализации функций всех трех слоев. В-третьих, как логические компоненты распределяются между компьютерами в сети. В-четвертых, какие механизмы используются для связи компонентов между собой.

Выделяются четыре подхода, реализованные в моделях:

- . модель файлового сервера (File Server - FS);
- . модель доступа к удаленным данным (Remote Data Access - RDA);
- . модель севера базы данных (DataBase Server - DBS);

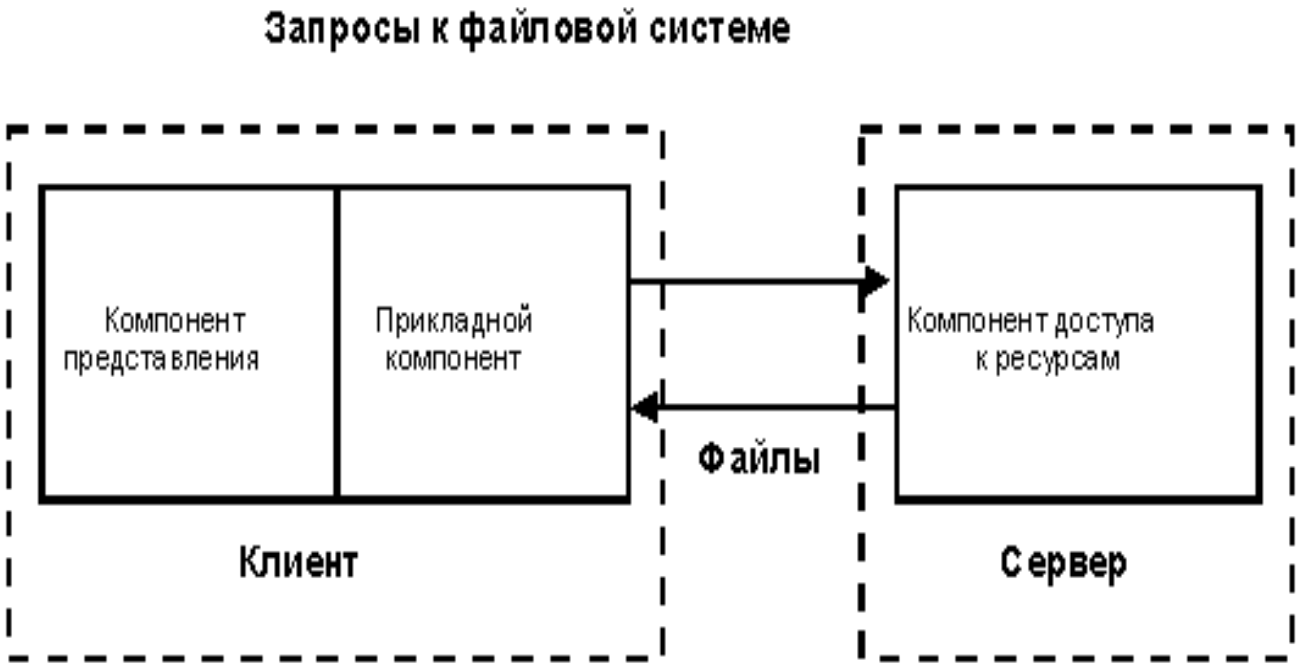

Рис.2

FS-модель является базовой для локальных сетей персональных компьютеров. Не так давно она была исключительно популярной среди отечественных разработчиков, использовавших такие системы, как FoxPRO, Clipper, Clarion, Paradox и т.д. Суть модели проста и всем известна. Один из компьютеров в сети считается файловым сервером и предоставляет услуги по обработке файлов другим компьютерам. Файловый сервер работает под управлением сетевой операционной системы (например, Novell NetWare) и играет роль компонента доступа к информационным ресурсам (то есть к файлам). На других компьютерах в сети функционирует приложение, в кодах которого совмещены компонент представления и прикладной компонент

К технологическим недостаткам модели относят высокий сетевой трафик (передача множества файлов, необходимых приложению), узкий спектр операций манипуляции с данными ("данные - это файлы"), отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы) и т.д. Собственно, перечисленное не есть недостатки, но - следствие внутренне присущих FS-модели ограничений

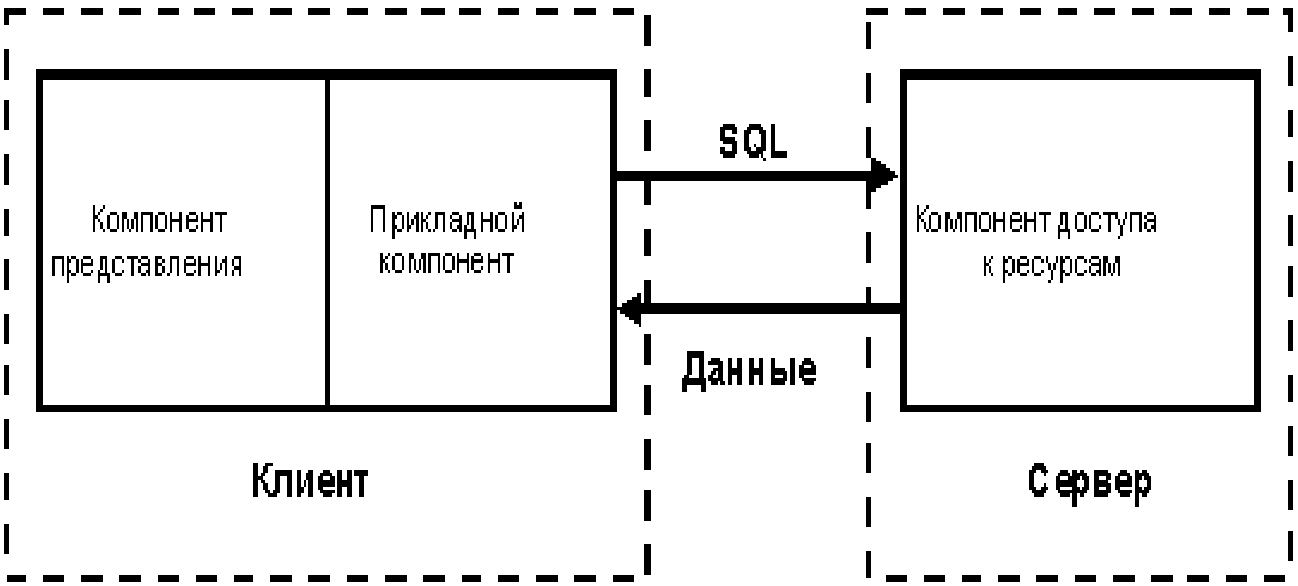

Рис.3

Более технологичная RDA-модель существенно отличается от FS-модели характером компонента доступа к информационным ресурсам. Это, как правило, SQL-сервер. В RDA-модели коды компонента представления и прикладного компонента совмещены и выполняются на компьютере-клиенте. Доступ к информационным ресурсам обеспечивается либо операторами специального языка (языка SQL), либо вызовами функций специальной библиотеки (если имеется соответствующий интерфейс прикладного программирования - API).
Основное достоинство RDA-модели - унификация интерфейса "клиент-сервер" в виде языка SQL

Наряду с RDA-моделью все большую популярность приобрела перспективная DBS-модель. Последняя реализована в реляционных СУБД (Informix, Ingres, Sybase, Oracle). Ее основу составляет механизм хранимых процедур - средство программирования SQL-сервера. Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, где функционирует SQL-сервер. Язык, на котором разрабатываются хранимые процедуры, представляет собой процедурное расширение языка запросов SQL и уникален для каждой конкретной СУБД.

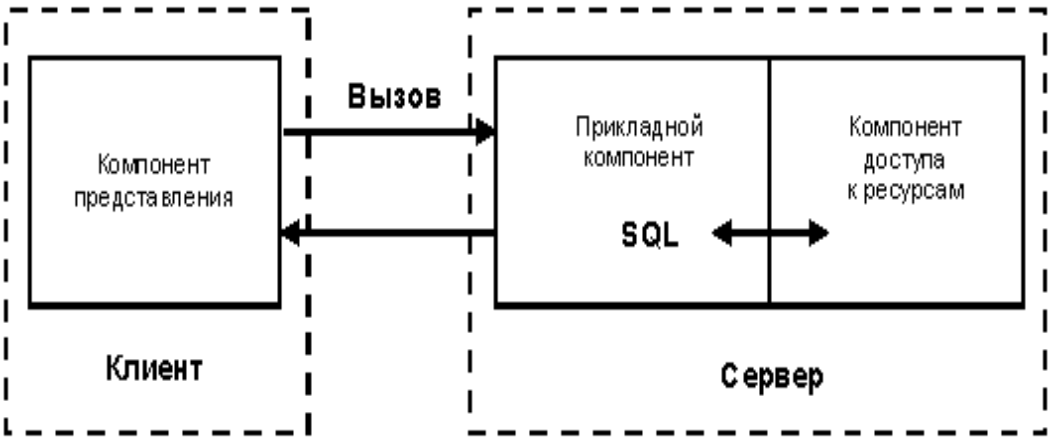

Рис.4

. Достоинства DBS-модели очевидны: это и возможность централизованного администрирования прикладных функций, и снижение трафика (вместо SQL-запросов по сети направляются вызовы хранимых процедур), и возможность разделения процедуры между несколькими приложениями, и экономия ресурсов компьютера за счет использования единожды созданного плана выполнения процедуры.

- К недостаткам модели можно отнести ограниченность средств, используемых для написания хранимых процедур, которые представляют собой разнообразные процедурные расширения SQL, не выдерживающие сравнения по изобразительным средствам и функциональным возможностям с языками третьего поколения, такими как C или Pascal.
- Сфера их использования ограничена конкретной СУБД, в большинстве СУБД отсутствуют возможности отладки и тестирования разработанных хранимых процедур

3-х уровневая архитектура «клиент – сервер»

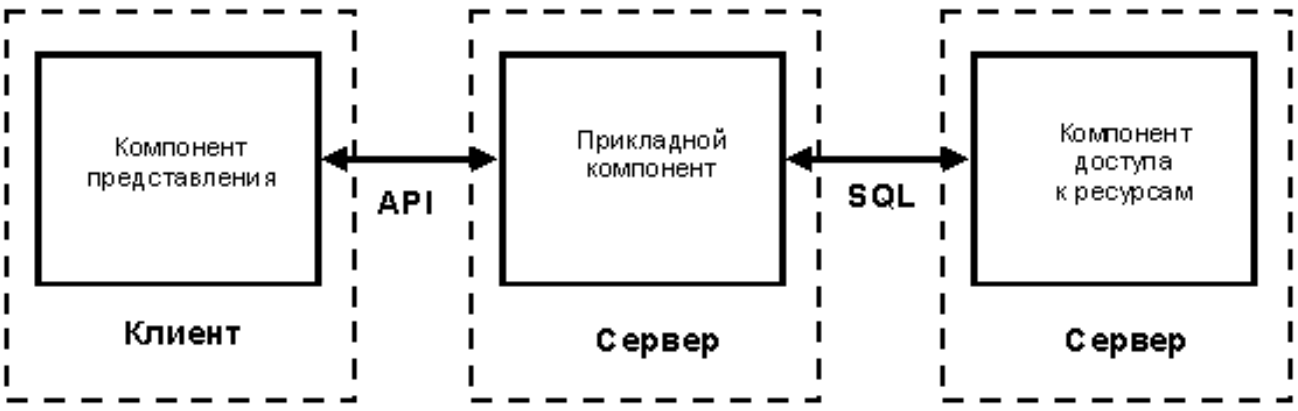

Рис.5

RDA- и DBS-модели опираются на двухзвенную схему разделения функций.

- В RDA-модели прикладные функции приданы программе-клиенту,

- в DBS-модели ответственность за их выполнение берет на себя ядро СУБД.
- В первом случае прикладной компонент сливается с компонентом представления, во-втором - интегрируется в компонент доступа к информационным ресурсам.
- В AS-модели реализована трехзвенная схема разделения функций, где прикладной компонент выделен как важнейший изолированный элемент приложения, получивший название ПО промежуточного слоя или Middleware.

На сегодняшний день существует три наиболее распространенных варианта технологий для построения ПО промежуточного уровня: **CORBA** на основе брокеров объектных запросов (Object Request Brokers — ORBs); **мониторы обработки транзакций** (Transaction Processing Monitors — TP-Monitors) и **серверы Web-приложений.**

Каждая из этих технологий имеет свои сильные стороны, но ни одна из них идеально не подходит для требований WebOLTP на промежуточном уровне.

Достоинства и недостатки CORBA

объекты CORBA имеют превосходные возможности построения многоуровневой архитектуры с вызовом сильно распределенных объектов и прочими сервисами.

но сложность общего решения и недостаток надежных средств поддержки ограничивает их применение только квалифицированными разработчиками.

К тому же, большинство ORB сегодня имеют примитивные механизмы исполнения на стороне сервера, что также ограничивает эффективность и масштабируемость

TP мониторы имеют устойчивые и отработанные механизмы выполнения, которые предоставляют превосходную эффективность и масштабируемость. Однако, подобно объектам ORB, их общая сложность и собственный интерфейс API зачастую делает TP мониторы трудными в использовании и дорогими с точки зрения установки, управления и поддержки

Серверы Web-приложений вообще являются специализированными (заказными) разработками на основе одного из инструментальных средств создания Web-узла Технология сервера Web-приложений появилась в результате попытки трансформировать Netscape и Web-серверы Microsoft в серверы приложений; рычагами к этому послужило последнее поколение соответствующих API — (NSAPI и ISAPI)..

Чтобы преодолеть слабые стороны существующих систем и удовлетворить требованиям WebOLTP на промежуточном уровне, обеспечивающим масштабируемость и простоту использования, был предложен новый класс системного ПО: серверы транзакций

- Серверы транзакций характеризуются следующими отличительными свойствами:
- предлагают встроенные возможности управления транзакциями,
- обеспечивают механизм запуска и управления сервлетами (servlets),
- поддерживают вызовы распределенных объектов для обеспечения связи в многоуровневых приложениях,

- поддерживают средства быстрой разработки ПО для промежуточного уровня, включая компонентную разработку

Серверы Транзакции объединяют самые лучшие возможности CORBA и мониторов TP с компонент-основанной разработкой: это дает возможность быстрому созданию масштабируемых WebOLTP приложений. Первыми доступными серверами транзакций стали Sybase Jaguar CTS (Компонентный сервер транзакций) от Sybase, Inc. и Microsoft Transaction Server (прежде известные как Viper).

Упрощенная структура Сервера транзакций

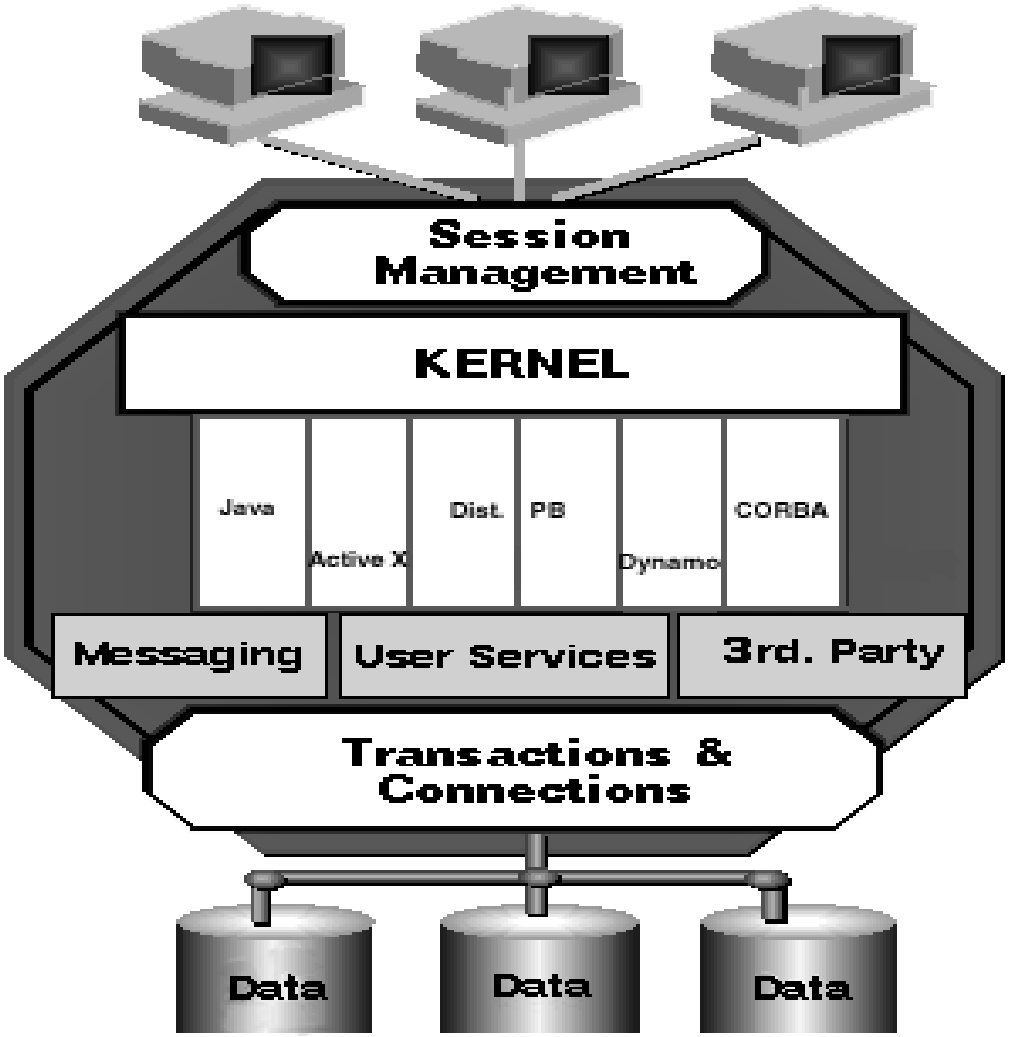

Рис.6

Jaguar CTS включают в себя:

масштабируемый, независимый от платформы механизм выполнения, быструю связь между всеми уровнями,

компонентную разработку с поддержкой всех ведущих моделей компонентов, включая: ActiveX, JavaBeans, C++ и объекты CORBA,

полную поддержку защиты в Интернет, включая SLL шифрование и авторизацию, а также списки управления доступом к уровню приложений,

гибкое управление транзакциями с поддержкой как обычной синхронной, так и асинхронной диалоговой обработки запросов.