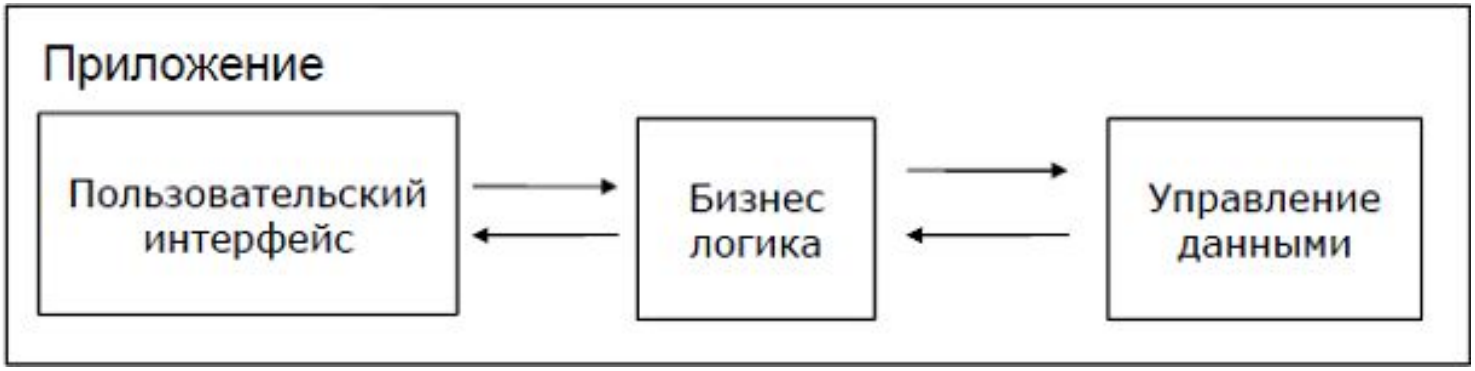
Учебный курс

# *Архитектура информационных систем*

# Архитектура многопользовательских СУБД

**Архитектура информационной системы** - концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

# Базовые функции информационных систем



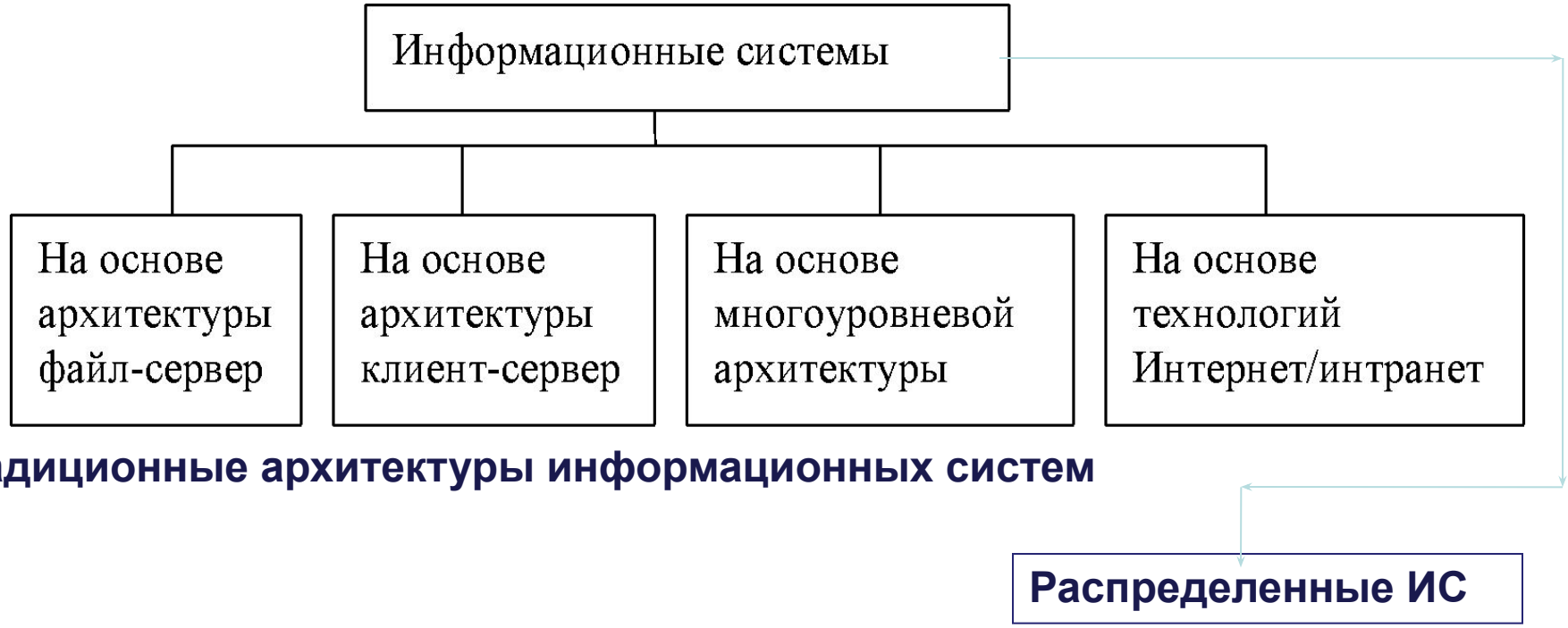

Компоненты информационной системы по выполняемым функциям можно разделить на три слоя: слой представления, слой бизнес-логики и слой доступа к данным.

- **Слой представления** - все, что связано с взаимодействием с пользователем: нажатие кнопок, движение мыши, отрисовка изображения, вывод результатов поиска и т.д.
- **Бизнес логика** - правила, алгоритмы реакции приложения на действия пользователя или на внутренние события, правила обработки данных.
- **Слой доступа к данным** - хранение, выборка, модификация и удаление данных, связанных с решаемой приложением прикладной задачей

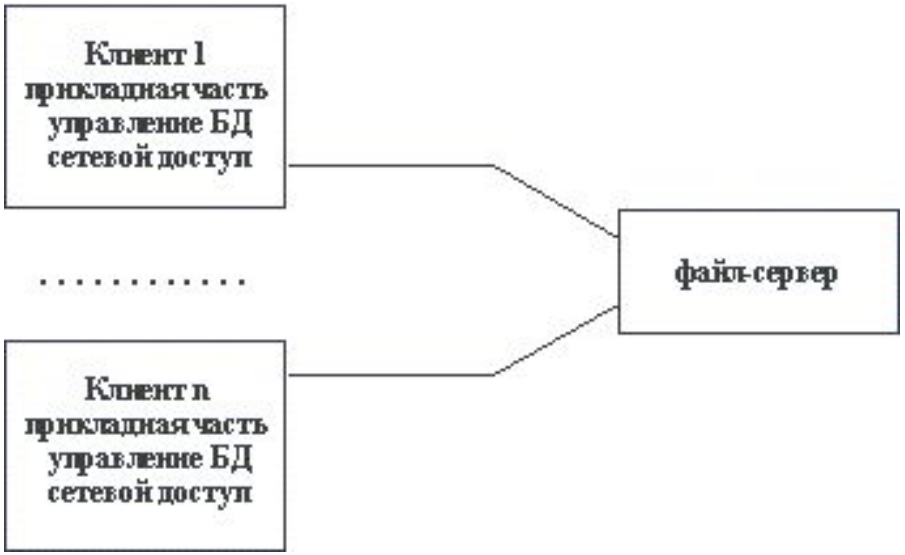
# Типовые функциональные компоненты информационной системы

| Обозначение | Наименование | Характеристика |
|---|---|---|
| **PS** | Presentation Services (средства представления) | Обслуживает пользовательский ввод и отображает 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, щелчке на кнопке или выборе пункта в списке |
| **BL** | Business Logic (прикладная логика) | Набор правил для принятия решений, вычислений и операций, которые должно выполнить приложение |
| **DL** | Data Logic (логика управления данными) | Операции с базой данных (реализуемые SQL-операторами), которые нужно выполнить для реализации прикладной логики управления данными |
| **DS** | Data Services (операции с базой данных) | Действия СУБД, реализующие логику управления данными, такие как, манипулирование данными, определение данных, фиксация или откат транзакций и т.п. СУБД обычно компилирует SQL-приложения |
| **FS** | File Services (файловые операции) | Дисковые операции чтения и записи данных для СУБД и других компонентов. Обычно являются функциями операционной системы (ОС) |

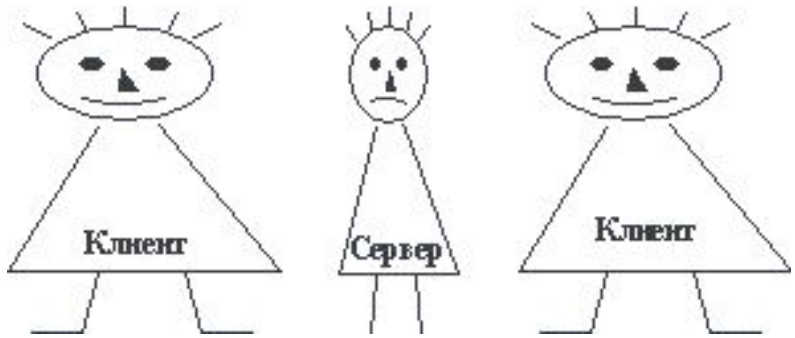
# Классификация ИС по способу организации



# Архитектура файл-сервер




Классическое представление ИС
в архитектуре "файл-сервер"

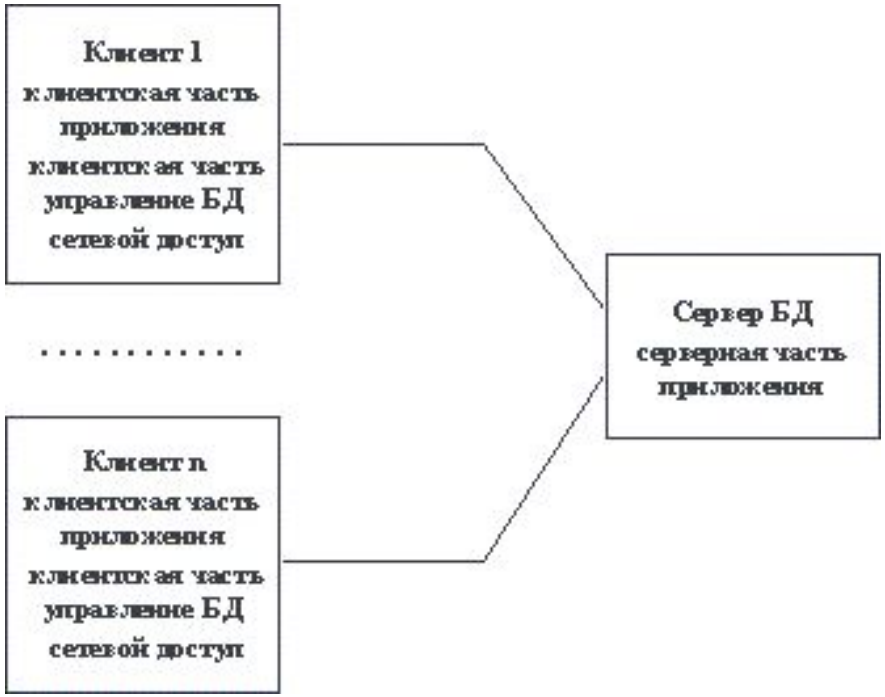



"Толстый" клиент и "тонкий" сервер
в файл-серверной архитектуре

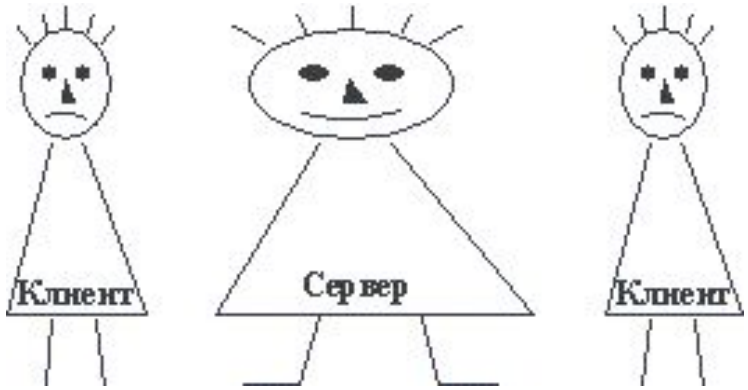
Объекты разработки: PL, BL, управление DL

Разработанное приложение реализуется либо в виде законченного загрузочного модуля, либо в виде специального кода для интерпретации.

# Клиент-серверные приложения



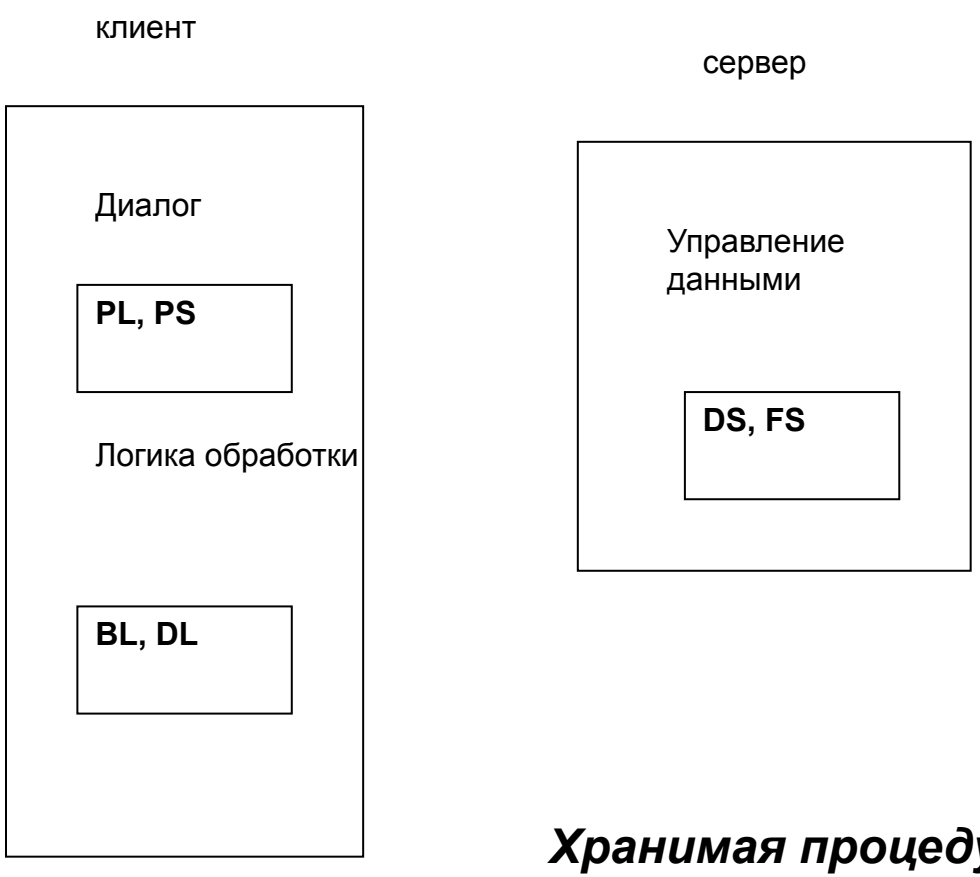

Общее представление ИС
в архитектуре "клиент-сервер"




"Тонкий" клиент и "толстый" сервер
в клиент-серверной архитектуре

- Особенностью архитектуры клиент-сервер является наличие **выделенных серверов баз данных**, понимающих запросы на языке структурированных запросов (Structured Query Language, SQL) и выполняющих поиск, сортировку и агрегирование информации.
- Отличительная черта серверов БД — наличие справочника данных, на котором записаны структура БД, ограничения целостности данных, форматы и даже серверные процедуры обработки данных по вызову или по событиям в программе
- Объектами разработки для таких приложений, помимо диалога (DL) и логики обработки (PL, BL) являются, прежде всего, реляционная модель данных и связанный с ней набор SQL-операторов для типовых запросов к базе данных

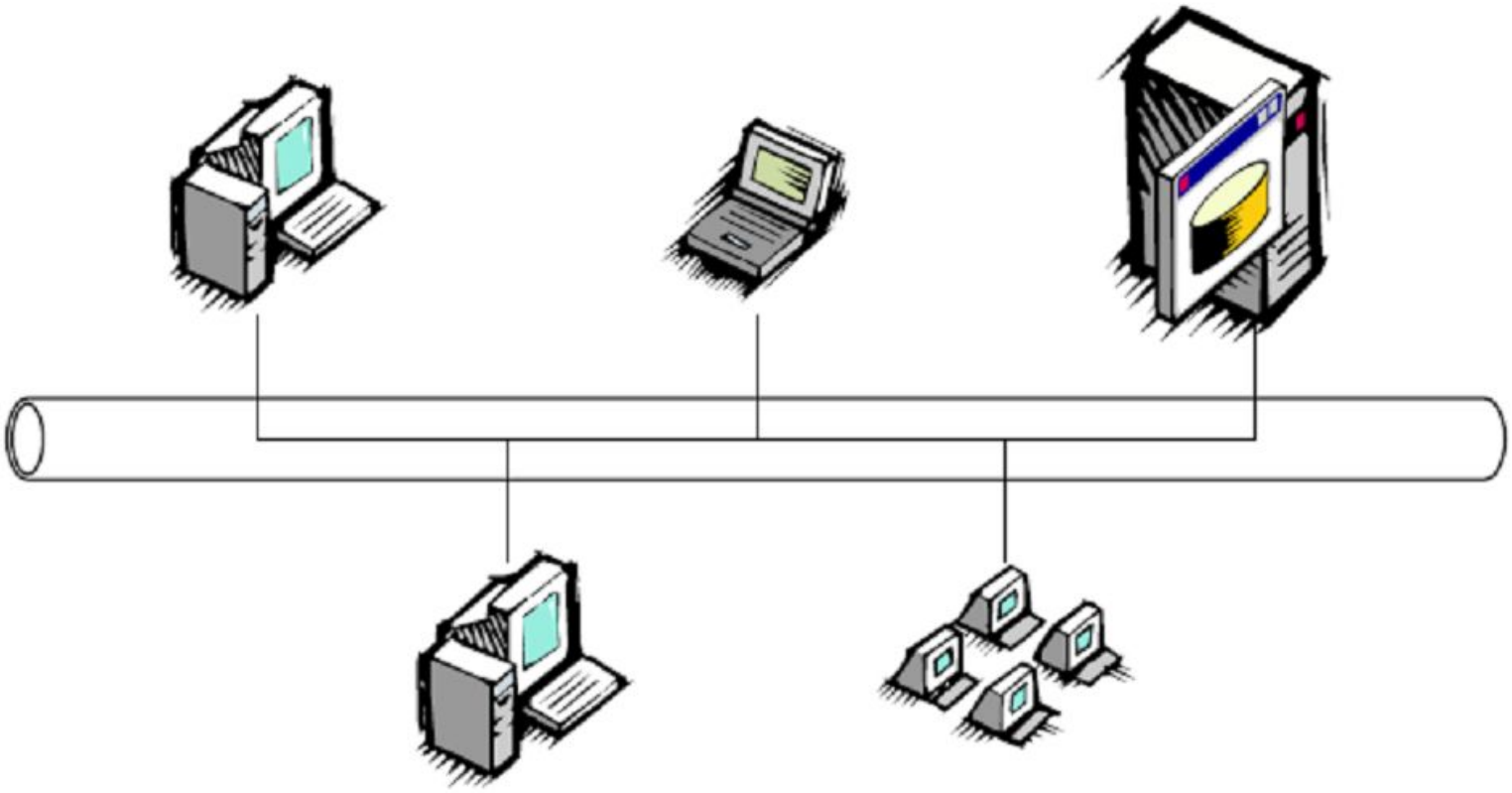
# Классический вариант клиент-серверной системы




Ключевым отличием архитектуры клиент-сервер от архитектуры файл-сервер является абстрагирование от внутреннего представления данных (физической схемы данных).
Теперь клиентские программы манипулируют данными на уровне логической схемы.

***Хранимая процедура*** — процедура с SQL-операторами для доступа к БД, вызываемая по имени с передачей требуемых параметров и выполняемая па сервере БД

# Клиент-серверная архитектура



Основные особенности:

- Клиентская программа работает с данными через запросы к серверному ПО.
- Базовые функции приложения разделены между клиентом и сервером.

# Модель сервера СУБД

# Клиент-серверная архитектура

**Плюсы:**

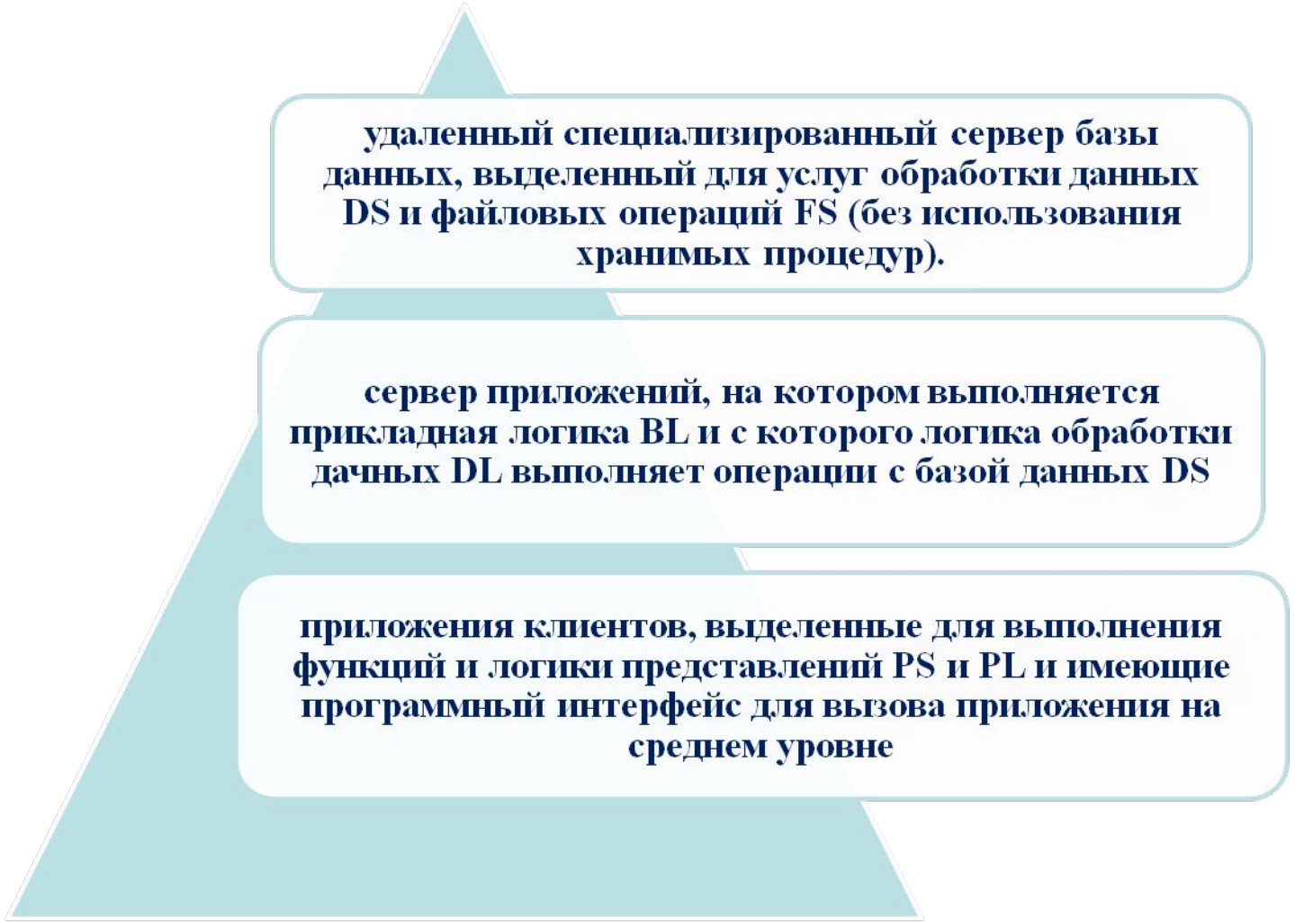
- Полная поддержка многопользовательской работы
- Гарантия целостности данных

**Минусы:**

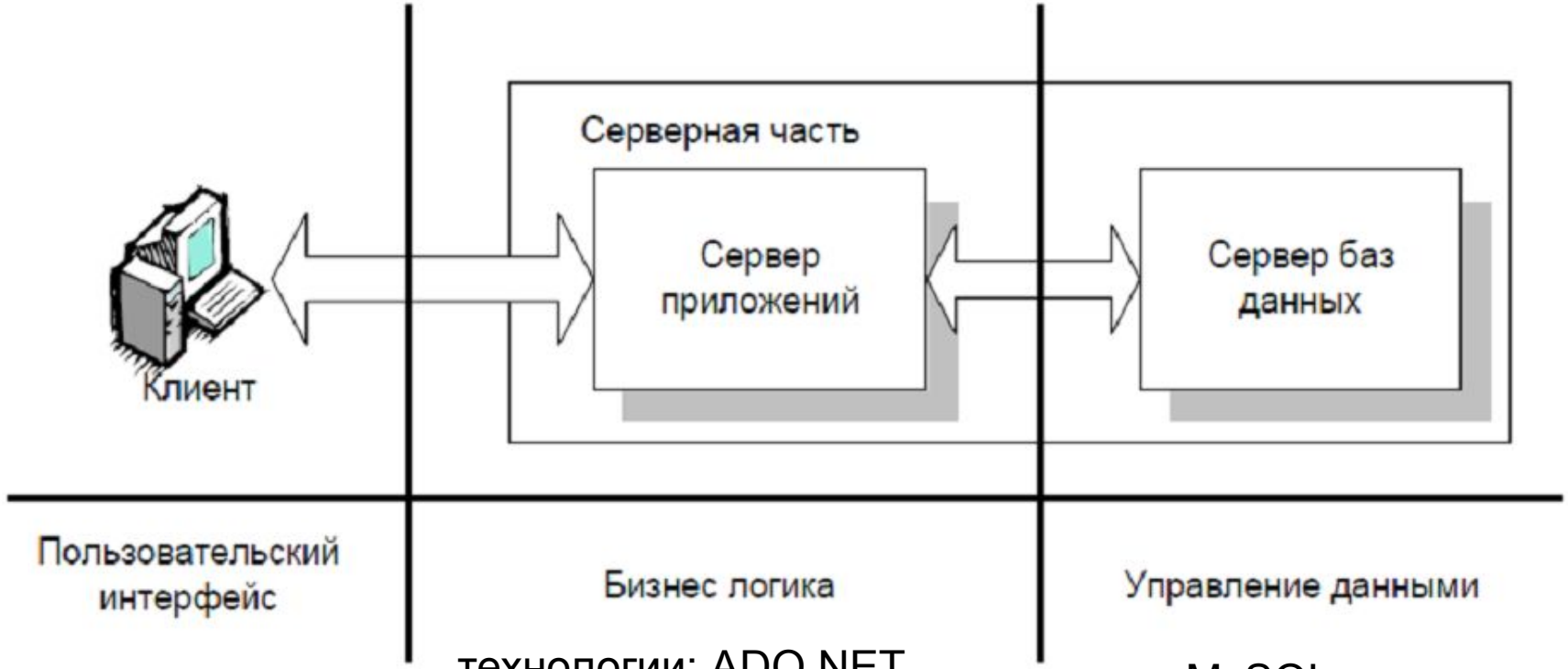
- Бизнес логика приложений осталась в клиентском ПО. При любом изменении алгоритмов, надо обновлять пользовательское ПО на каждом клиенте.
- Высокие требования к пропускной способности коммуникационных каналов с сервером, что препятствует использование клиентских станций иначе как в локальной сети.
- Слабая защита данных от взлома, в особенности от недобросовестных пользователей системы.
- Высокая сложность администрирования и настройки рабочих мест пользователей системы.
- Необходимость использовать мощные ПК на клиентских местах.
- Высокая сложность разработки системы из-за необходимости выполнять бизнес-логику и обеспечивать пользовательский интерфейс в одной программе.

# Многоуровневая архитектура

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



# Трехуровневая клиент-серверная архитектура




любой web-браузер.

технологии: ADO.NET, ASP.NET и web-сервером IIS

MySQL-сервер

Компоненты трехзвенной архитектуры, с точки зрения программного обеспечения реализуют определенные сервера БД, web-сервера и браузеры. Место любого из этих компонентов может занять программное обеспечение любого производителя.

**Браузер клиента 1-> Сервер IIS 2-> Исполняющая среда ASP.NET 2.0 3-> Провайдер данных ADO.NET 2.0 4-> Сервер MySQL 5-> Провайдер данных ADO.NET 2.0 6-> Исполняющая среда ASP.NET 2.0 7-> Сервер IIS 8-> Браузер клиента**

1 — браузер клиента отправляет HTTP-запрос;
2 — на стороне сервера служба Web Internet Information Server (web-сервер IIS) определяет тип запрашиваемого ресурса, и для случая запроса *.aspx (расширение файлов страниц ASP.NET) загружает соответствующее ему (запросу) расширение Internet Server Aplication Programming Interface (ISAPI). Для страниц aspx это расширение isapi_aspnet.dll. IIS также осуществляет идентификацию и авторизацию пользователя от которого поступил запрос. В свою очередь расширение isapi_aspnet.dll загружает фабрику обработчиков ASP.NET. Далее, фабрика обработчиков создает объектную модель запрашиваемой страницы и обрабатывает действия пользователя.
3 — в ходе генерации ответа приложению ASP.NET может потребоваться обращение к БД, в этом случае используя библиотеки классов провайдера данных ADO.NET 2.0, выполняющая среда обращается к серверу БД;
4 — провайдер данных ADO.NET 2.0 передает запрос на операцию с БД серверу MySQL;
5 — сервер MySQL осуществляет обработку запроса, выполняя соответствующие операции с БД ;
6 — провайдер данных ADO.NET 2.0 передает результаты запроса объекту страницы;
7 — объект страницы с учетом полученных данных осуществляет рендеринг графического интерфейса страницы и направляет результаты в выходной поток;
8 — сервер IIS отправляет содержимое сгенерированной страницы клиентскому браузеру.

# Трехуровневая клиент-серверная архитектура

**Плюсы:**

1. Тонкий клиент.
2. Между клиентской программой и сервером приложения передается лишь минимально необходимый поток данных - аргументы вызываемых функций и возвращаемые от них значения. Это теоретический предел эффективности использования линий связи, даже работа с ANSI-терминалами (не говоря уже об использование протокола http) требует большей нагрузки на сеть.
3. Сервер приложения ИС может быть запущен в одном или нескольких экземплярах на одном или нескольких компьютерах, что позволяет использовать вычислительные мощности организации столь эффективно и безопасно как этого пожелает администратор ИС.
4. Дешевый трафик между сервером приложений и СУБД. Трафик между сервером приложений и СУБД может быть большим, однако это всегда трафик локальной сети, а их пропускная способность достаточно велика и дешева. В крайнем случае, всегда можно запустить СП и СУБД на одной машине, что автоматически сведет сетевой трафик к нулю.
5. Снижение нагрузки на сервер данных по сравнению с 2.5-слойной схемой, а значит и повышение скорости работы системы в целом.
6. Дешевле наращивать функциональность и обновлять ПО.

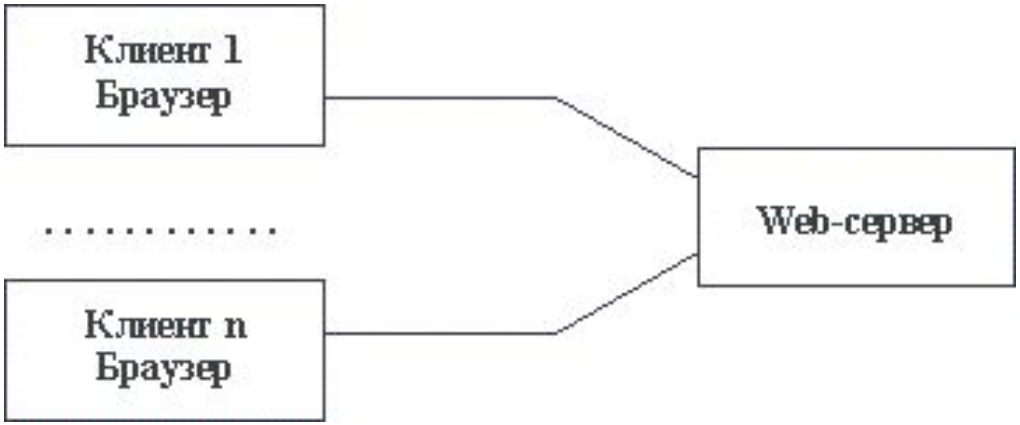
# Трехуровневая клиент-серверная архитектура

**Минусы:**

1. Выше расходы на администрирование и обслуживание серверной части.

# Архитектура Web-СУБД

## Intranet-приложения




Простая организация Intranet-системы с использованием средств WWW

при применении Web-технологии существует 2 подхода в ее реализации на стороне Web-сервера:

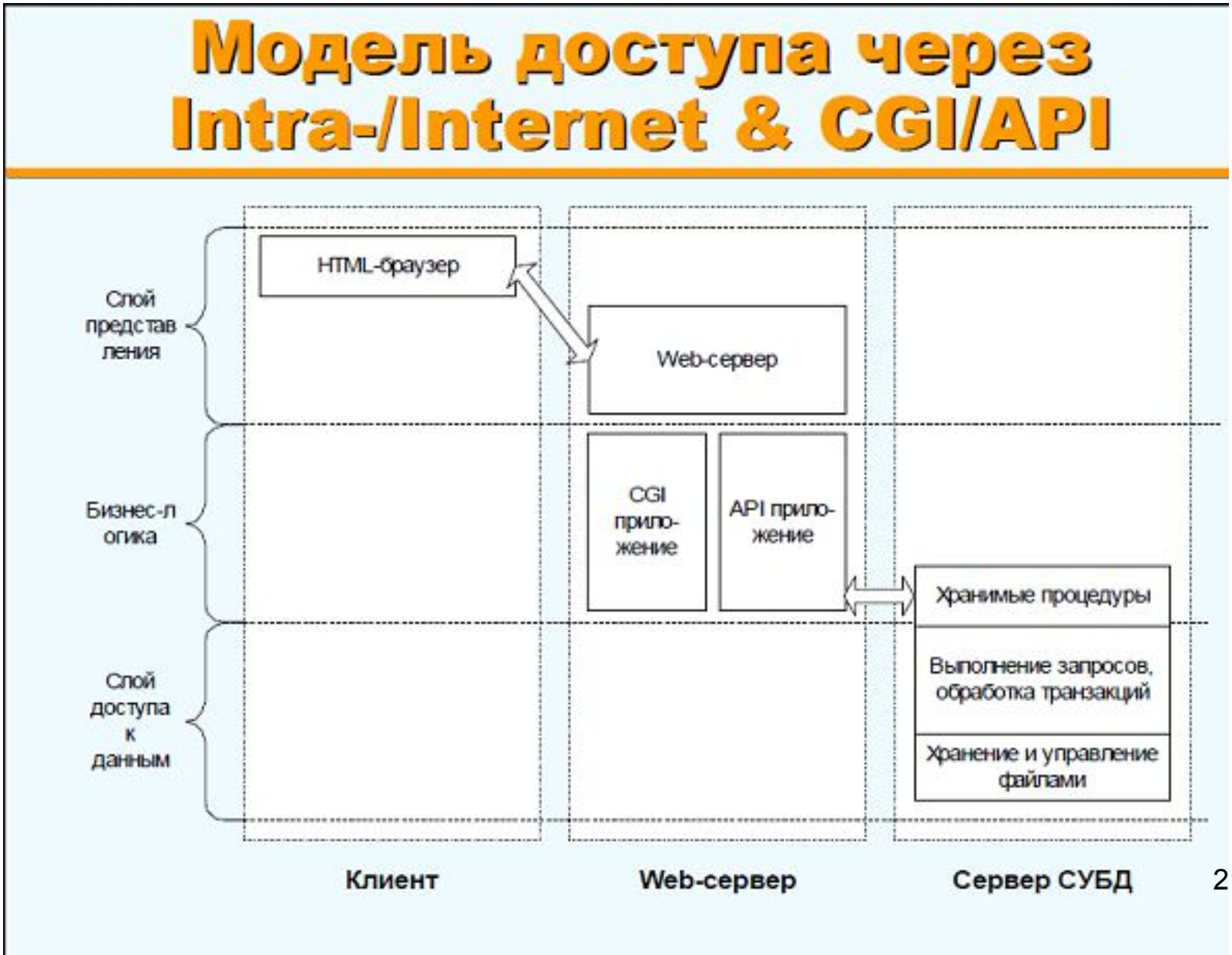
✔*CGI (Common Gateway Interface)* – внешняя программа выполняется в отдельном адресном пространстве

✔*API (Application Programming Interface)* – внешние процедуры компонуются совместно со стандартной частью Web-сервера

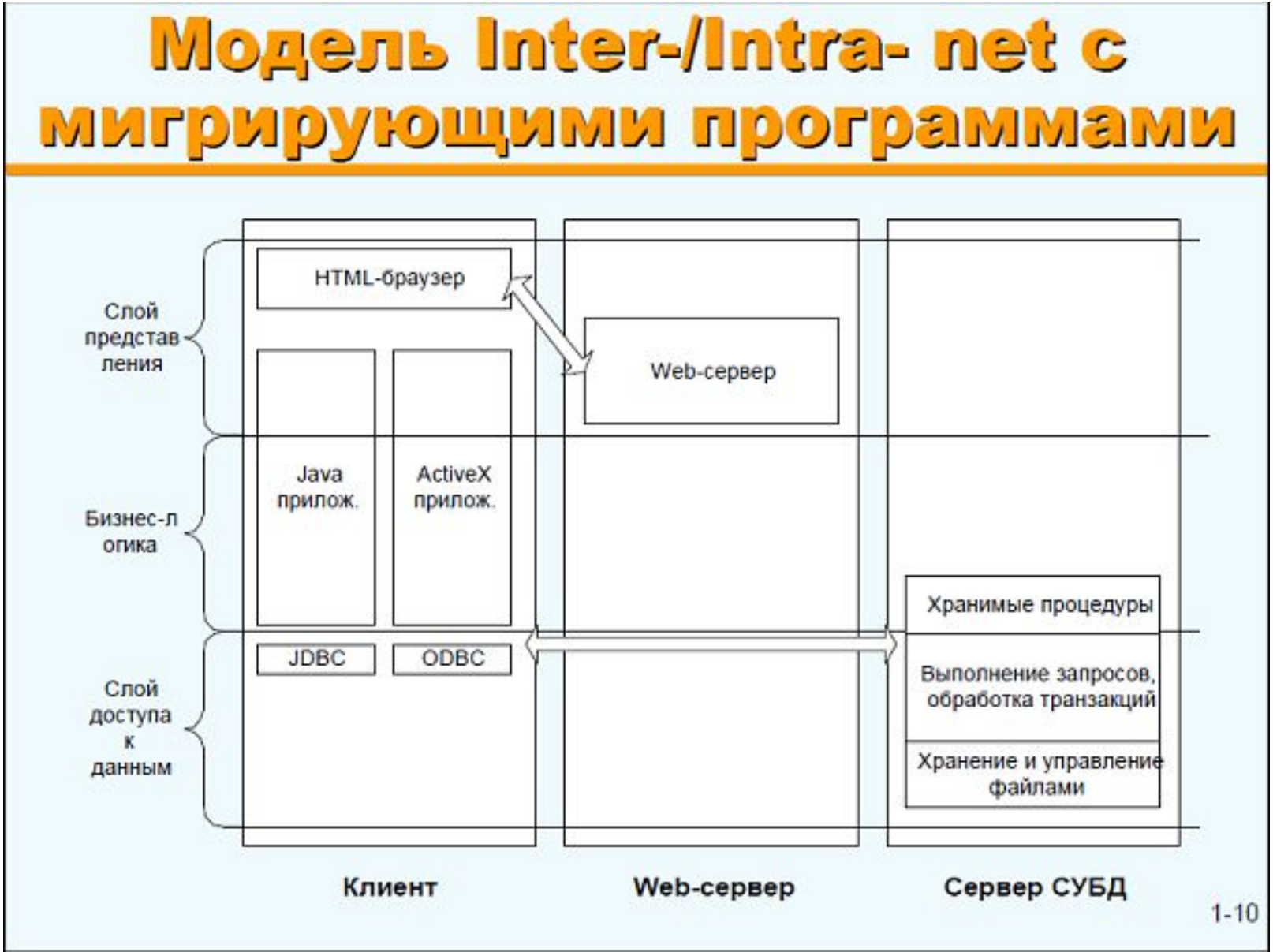

Вызов внешней процедуры Web-сервера


Доступ к базе данных
в Intranet-системе

# Internet/Intranet - технологии



# Архитектура на основе Internet/Intranet с мигрирующими программами



# Распределенные информационные системы

**Распределенная система** — это набор независимых вычислительных машин, представляющийся их пользователям единой объединенной системой.

**Характеристики распределенных систем:**

1. От пользователей скрыты различия между компьютерами и способы связи между ними. То же самое относится и к внешней организации распределенных систем.
2. Пользователи и приложения единообразно работают в распределенных системах, независимо от того, где и когда происходит их взаимодействие.
3. Распределенные системы должны также относительно легко поддаваться расширению, или масштабированию.

# система промежуточного уровня (middleware)


Распределенная система организована в виде службы промежуточного уровня.

# Особенности распределенных ИС

- Ссылки
- Задержки выполнения запросов
- Активация/деактивация
- Постоянное хранение
- Параллельное исполнение
- Отказы
- Безопасность

# Ссылки

Ссылки на объекты в программных модулях на ОО языках программирования (например, C++) являются указателями в памяти.

1. Ссылки на объекты в распределенных системах в противоположность являются более комплексными:
1.1.Содержат информацию о размещении
1.2.Информацию о безопасности
1.3.Ссылки на объектные типы

2. Ссылки на распределенные объекты значительно больше (40 байт для Orbix)

# Задержки выполнения запросов

Локальные вызовы требуют порядка пары сотен наносекунд

Запрос к объекту требует от 0.1 до 10 миллисекунд

Интерфейсы в распределенной системе должны быть спроектированы так, чтобы снизить время выполнения запросов:

1. Снизить частоту обращения;
2. Укрупнить выполняемые функции.

# Активация/деактивация

Объекты в ОО языках находятся в виртуальной памяти от создания до уничтожения

В распределенных системах
1. Больше объектов
2. Объекты могут не использоваться на протяжении долгого времени

Реализации распределенных объектов
1. Переносятся в память при активации
2. Удаляются из памяти при деактивации

# Постоянное хранение

Объекты могут иметь или не иметь состояние.

Объекты имеющие состояние должны сохранять его на постоянный носитель между:

1. Деактивацией объекта
2. Активацией объекта

Может быть достигнуто:

1. Записью в файловую систему
2. Отражением на реляционные БД
3. С помощью объектных БД

# Параллельное исполнение

В нераспределенных системах исполнение в основном последовательное, иногда конкурентное в разных нитях процессов.

Распределенные компоненты выполняются параллельно, что приводит к необходимости согласования выполнения.

# Отказы

Запросы в распределенных системах имеют большую вероятность отказов

Клиенты обязаны проверять факт выполнения запросов сервером

# Безопасность

Безопасность в ОО приложениях может выполняться на основе контроля сеансов.

При работе распределенных систем возникают вопросы безопасности:

1. Кто запрашивает выполнение операции?
2. Как мы можем удостовериться, что субъект является именно тем за кого он себя
выдает?
3. Как мы примем решение предоставлять или нет субъекту право на выполнение
сервиса?
4. Как мы можем неопровержимо доказать, что сервис был предоставлен?