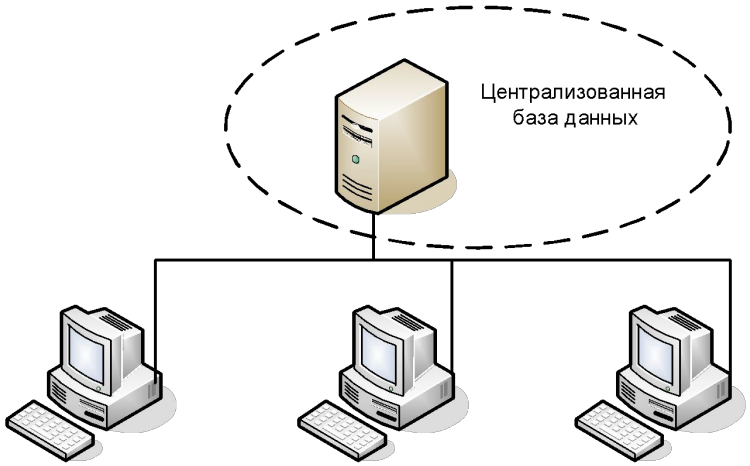
Основы проектирования баз данных

Распределенная обработка данных

Система распределенной обработки данных

Режим распределенного доступа к централизованной БД:

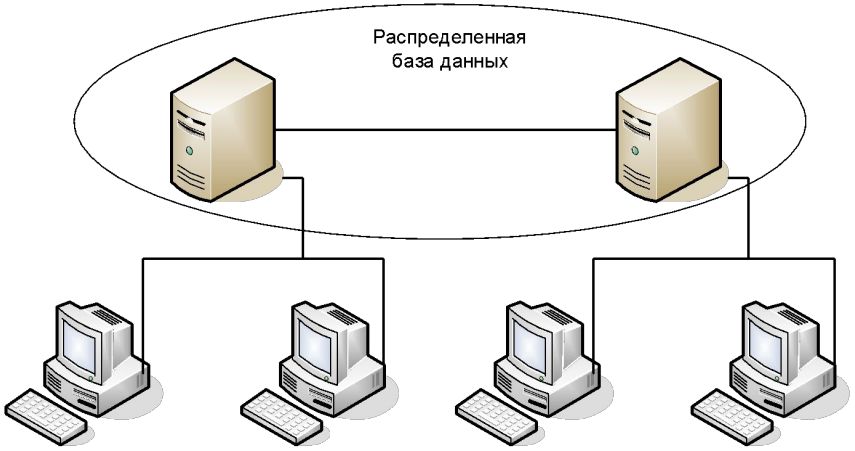
- параллельный доступ к одной базе данных нескольких пользователей;
- база данных физически расположена на одной машине.



Система распределенных баз данных

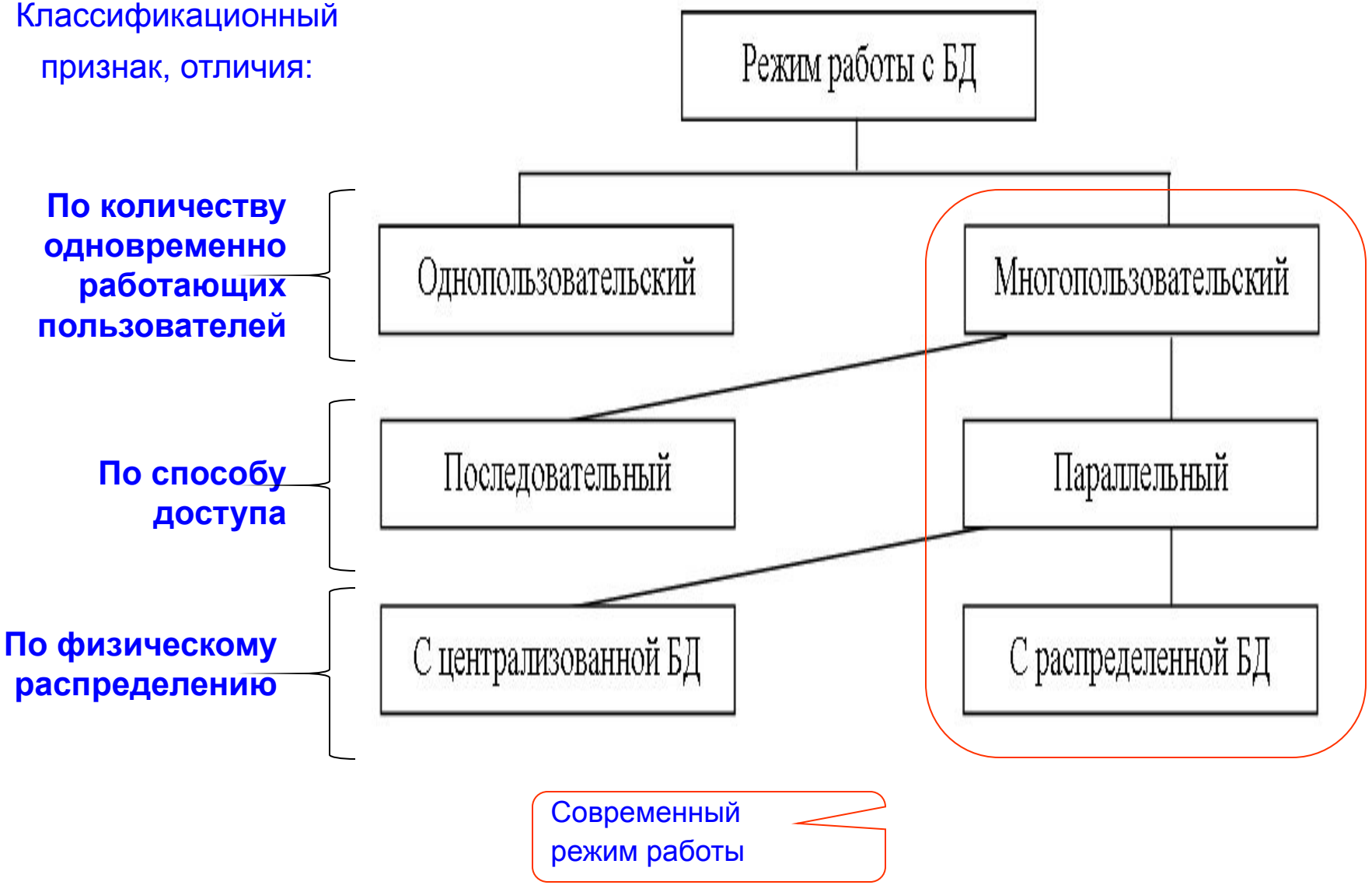
Режим параллельного доступа к распределенной БД:

- база данных физически распределена по нескольким компьютерам, расположенным в сети;
- к базе данных возможен параллельный доступ нескольких пользователей.



Режимы работы с базами данных

Классификационный признак, отличия:



Модели «клиент-сервер» в технологии баз данных

Основные программные процессы:

- **«Клиент»**
  сторона, запрашивающая функции/обслуживание

- **«Сервер»**
  сторона, предоставляющая функции/обслуживание

Особенность

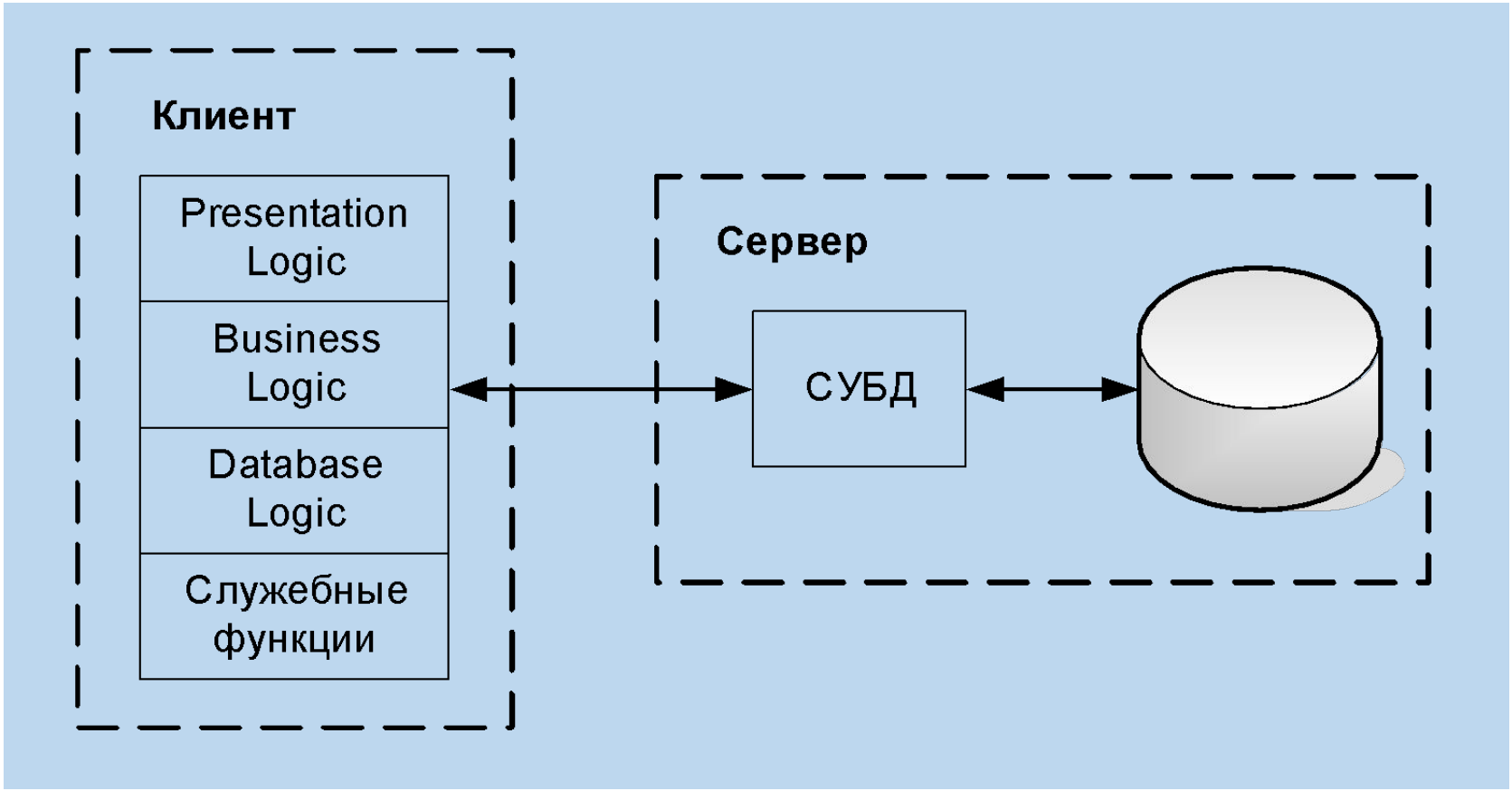
На уровне программного обеспечения разделение системы на клиента и сервер является **логическим**, а именно процессы клиента и сервера могут **физически** размещаться как на одной, так и на разных машинах.

Основной принцип технологии «клиент–сервер» применительно к технологии баз данных – разделение функций приложения на группы

Основные функции стандартного интерактивного приложения:

1) функции ввода и отображения данных (Интерфейс пользователя, **Presentation Logic**, Логика представления);
2) прикладные функции, определяющие основные алгоритмы решения задач приложения (**Business Logic**, Бизнес-правила);
3) функции обработки данных внутри приложения (**Database Logic**);
4) функции управления информационными ресурсами (**Database Manager System**);
5) служебные функции, играющие роль связок между функциями первых четырех групп.

Структура типового интерактивного приложения, работающего с базой данных



Презентационная логика

(Presentation Logic)

Это часть кода приложения, которая определяет ***что*** пользователь видит на своем экране, когда работает приложение.

Основные задачи:

- формирование экранных изображений;
- чтение и запись в экранные формы информации;
- управление экраном;
- обработка движений мыши и нажатие клавиш клавиатуры.

Бизнес-логика, или логика собственно приложений

(Business Processing Logic)

Это часть кода приложения, которая определяет алгоритмы решения конкретных задач приложения.

В зависимости от конкретных функциональных требований и сложности задач может оказаться полезным подразделить эту часть на несколько компонентов (бизнес-правила, правила целостности и др.).

Реализация

Конкретная реализация каждого компонента может быть представлена в виде набора процедур (библиотек), класса или классов объектов, отдельных программ.

Как правило, код компонентов пишут с использованием различных языков программирования, таких как C, C++, C#, Pascal, Visual Basic и др.

Логика обработки данных, или логика доступа к данным

(Database Logic)

Это часть кода приложения, которая связана с обработкой данных внутри приложения.

Назначение

Осуществляет перевод специфических для конкретного приложения запросов на язык SQL (интерфейс к СУБД), получение результатов и перевод этих результатов обратно в специфические для конкретного приложения структуры данных.

Процессор управления данными (Database Manager System Processing)

Это собственно СУБД, которая обеспечивает хранение и управление базами данных.

В идеале функции СУБД должны быть скрыты от бизнес-логики приложения, однако для рассмотрения архитектуры приложения их выделяют в отдельную часть приложения.

Варианты архитектуры приложения

Централизованная

Все части приложения располагаются в единой среде и комбинируются внутри одной исполняемой программы.

Децентрализованная

Все задачи могут быть по-разному распределены между серверным и клиентским процессами.

Различие архитектурных реализаций определяется тем:

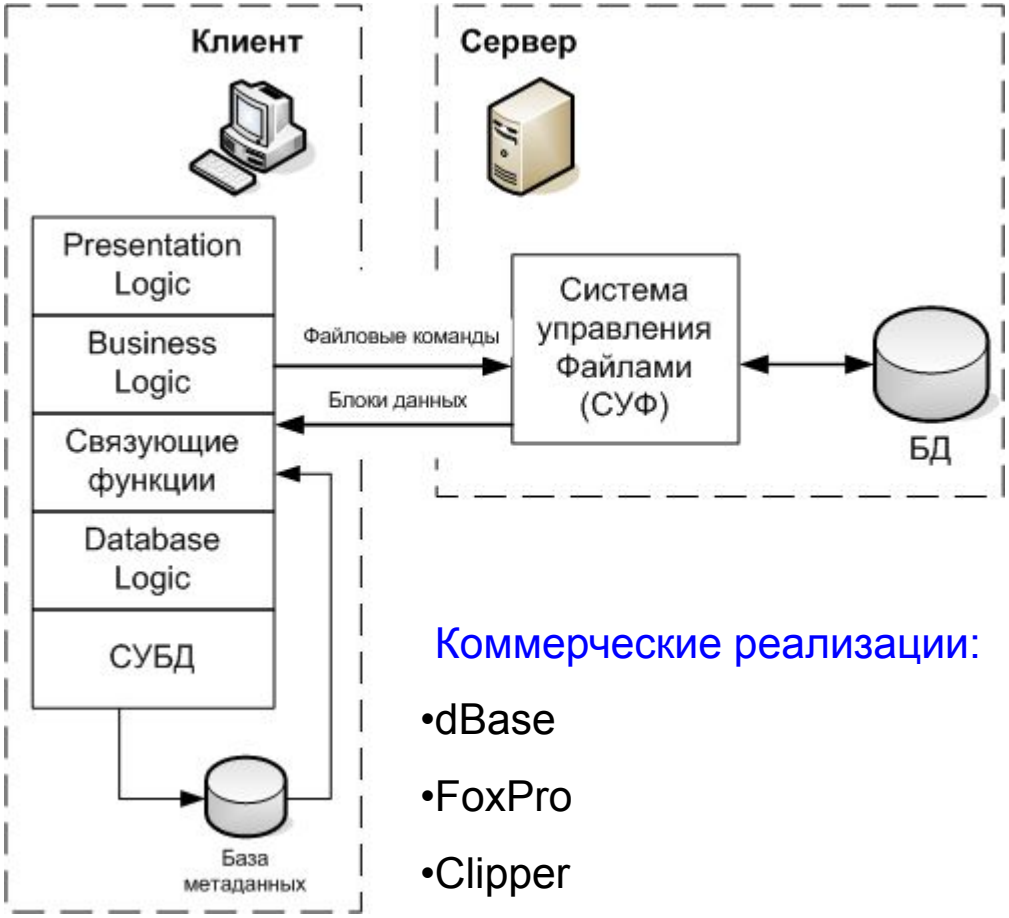
- как логические компоненты базы данных распределены в сети;
- какие механизмы используются для связи компонентов между собой.

Варианты (модели) построения архитектуры приложения:

- **двухуровневые**
  - модель файлового сервера (модель удаленного управления данными);
  - модель удаленного доступа к данным;
  - модель сервера баз данных;
- **трехуровневая**
  - модель сервера приложений.

Двухуровневые модели

Модель файлового сервера (File Server, FS) (модель удаленного управления данными)



Коммерческие реализации:

- dBase
- FoxPro
- Clipper

Преимущества

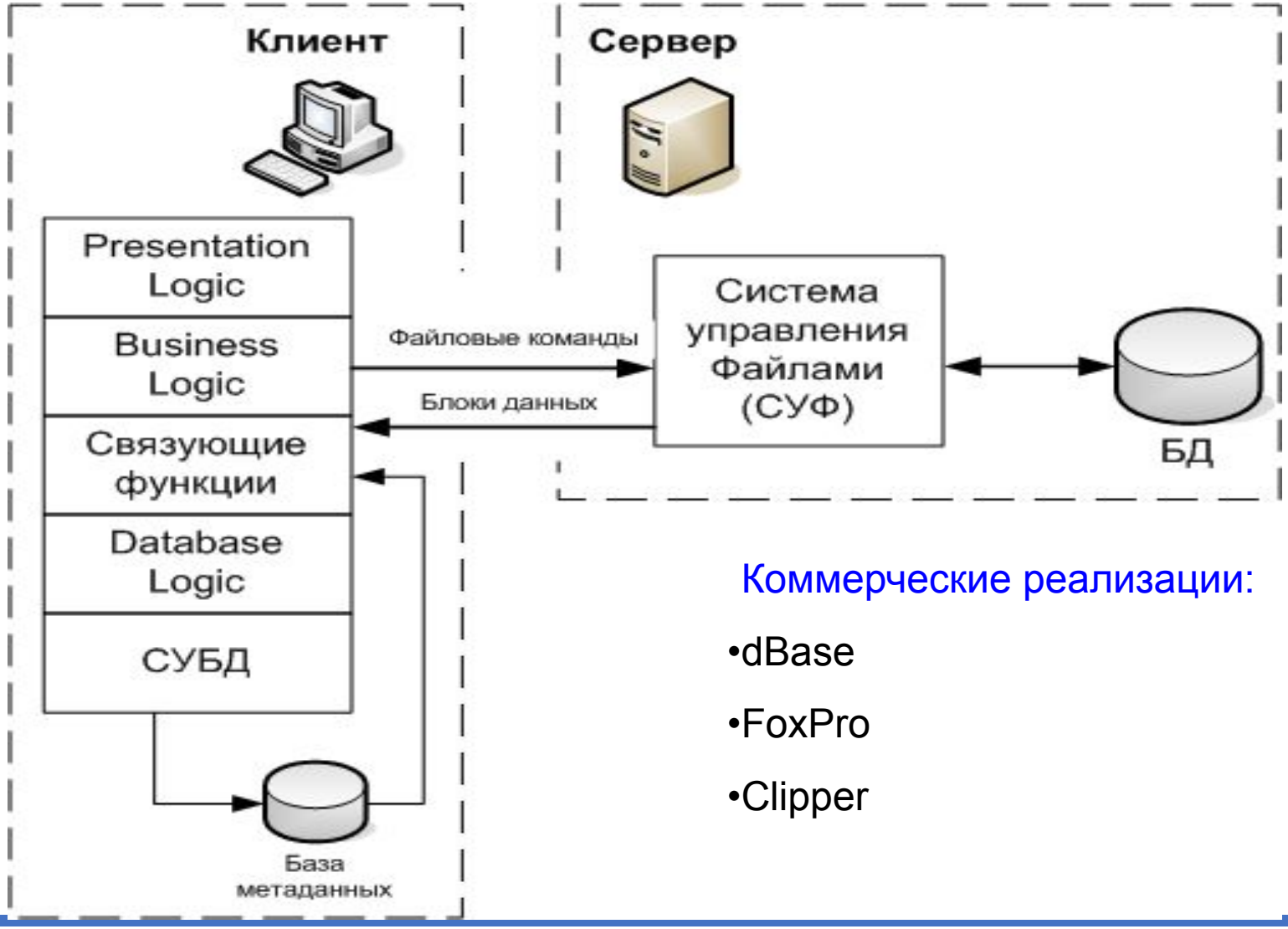
- разделение монопольного приложения на два взаимодействующих процесса;
- сервер (серверный процесс) может обслуживать множество клиентов

Недостатки

- высокий сетевой трафик, который связан с передачей по сети множества блоков и файлов, необходимых приложению;
- узкий спектр операций манипулирования с данными, который определяется только файловыми командами. Драйвер «умеет» обрабатывать только простые запросы;
- отсутствуют механизмы оптимизаций запросов и кэширования;
- отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы).

Двухуровневые модели

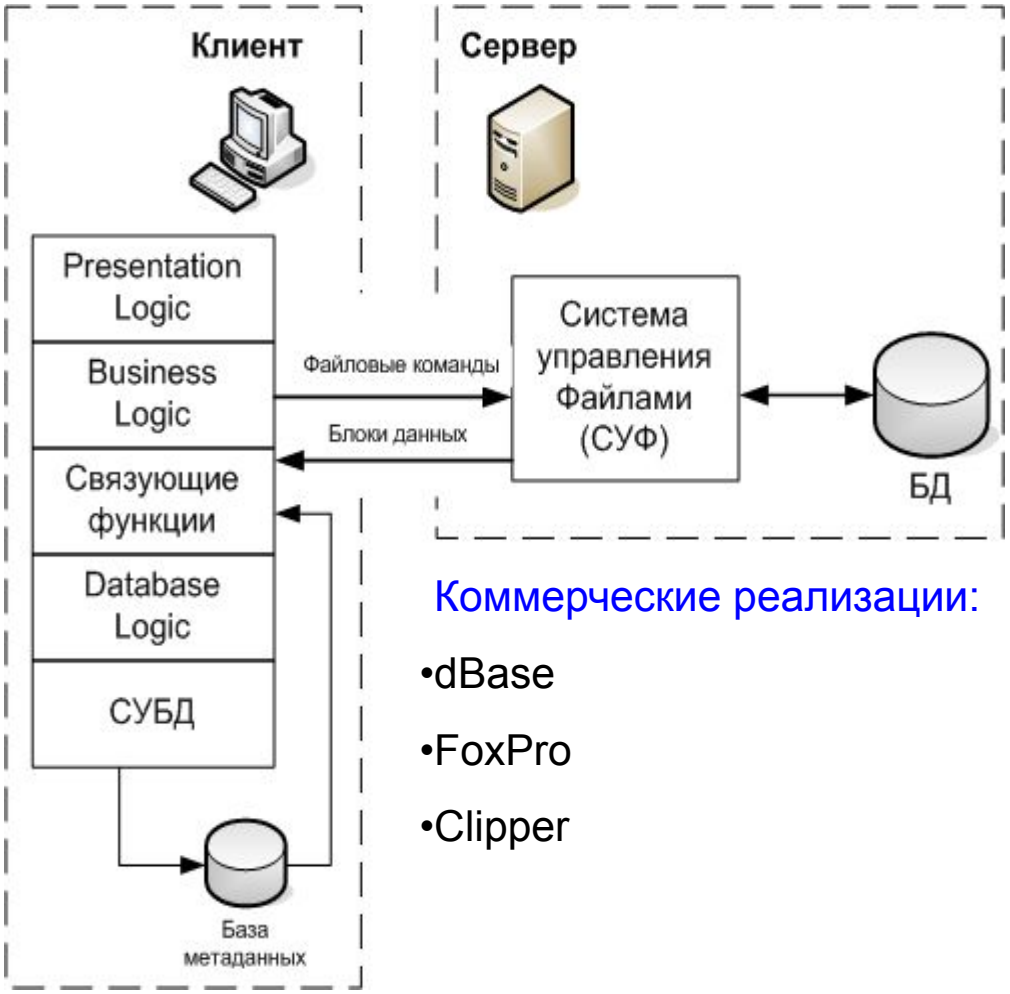
Модель файлового сервера (File Server, FS) (модель удаленного управления данными)



Коммерческие реализации:

- dBase
- FoxPro
- Clipper

Модель файлового сервера (File Server, FS) (модель удаленного управления данными)



Коммерческие реализации:

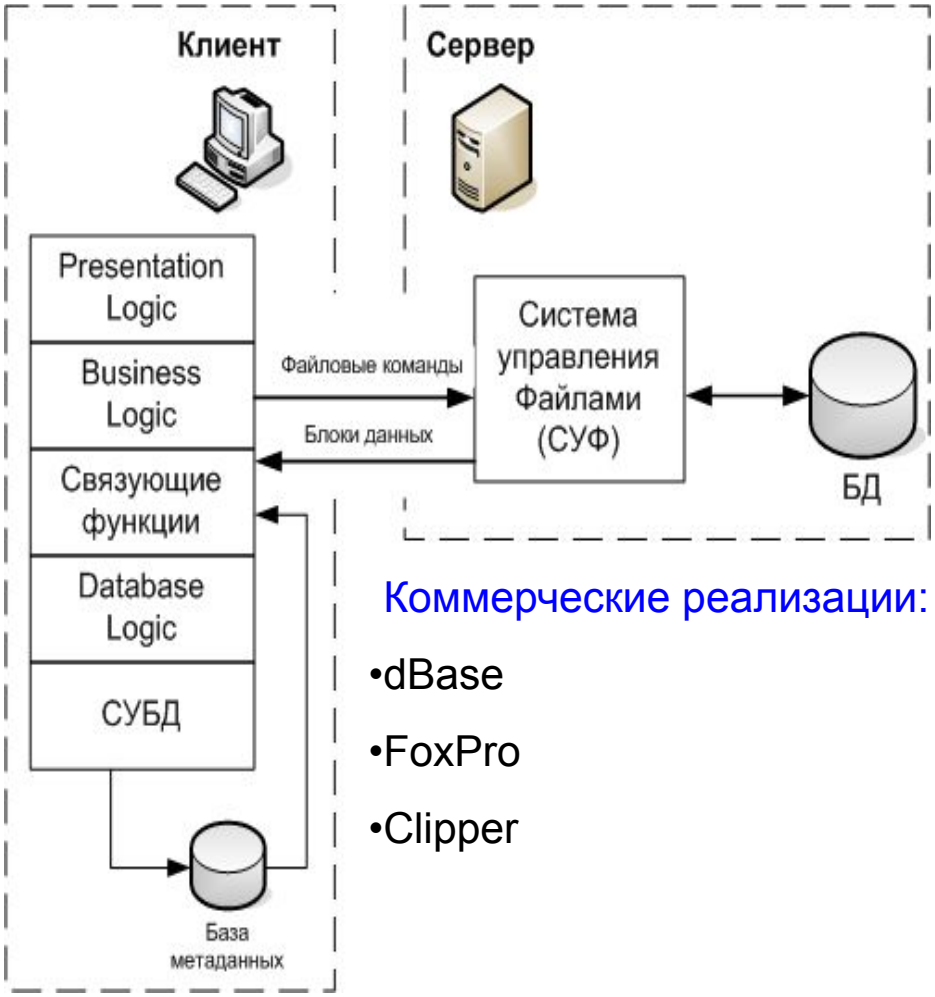
- dBase
- FoxPro
- Clipper

Преимущества

- разделение монопольного приложения на два взаимодействующих процесса;
- сервер (серверный процесс) может обслуживать множество клиентов

Двухуровневые модели

Модель файлового сервера (File Server, FS
(модель удаленного управления данным



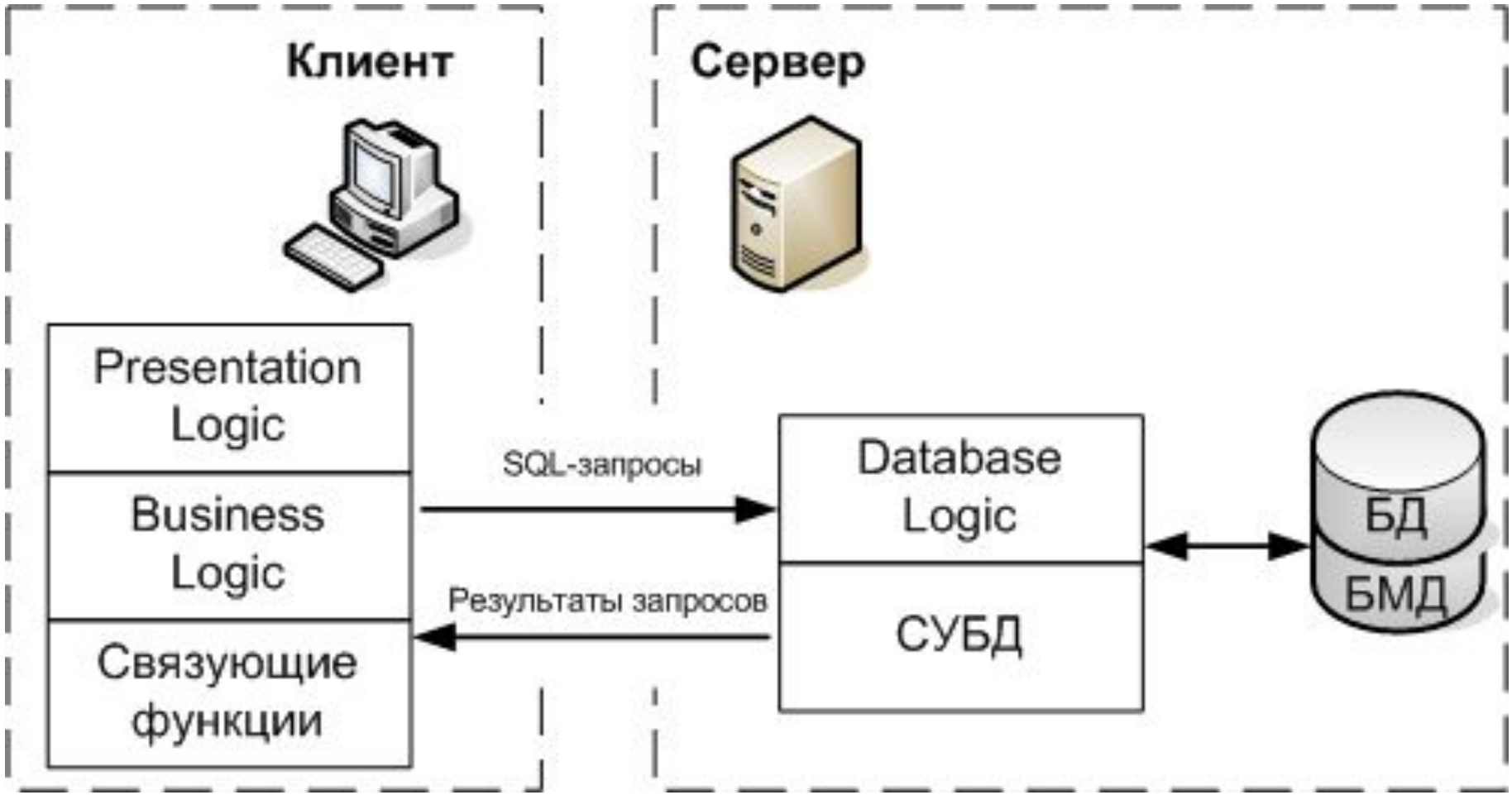
Коммерческие реализации:

- dBase
- FoxPro
- Clipper

Недостатки

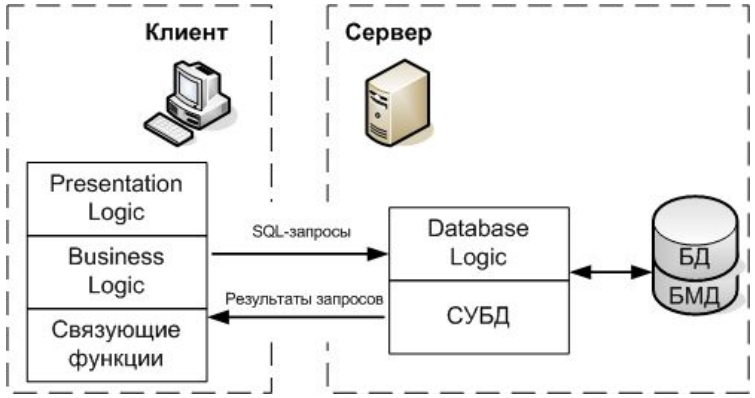
- высокий сетевой трафик, который связан с передачей по сети множества блоков и файлов, необходимых приложению;
- узкий спектр операций манипулирования с данными, который определяется только файловыми командами. Драйвер «умеет» обрабатывать только простые запросы;
- отсутствуют механизмы оптимизаций запросов и кэширования;
- отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы).

Модель удаленного доступа к данным (Remote Data Access, RDA)



Коммерческие реализации: Microsoft Access

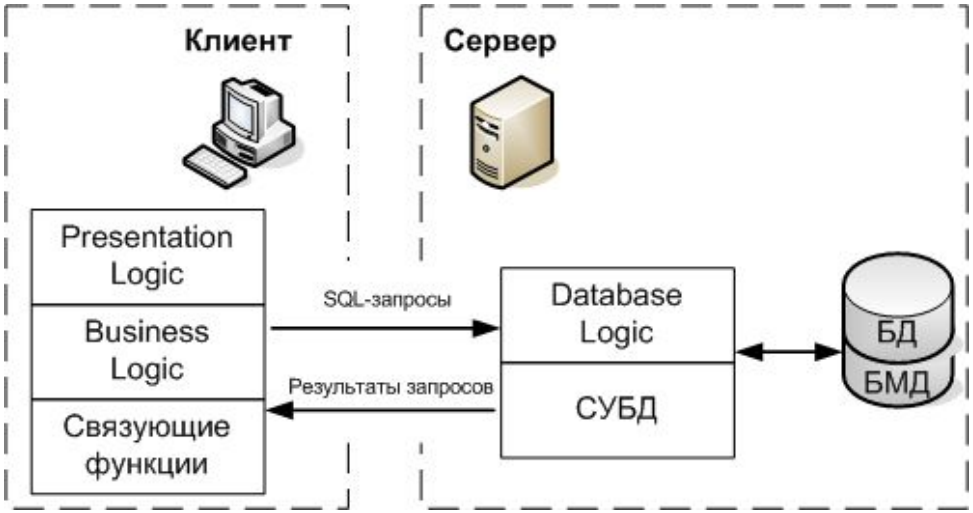
Модель удаленного доступа к данным (Remote Data Access, RDA)



Недостатки

- запросы на языке SQL при интенсивной работе клиентских приложений могут существенно загрузить сеть;
- сложность администрирования при изменении бизнес-правил, вызванная излишним дублированием кода приложений;
- сервер играет пассивную роль, поэтому функции управления информационными ресурсами должен выполнять клиент.

Модель удаленного доступа к данным (Remote Data Access, RDA)



Преимущества

- резко уменьшается загрузка сети, так как по ней от клиентов к серверу передаются не запросы на ввод-вывод в файловой терминологии, а запросы на SQL, и их объем существенно меньше. В ответ на запросы клиент получает только данные, релевантные (соответствующие смыслу) запросу, а не блоки файлов, как в FS-модели
- более гибкое распределение доступа к данным (на уровне отдельных записей);
- унификация (стандартизация) интерфейса «клиент-сервер», стандартом при общении приложения-клиента и сервера становится язык SQL.

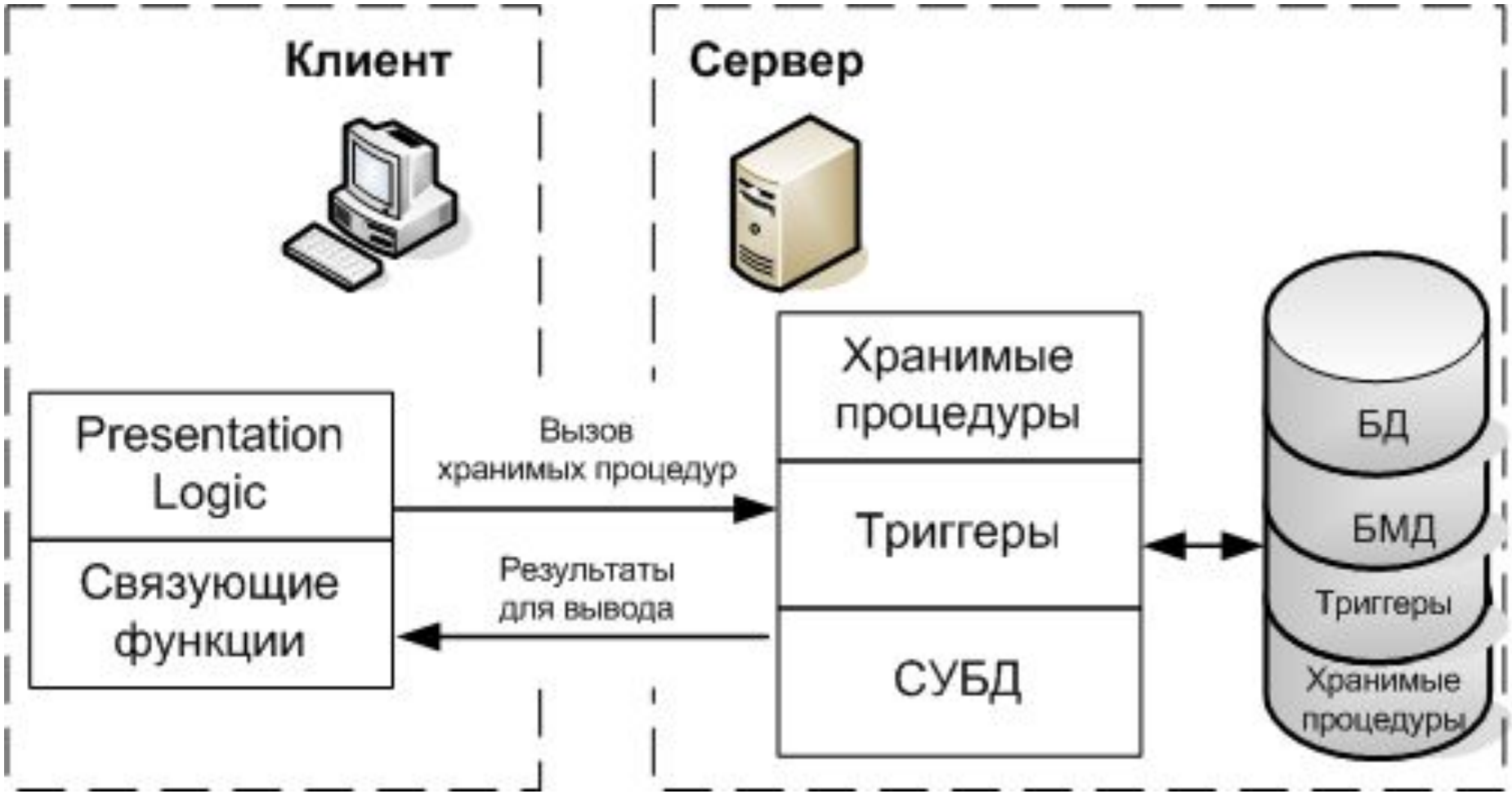
Требования к серверу

- База данных должна в каждый момент отражать текущее состояние предметной области. Непротиворечивость хранимых в базе данных
- База данных должна отражать некоторые общие правила предметной области (бизнес-правила)
- Необходим постоянный контроль за состоянием базы данных, отслеживание всех изменений и адекватная реакция на них
- Необходимо, чтобы возникновение некоторой ситуации в базе данных четко и оперативно влияло на ход выполнения прикладной задачи.
- Необходимо контролировать в базе данных семантическую составляющую бизнес-правил.

Механизмы на сервере для реализации требований

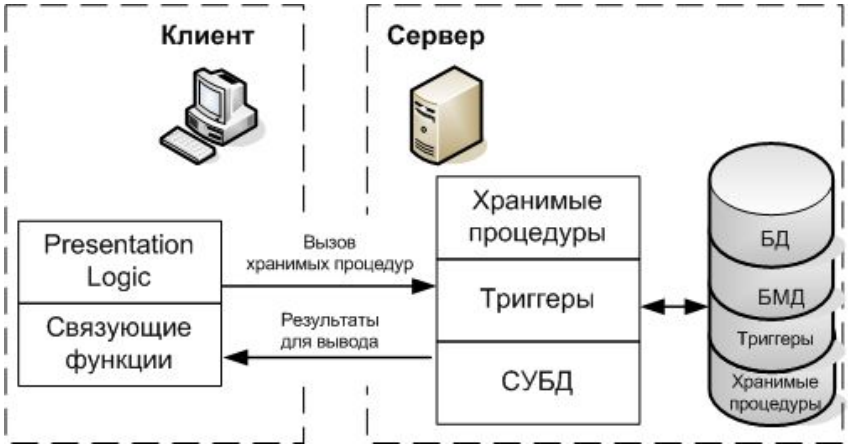
- Хранимые процедуры (Stored Procedure)
- Триггеры (Trigger)

Модель сервера баз данных (Server DataBase)



Коммерческие реализации: Oracle, Microsoft SQL Server

Модель сервера баз данных (Server DataBase)



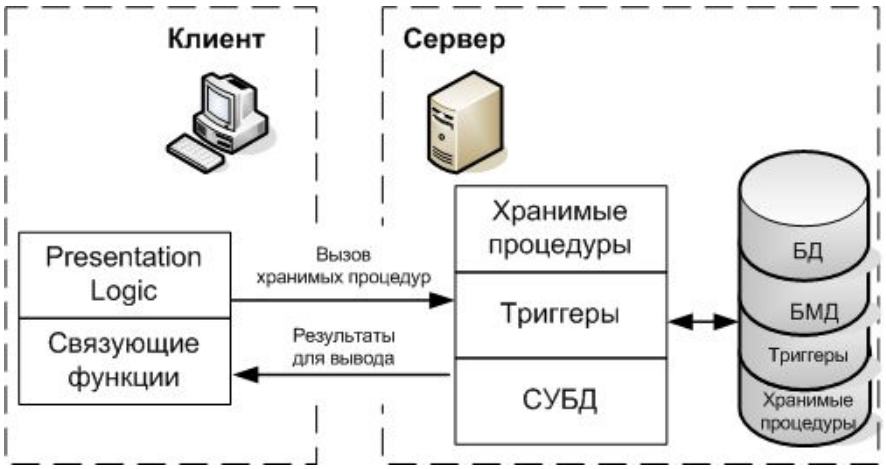
Коммерческие реализации:

- Oracle
- Microsoft SQL Server

Преимущества

- снижение сетевого трафика;
- реализация общей для клиентов бизнес-логики средствами сервера, что существенно уменьшает дублирование алгоритмов обработки данных в разных клиентских приложениях;
- упрощение администрирования сервера БД (наличие встроенных механизмов решения административных задач по обслуживанию сервера, например репликации данных между серверами, автоматического запуска задач, оповещения и др.).

Модель сервера баз данных (Server DataBase)



Коммерческие реализации:

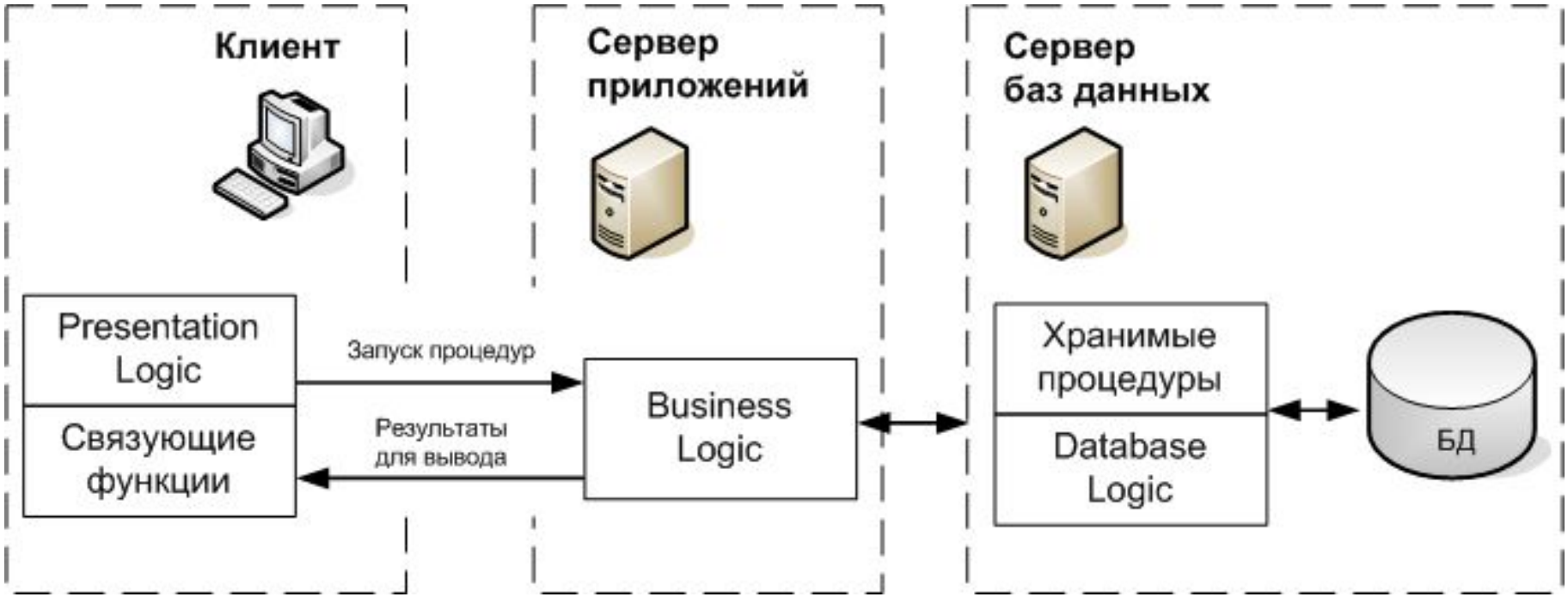
- Oracle
- Microsoft SQL Server

Недостатки

- очень большая загрузка сервера;
- в настоящее время в коммерческих СУБД нет удобного инструмента для написания и отладки хранимых процедур и триггеров;
- отсутствие стандартов на хранимые процедуры.

Трехуровневая модель

Модель сервера приложений (Application Server, AS)

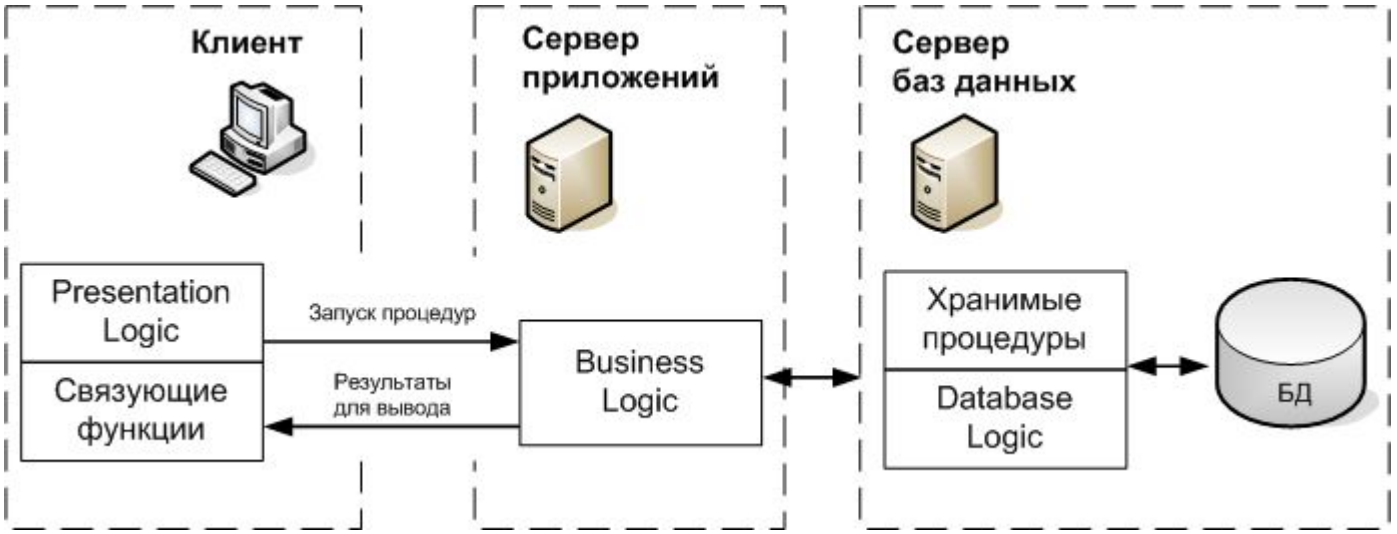


Модель сервера приложений (Application Server, AS)

Преимущества

- преодоление фундаментальных ограничений двухуровневой архитектуры по количеству одновременно подключенных клиентов;
- модель обладает б***о***льшей гибкостью, чем двухуровневые модели. Наиболее заметны преимущества в случаях выполнения сложных аналитические расчетов над базой данных (OLAP, On-line analytical processing);
- повышение переносимости системы и ее масштабируемости, поскольку б***о***льшая часть бизнес-логики клиента изолирована от возможностей встроенного SQL, реализованного в конкретной СУБД, и может быть выполнена на стандартных языках программирования.

Модель сервера приложений (Application Server, AS)



Недостаток

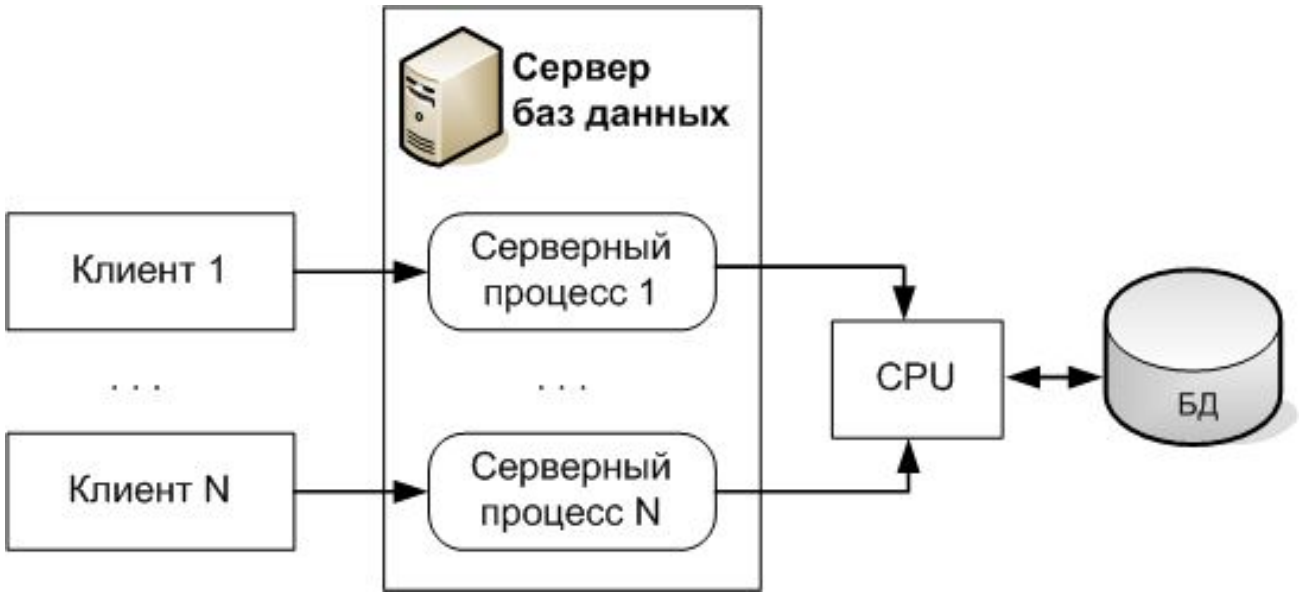
- сложность программирования системы баз данных

Модели серверов баз данных

Архитектуры построения серверов баз данных

- Модель «один-к-одному»
- Модель системы с выделенным сервером
- Модель виртуального сервера
- Модель многопотоковой архитектуры с несколькими серверами

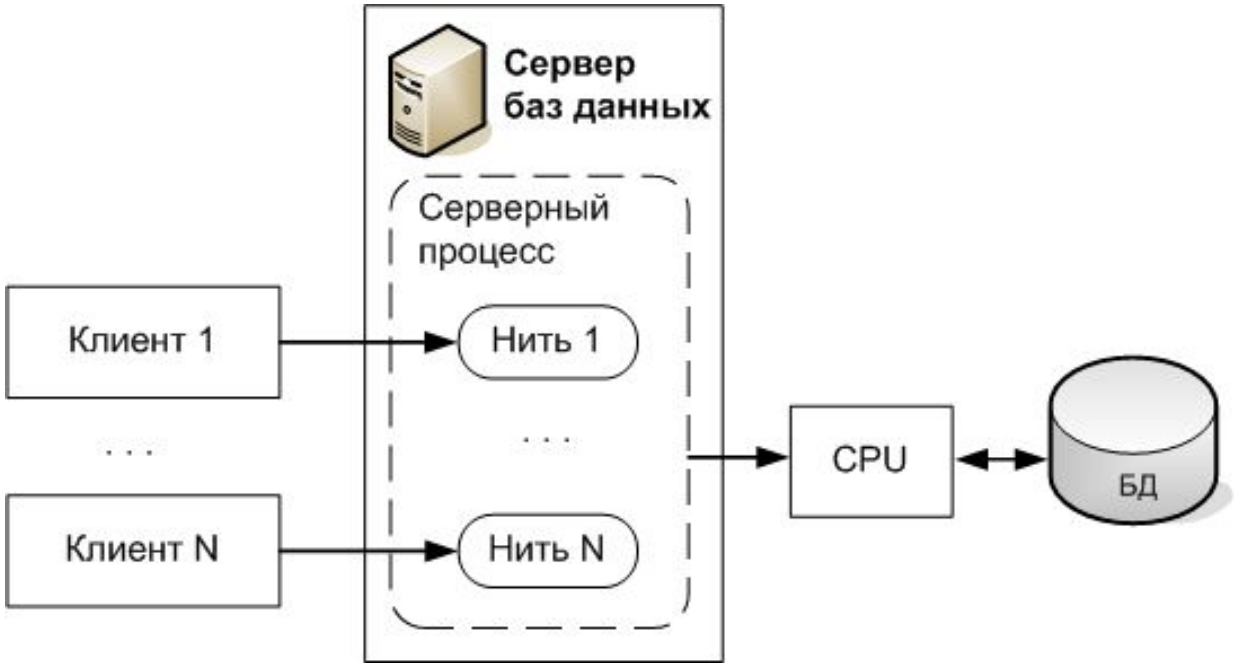
Модель «один-к-одному»



Особенности

- сервер обслуживает запросы только одного пользователя (клиента),
- для обслуживания нескольких клиентов нужно было запустить эквивалентное число серверов (серверных процессов в операционной системе)
- каждый серверный процесс в этой модели запускается как независимый, поэтому если один клиент сформировал запрос, который был только что выполнен другим серверным процессом для другого клиента, то запрос тем не менее выполнялся повторно.

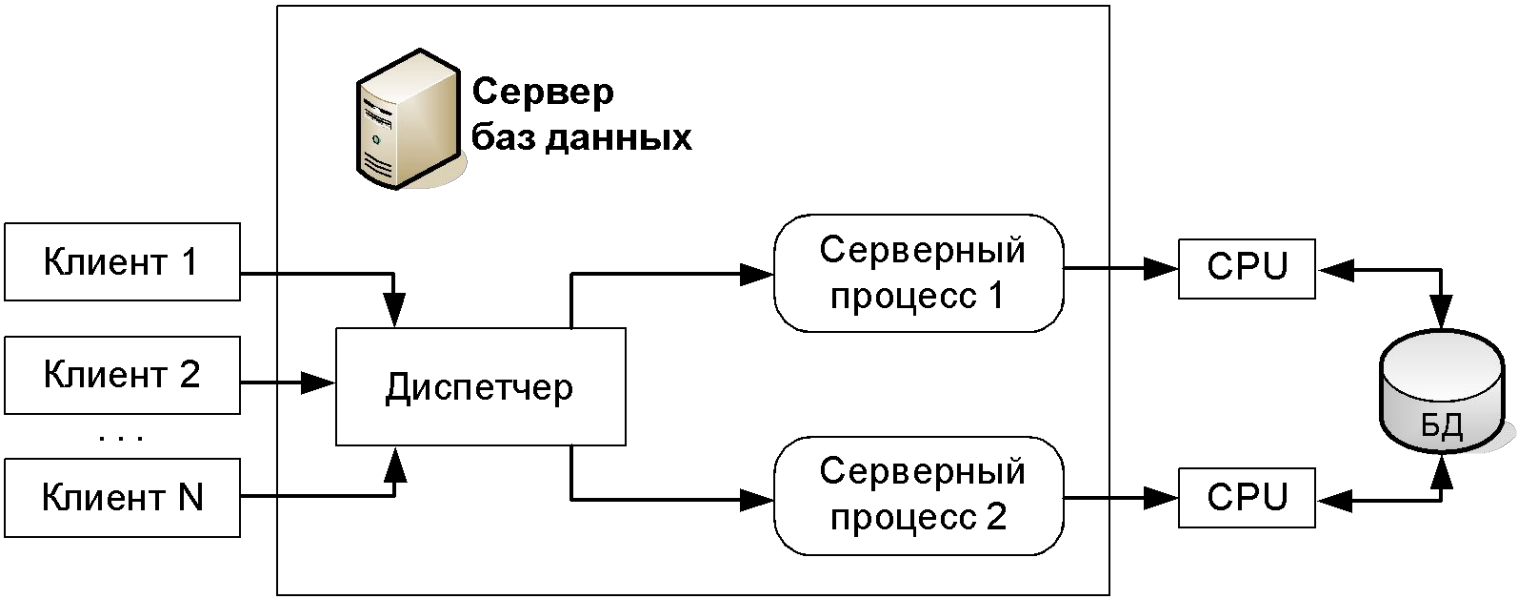
Система в выделенным сервером (многопотоковая односерверная архитектура, multi-threaded)



Особенности

- единственный серверный процесс обладает монополией на управление данными и взаимодействует одновременно со многими клиентами;
- логически каждый клиент связан с сервером отдельной нитью («thread»), или потоком, по которому пересылаются запросы.

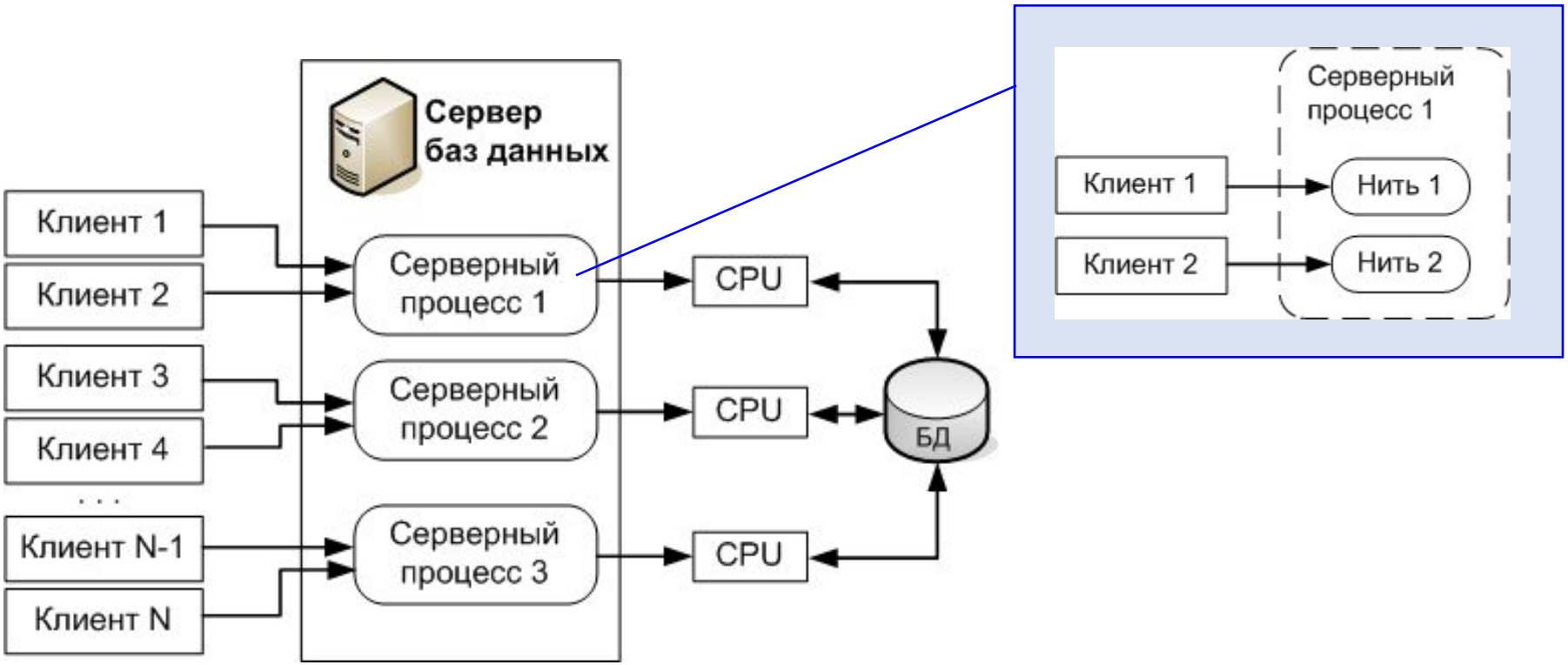
Модель виртуального сервера (virtual server)



Особенности

- клиенты подключаются не к реальному серверу, а к промежуточному звену, называемому диспетчером, который выполняет только функции диспетчеризации запросов к актуальным серверам;
- нет ограничений на использование многопроцессорных платформ: количество актуальных серверов может быть согласовано с количеством процессоров в системе.

Модель многопотоковой архитектуры с несколькими серверами (многонитиевая мультисерверная архитектура)



Особенности

- возможности запуска нескольких серверов базы данных, в том числе и на различных процессорах;
- каждый из серверов должен быть многопотоковым.

Распараллеливание выполнения запросов

Цель

Повысить производительность работы серверов баз данных за счет разбиения пользовательского запроса на ряд подзапросов, которые могут выполняться параллельно, а результаты их выполнения потом объединить
в общий результат выполнения запроса.

Способы

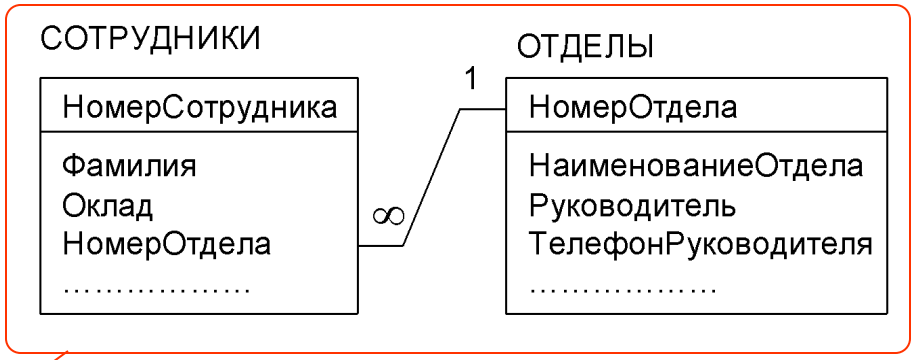
- выполнение подзапросов отдельными серверными процессами (нитями, threads)
- фрагментация выполнения запросов (параллелизм)

Виды параллелизма

- вертикальный
- горизонтальный

Распараллеливание выполнения запросов

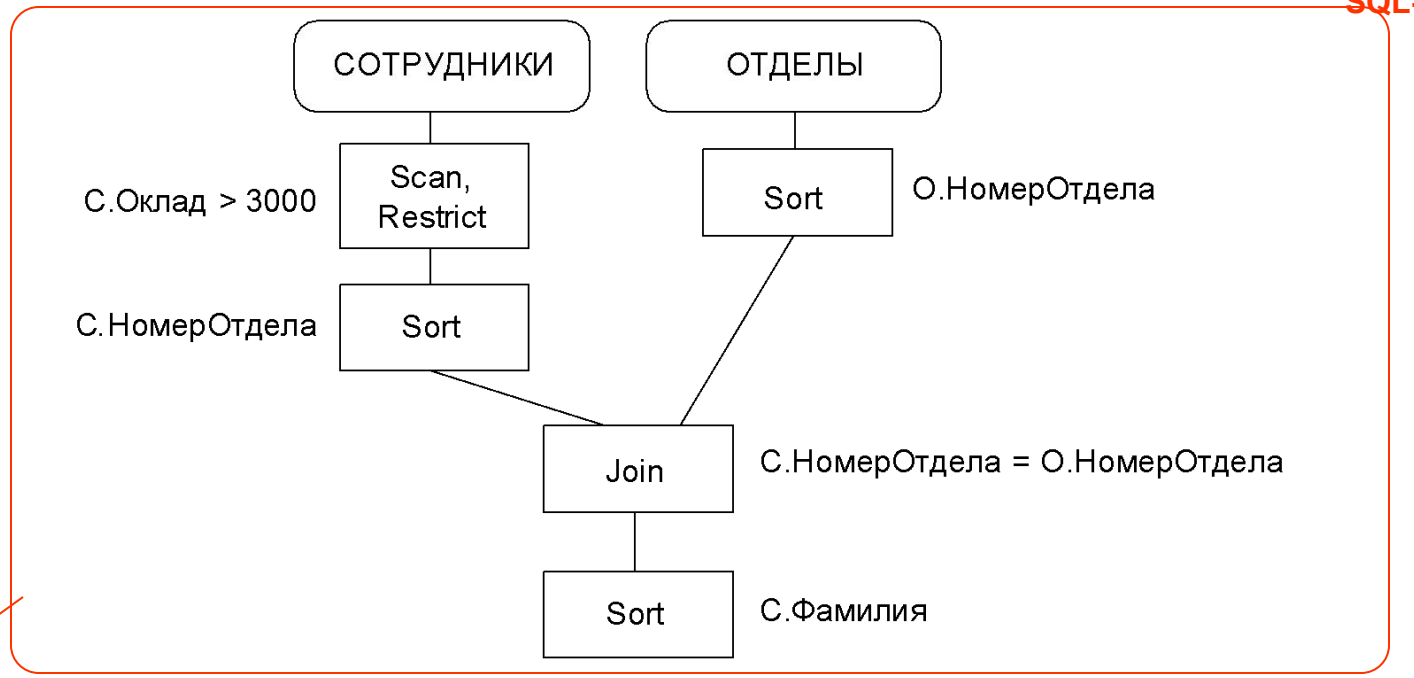
Запрос: Показать информацию о сотрудниках, зарплата которых превышает 3000. Результат упорядочить по фамилиям в алфавитном порядке



ER-диаграмма

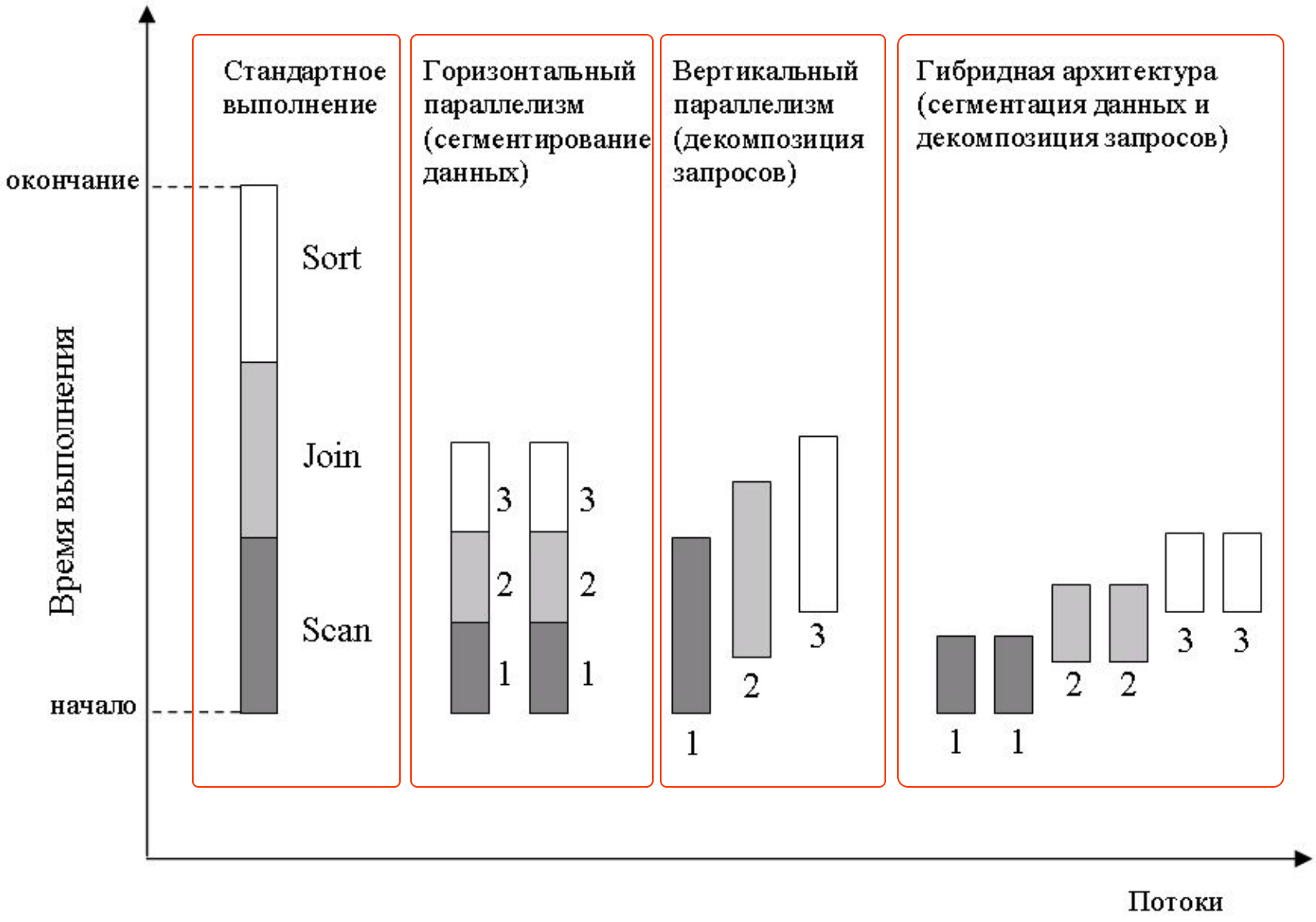
```
SELECT *
FROM Сотрудники С, ОТДЕЛЫ О
WHERE С.Оклад > 3000
AND С.НомерОтдела = О.НомерОтдела
ORDER BY С.Фамилия
```

SQL-запрос



План выполнения SQL-запроса

Многонитиевая мультисерверная архитектура



Преимущества

- рабочая загрузка естественным образом распределена на множестве компьютеров;
- пользователи могут легко совместно пользоваться данными;
- чувствительные к повреждениям данные можно надежно защитить в централизованном порядке;
- использование нескольких малых компьютеров, предназначенных для решения отдельных задач, улучшает показатель «стоимость / эффективность» по сравнению с применением одного мощного компьютера.

Недостатки

- проектировщик должен определить, какие задачи следует выполнять клиенту, а какие – серверу;
- проектировщик должен выбрать для решения каждого типа задач подходящее аппаратное обеспечение;
- переходы к новым версиям программного обеспечения клиента должны контролироваться очень строго;
- изменения конструкции базы данных сервера могут повлиять на всех клиентов;
- сетевая топология часто очень сложна;
- контроль производительности и необходимые регулировки могут оказаться немного сложнее, чем при централизованной архитектуре.

Структура SQL