Понятия информационной системы (ИС) и информационного приложения. Классификация ИС.

Информационная система – это система (программный продукт), предназначенная для сбора, хранения и обработки информации.

Дружественный пользовательский интерфейс ИС

Способ хранения информации в ИС – БД

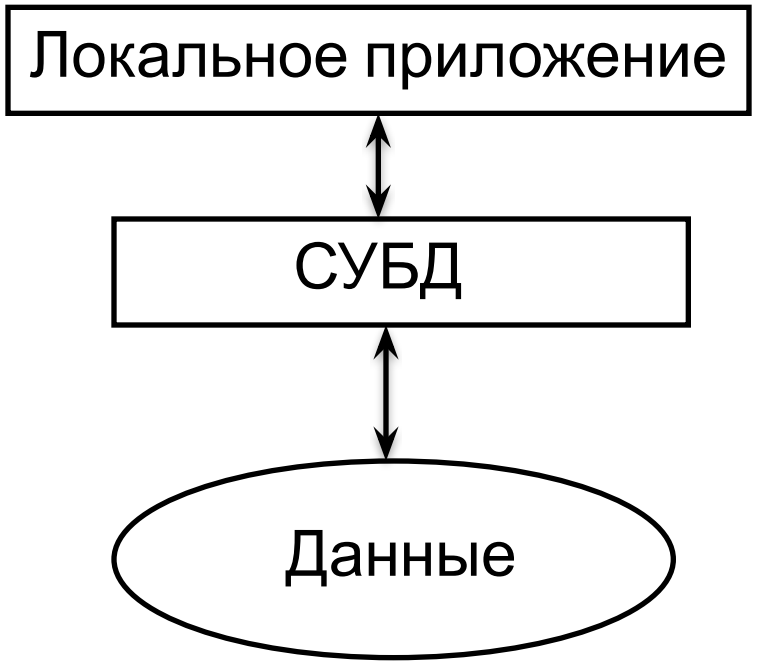
Локальные и распределенные ИС

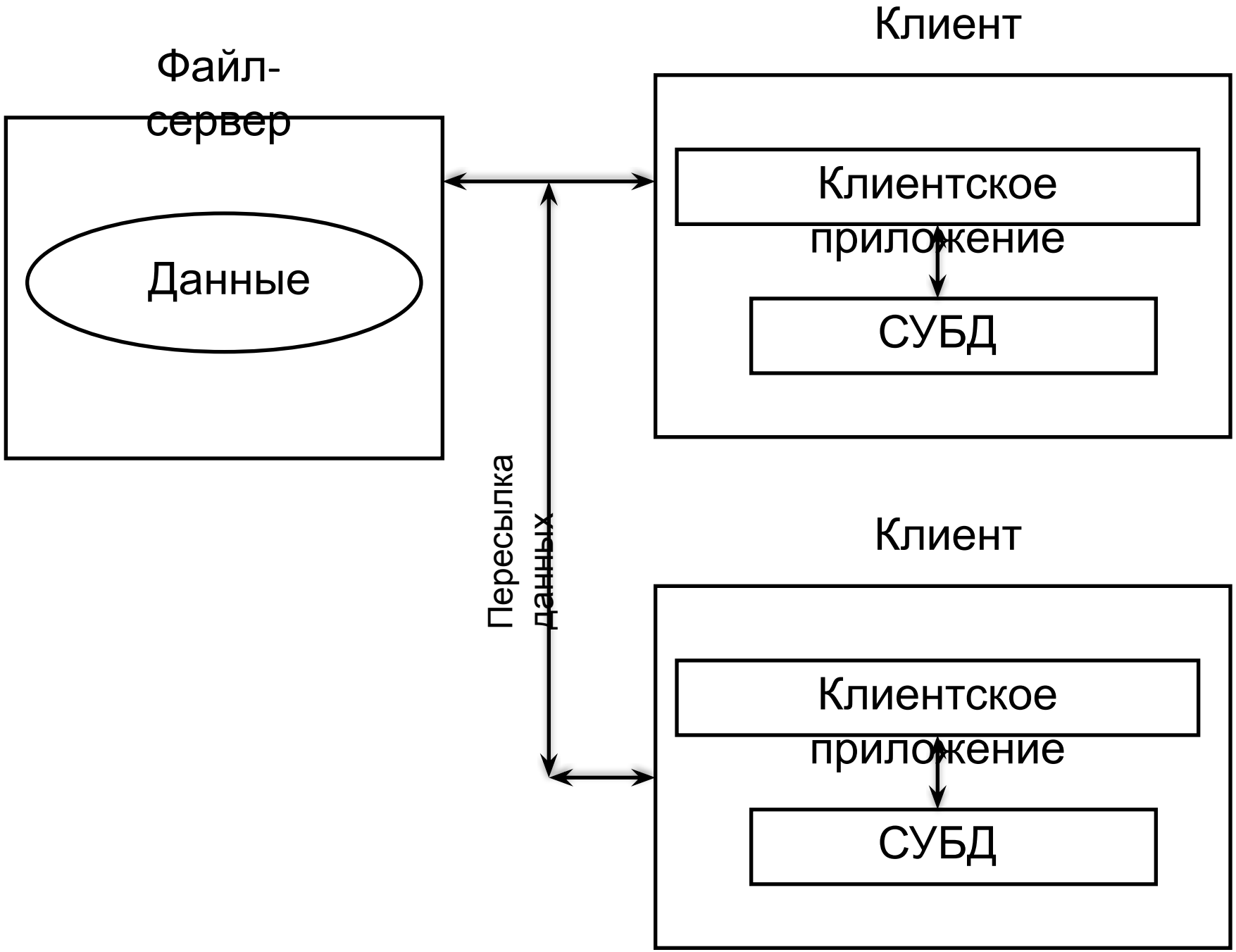
Локальные ИС – все компоненты системы размещаются на одном компьютере

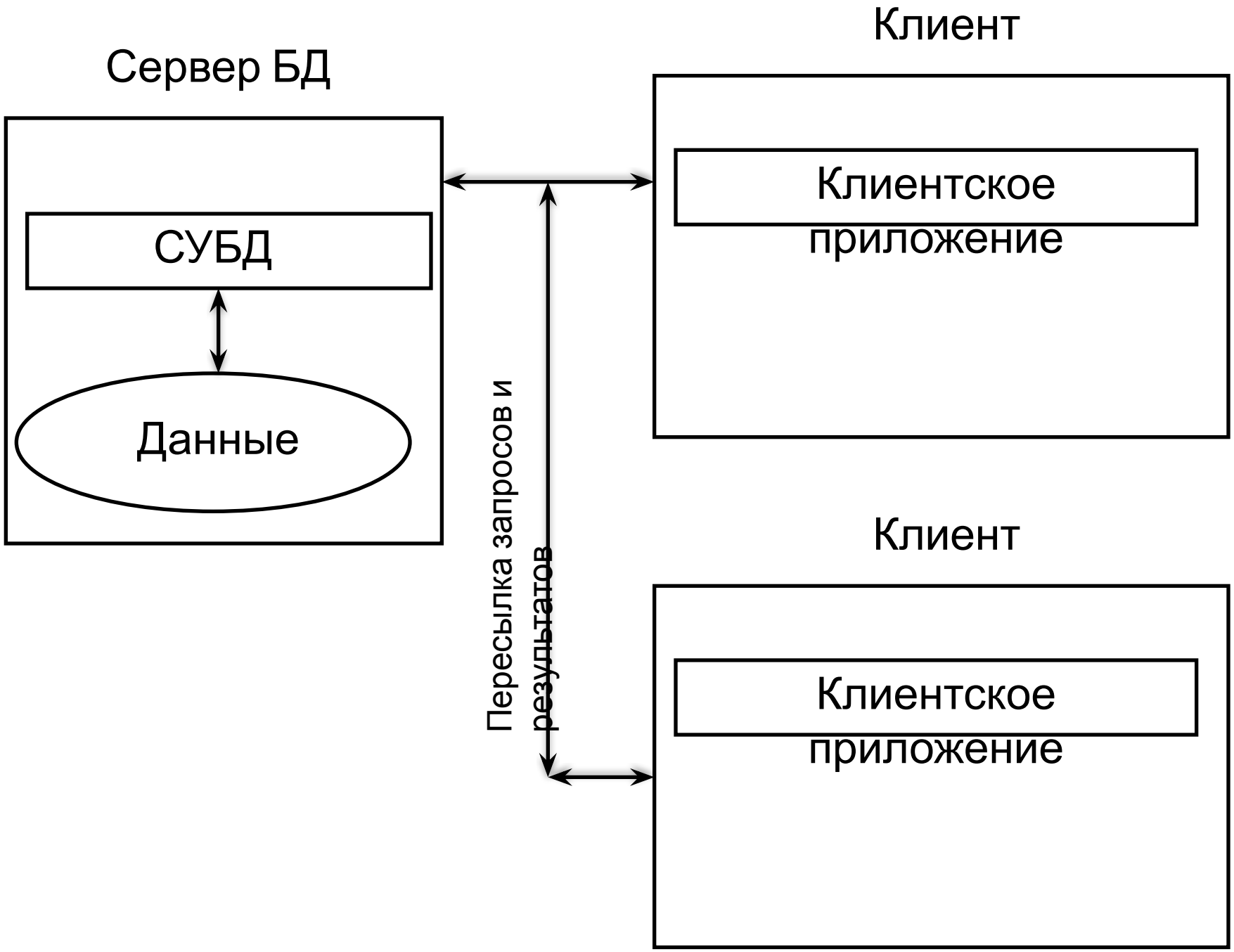
Распределенные – компоненты системы распределяются между несколькими компьютерами

Информационное приложение – прикладная программная подсистема, ориентированная на сбор, хранение, поиск и обработку текстовой и/или фактографической информации.


ИНФОРМАЦИОННЫЕ СИСТЕМЫ
Обработка транзакций
Системы поддержки принятия решений
Информационно-справочные системы
Офисные информационные системы
Оперативная обработка транзакций
Пакетная обработка транзакций
Оперативная аналитическая обработка
Экспертные системы
Системы электронной документации
Географические информационные системы
Гипертекстовые системы
Документальные системы
Автоматизация делопроизводства
Управление документооборотом


Локальное приложение
СУБД
Данные


Файл-сервер
Данные
Клиент
Клиентское приложение
СУБД
Пересылка данных
Клиент
Клиентское приложение
СУБД


Сервер БД
СУБД
Данные
Клиент
Клиентское приложение
Пересылка запросов и результатов
Клиент
Клиентское приложение

Сервер базы данных (БД).

Любое клиент-серверное приложение разделяется на две **части**.
Клиентская часть обеспечивает удобный графический интерфейс и размещается на компьютере пользователя.
Серверная часть осуществляет управление данными, разделение информации, администрирование и обеспечивает безопасность информации.
Сервер БД в любой информационной системе отвечает за хранение данных и предоставление доступа к БД.

Основное требование к БД – минимальное время обработки запросов (max производительность).
Выделяют два вида архитектур сервера БД:

1)Многопроцессорная архитектура сервера БД (ORACLE)

2)Многопоточная архитектура сервера БД (Microsoft)

Виды архитектур серверов базы данных (БД).

1) Многопроцессорная архитектура (ORACLE). При каждом сеансе связи пользователя (клиента) с сервером создается новый исполняемый файл (новый экземпляр приложения).

То есть при каждом подключении пользователя на сервере БД запускается новый исполняемый файл. ОС разделяет процессное время между этими экземплярами приложения.

Преимущество: масштабируемость

Недостаток: необходима большая ресурсоемкость по памяти, поскольку работают сразу несколько экземпляров приложения

2) **Многопоточная архитектура** (Microsoft SQL Server). Запускается всегда только один исполняемый файл, но исполняется несколько потоков (то есть используется один экземпляр приложения с несколькими потоками).

Сервер БД разделяет время, которое имеет каждый поток при доступе к приложению. Задачи ранжируются по приоритету.

Преимущество: меньшая ресурсоемкость по памяти.

Недостаток: меньшая масштабируемость

Принципы клиент-серверного взаимодействия.

1) Приложение имеет серверную и клиентскую части. БД хранится на сервере , но запрос на доступ к БД инициируется клиентом (клиентской частью).

Запрос – это объект базы данных, который служит для поиска нужных сведений в базе данных

2) Основа интерфейса клиент-серверного приложения – это язык **SQL** (*Structured Query Language*, **язык структурированных запросов**). Язык разработан компанией IBM в 1970 году (имел название SEQVEL). В основу языка SQL легла разновидность языка T-SQL (Transact-SQL).

Транзакция – это операция с базой данных или группа последовательных операций с базой данных, которая полностью и успешно выполнена и завершена, либо отменена с откатом в исходное состояние (в случае неудачи)

Принципы клиент-серверного взаимодействия.

3) В качестве сервера БД используется **SQL-сервер**. SQL-сервер – это программный компонент, то есть это логическое имя сос стандартной системой в виде языка запросов SQL (физически это может быть как компьютер, так и группа компьютеров, кластер распределенной системы).

SQL-сервер имеет стандартный интерфейс, поэтому в целом любой SQL-клиент совместим с любым SQL-сервером.

SQL-сервер содержит СУБД и БД.

4) SQL-сервер является центральным звеном ИС. На него возлагается большая нагрузка. SQL-сервер является и узким местом системы.

5) организация клиент-серверного взаимодействия подразумевает балансировку (гибкое распределение) нагрузки между клиентом и сервером.

Для организации клиент-серверного взаимодействия используется **RPC** (*Remote Procedure Call,* **удаленный вызов процедур***)*

RPC — это технология, позволяющая компьютерным программам вызывать функции или процедуры либо на удалённых компьютерах, либо в независимой сторонней системе на том же устройстве.

Принципы клиент-серверного взаимодействия.

В клиент-серверном взаимодействии **RPC** используется следующим образом:

- Позволяет осуществлять балансировку нагрузки между клиентом и сервером;
- Если ЛВС неоднородна, то взаимодействие на уровне процедурных сервисов позволяет скрыть различие между взаимодействующими компьютерами в физически неоднородной сети.
- Позволяет снизить важность аппаратной совместимости серверов и рабочих станций.

Компоненты сетевого информационного приложения

отвечает за пользовательский интерфейс;

реализует алгоритм решения конкретной задачи;

ресурсом обеспечивает доступ к необходимым ресурсам.

Типовые компоненты информационных приложений

PS (Presentation Services) - средства представления. Обеспечиваются устройствами, принимающими ввод от пользователя и отображающим то, что сообщает ему компонент логики представления PL, плюс соответствующая программная поддержка. Может быть текстовым терминалом или X-терминалом, а также ПК или рабочей станцией в режиме программной эмуляции терминала или X-терминала.

PL (Presentation Logic) - логика представления. Управляет взаимодействием между пользователем и ЭВМ. Обрабатывает действия пользователя по выбору альтернативы меню, по нажатию кнопки или при выборе элемента из списка.

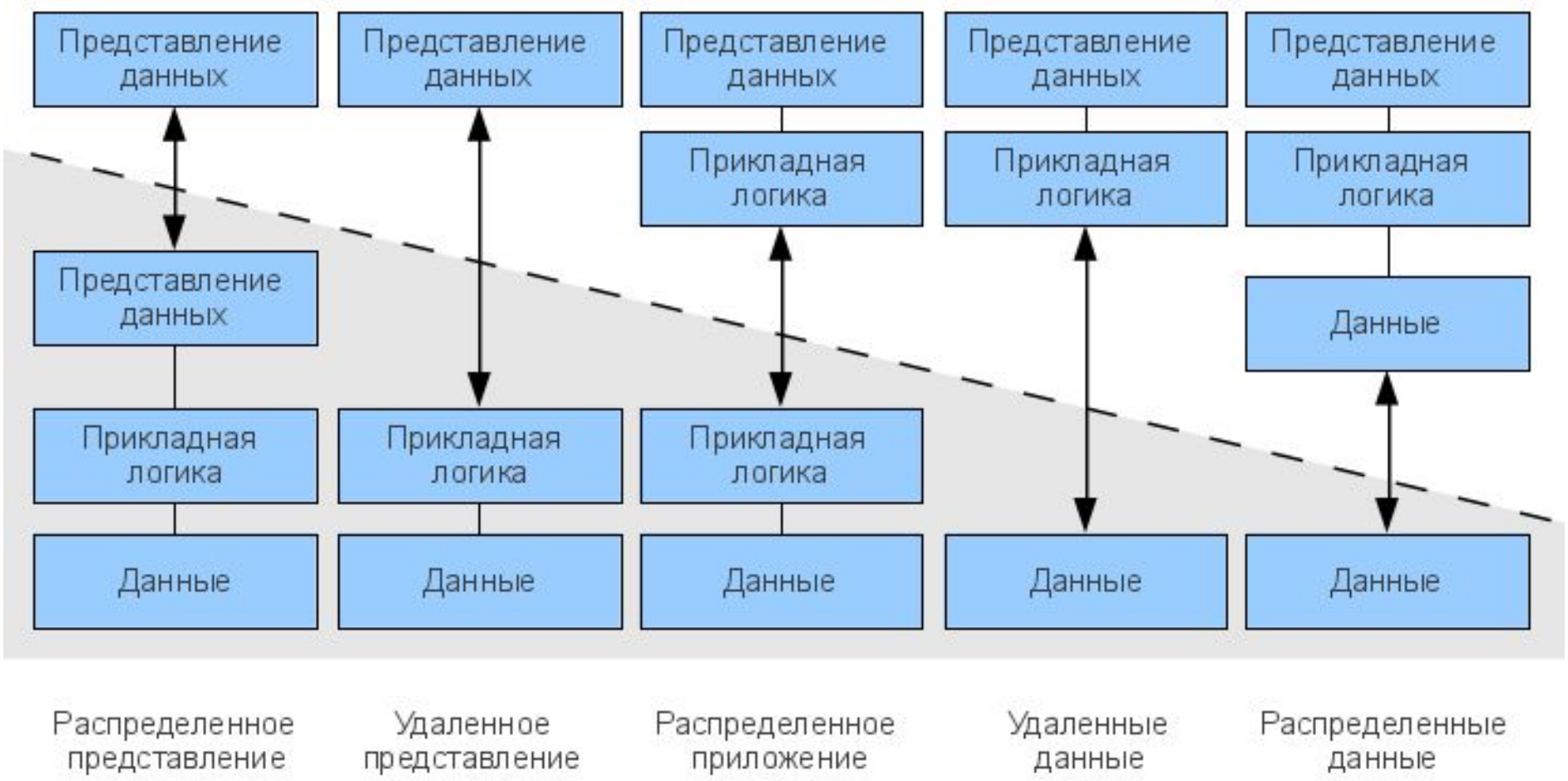
BL (Business or Application Logic) - прикладная логика. Набор правил для принятия решений, вычислений и операций, которые должно выполнить приложение.

DL (Data Logic) - логика управления данными. Операции с базой данных (SQL-операторы SELECT, UPDATE и INSERT), 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 компонент. Обычно являются функциями ОС.

Модели клиент-серверного взаимодействия



Модели архитектуры «клиент-сервер»:

-модель «толстый клиент»;

-модель «тонкий клиент»;

- модель «сервер приложения»

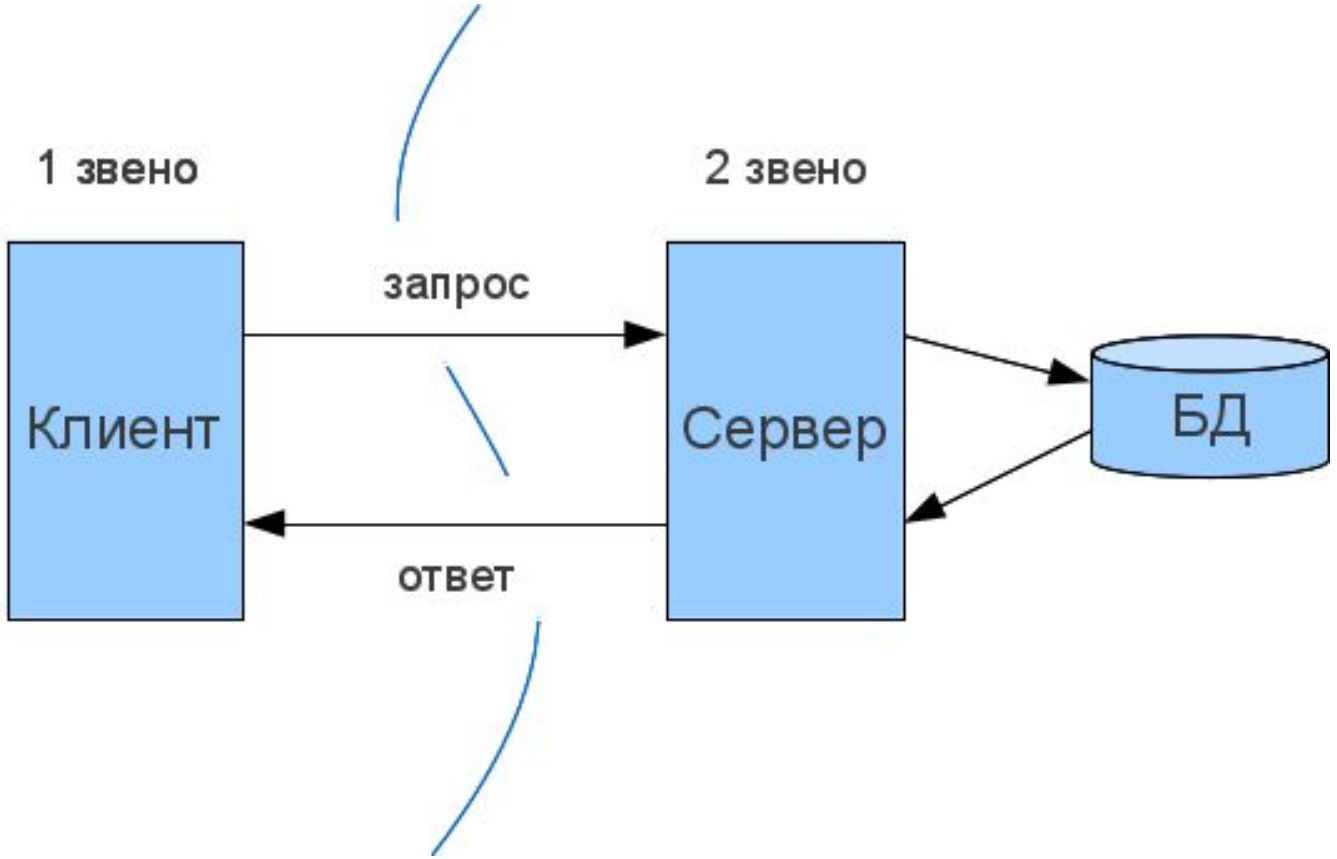
По количеству уровней:

- двухзвенная (двухуровневая) модель;

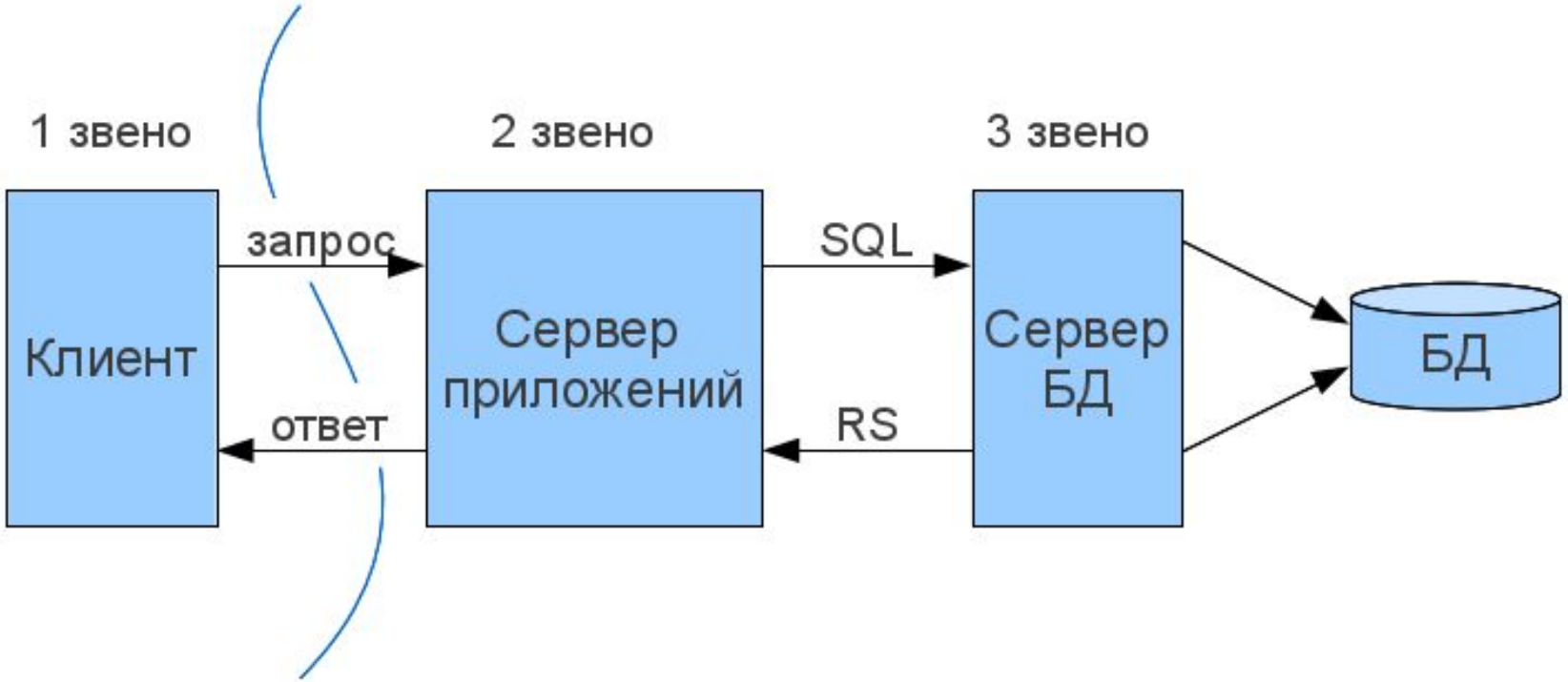
-трехзвенная (трехуровневая) модель;

-многозвенная (многоуровневая) модель.

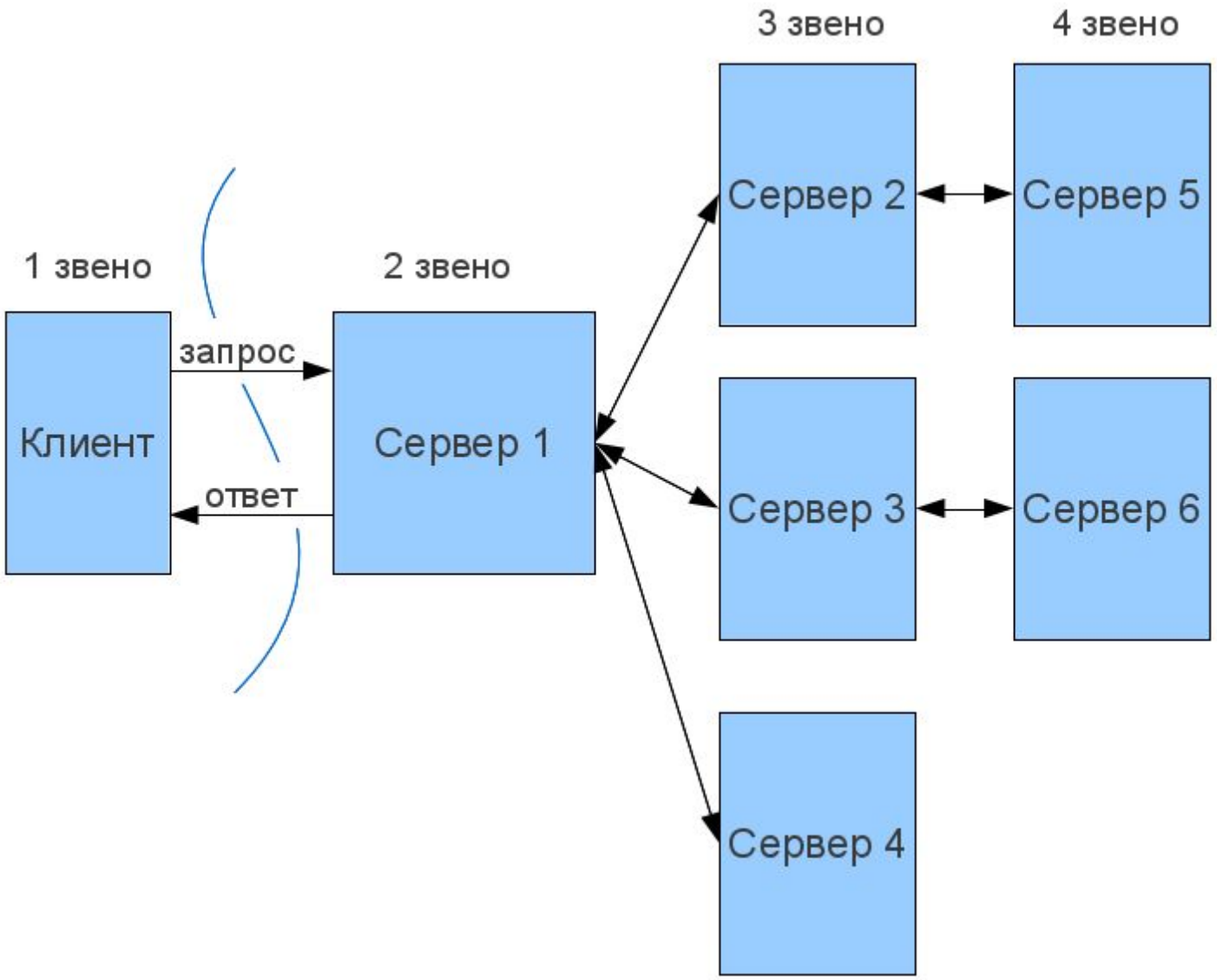
Двухзвенная клиент-серверная архитектура



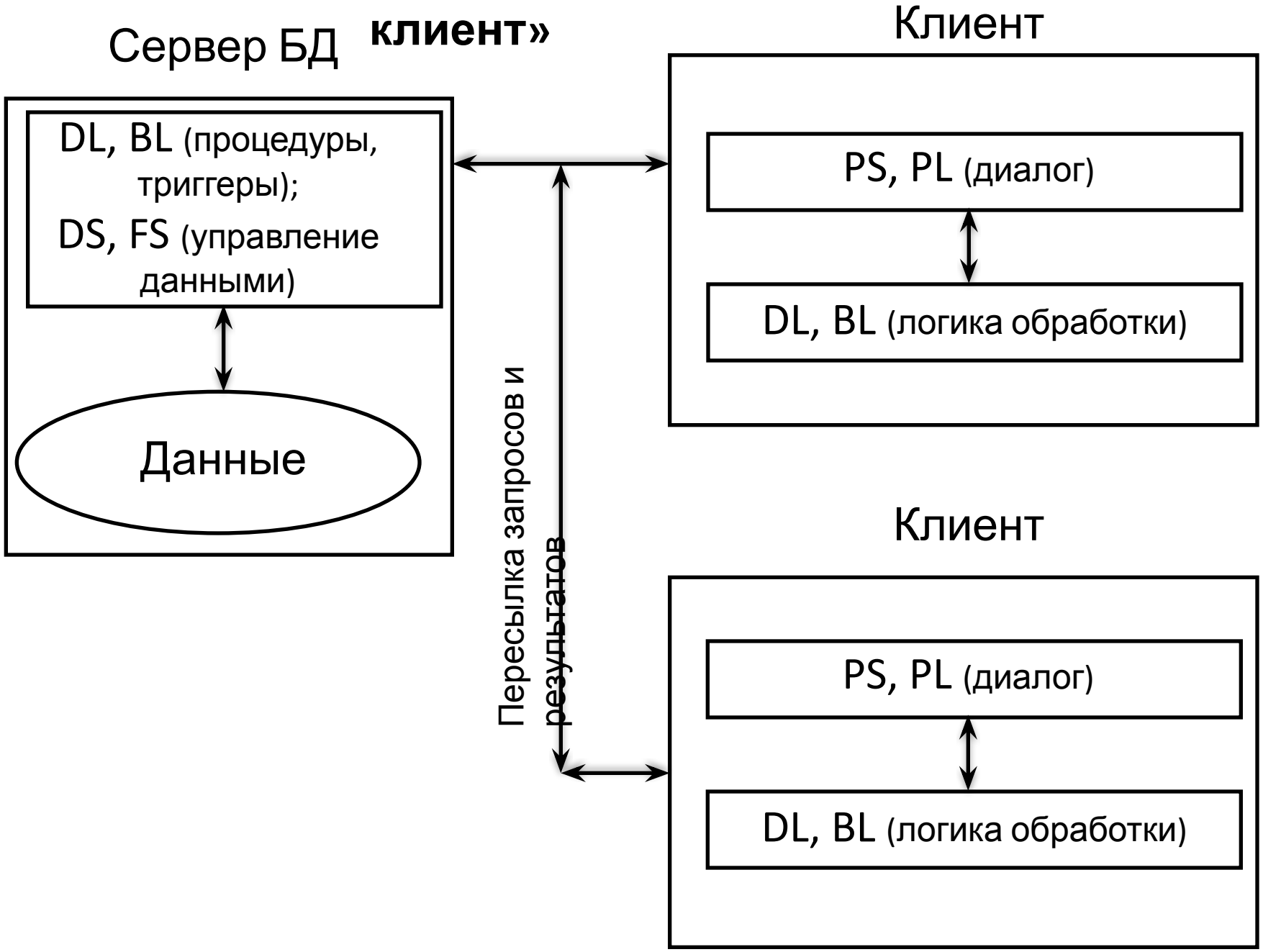
Трехзвенная клиент-серверная архитектура



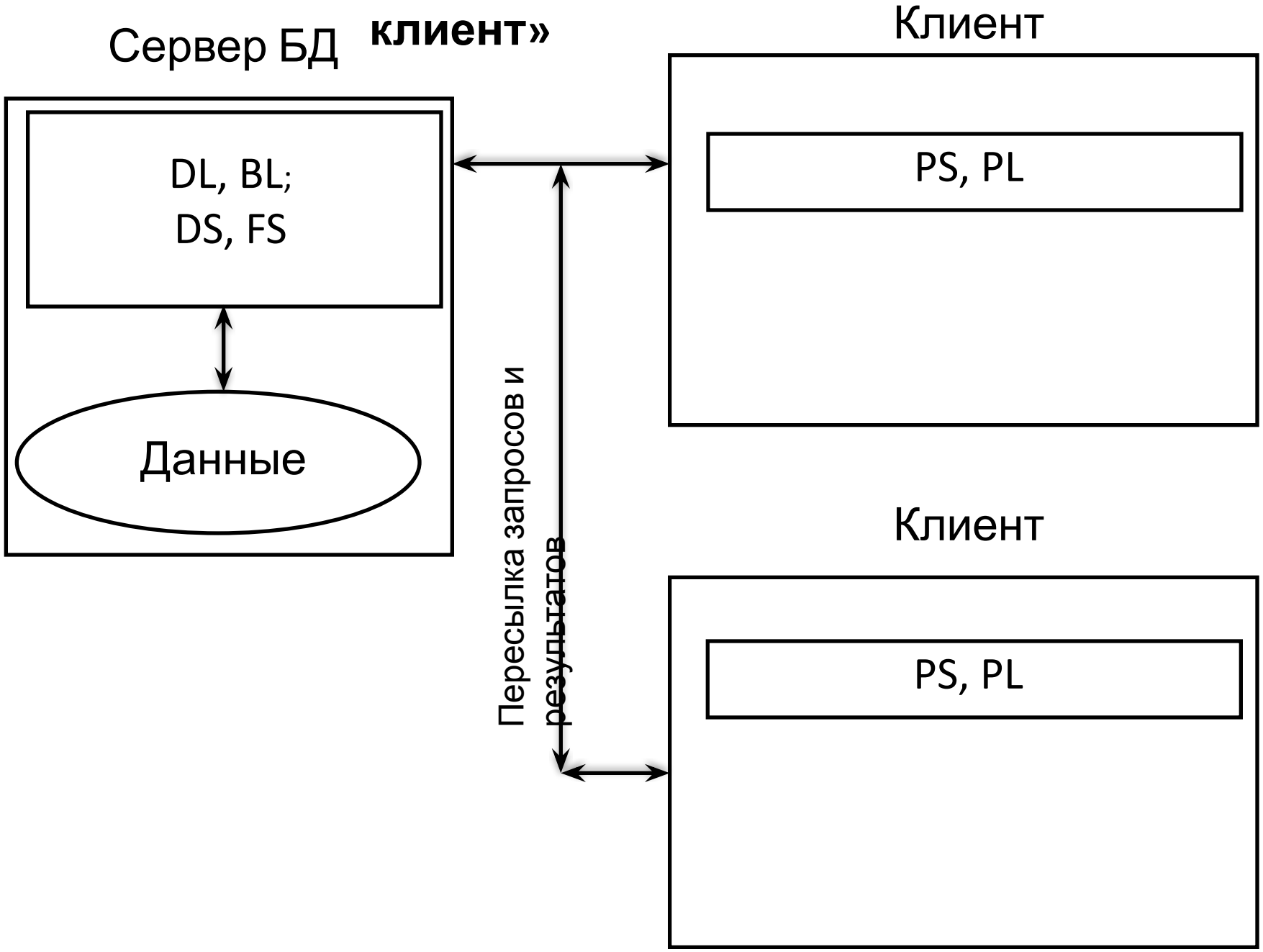
Многозвенная (N-tier) клиент-серверная архитектура



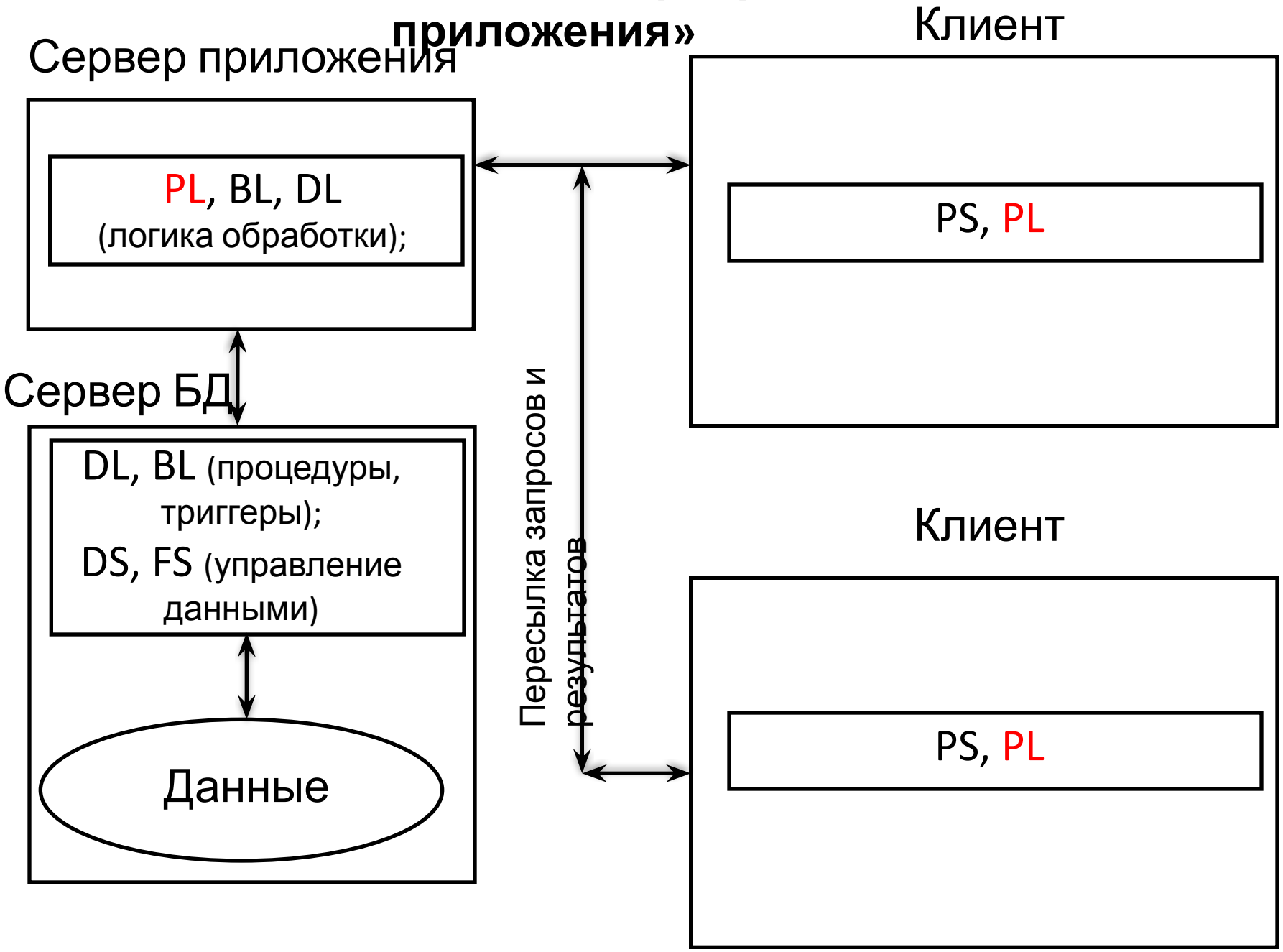
Модель «толстый клиент»



Модель «тонкий клиент»



Модель «сервер приложения»



Схемы построения информационных систем

Сх	Описание схемы	Клиент	Сервер - 1	Сервер - 2	Пример реализации
1	Централизованная многотерминальная система	PS	PL, BL, DL, DS, FS	-	Сервер Sun с X-терминалами в среде ОС Solaris
2	Локальная сеть ПК с файл серверными приложениями	PS, PL, BL, DL, DS	FS	-	Локальная сеть ПК в среде NetWare, программы на FoxPro, Clipper и др
3	Удаленный доступ к данным на сервере БД	PS, PL, BL, DL	DS, FS	-	Система клиент-сервер с доступом ПК к серверу БД Informix (NetWare)
4	Удаленный доступ к БД с использованием хранимых процедур	PS, PL, DL	BL, DS, FS	-	Система клиент-сервер, доступ ПК к серверу ORACLE в среде SCO Unix
5	Удаленный доступ к БД с разделением логики приложения	PS, PL, BL, DL	BL, DL, DS, FS	-	Система клиент-сервер, доступ ПК к серверу ORACLE на Sun (Solaris)
6	Удаленное представление данных с доступом к Unix-системе	PS, PL	BL, DL, DS,FS	-	Сеть ПК/станций, приложения на мониторе транзакций и СУБД в Unix
7	Удаленное управление файл-серверным приложением в сети	PS	PL, BL, DL, DS	FS	Связь удаленных ПК с сервером доступа WinView в сети для работы с СУБД FoxPro
8	Многотерминальный сервер приложений для доступа к СУБД	PS	PL, BL, DL	DS, FS	Сервер приложений на SCO Unix, доступ терминалов к ORACLE на HP
9	3-х звенная система на Unix с монитором транзакций	PS, PL	BL, DL	DS, FS	Сеть ПК, сервер приложений на TUXEDO и СУБД ORACLE в среде Solaris на Sun
10	3-х звенная система с монитором транзакций и	PS, PL, BL	BL, DL	DS, FS	Аналогично предыдущему, но контроль данных выполняется на

Способы построения информационных приложений :

-многотерминальные централизованные вычислительные системы;

-системы на основе локальной сети ПК (файл-серверные приложения);

-системы с архитектурой клиент-сервер;

-системы с распределенными вычислениями;

-системы на основе Internet/Intranet-технологий;

- системы с сервис-ориентированной архитектурой

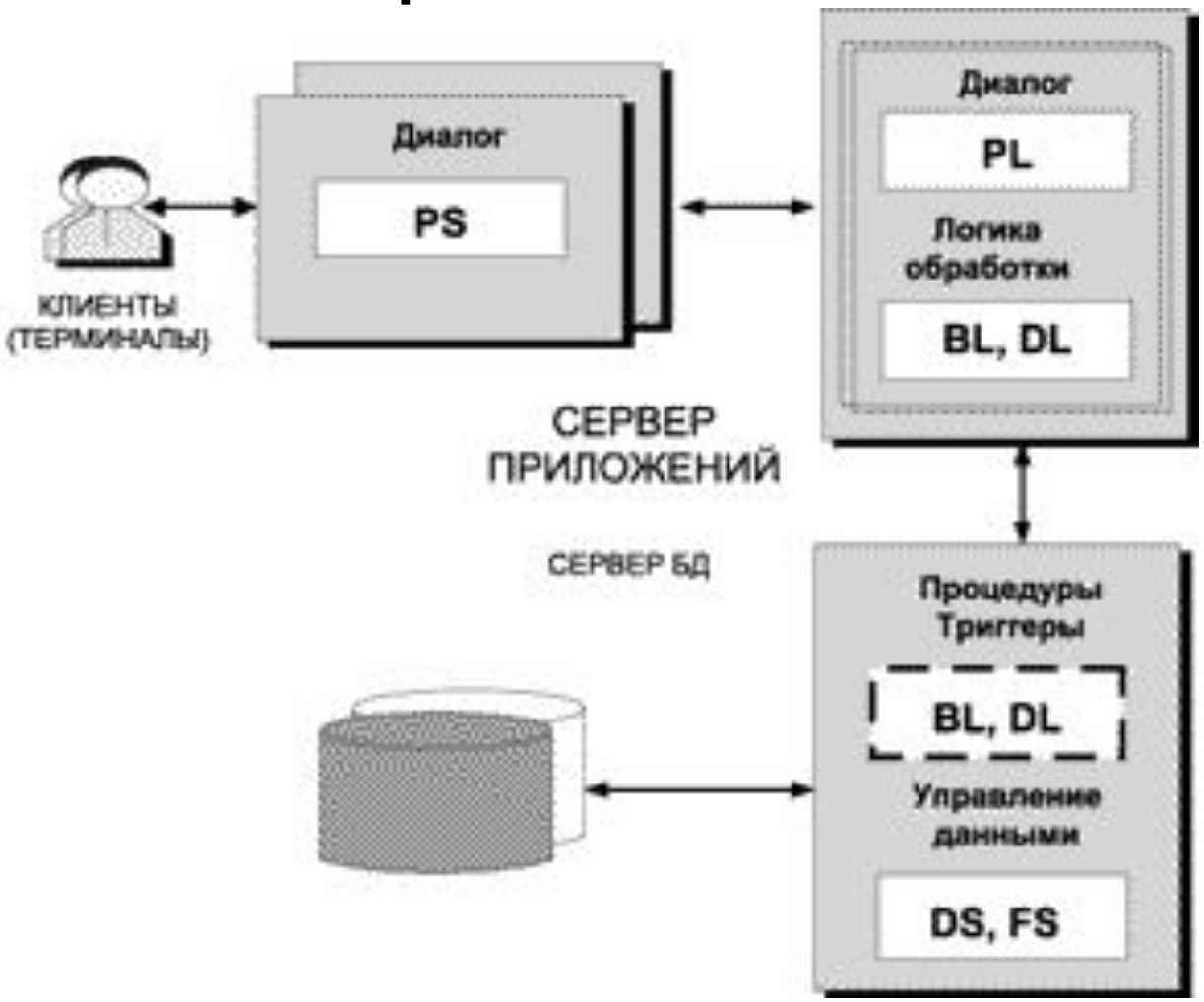
-офисные системы.

Централизованная архитектура

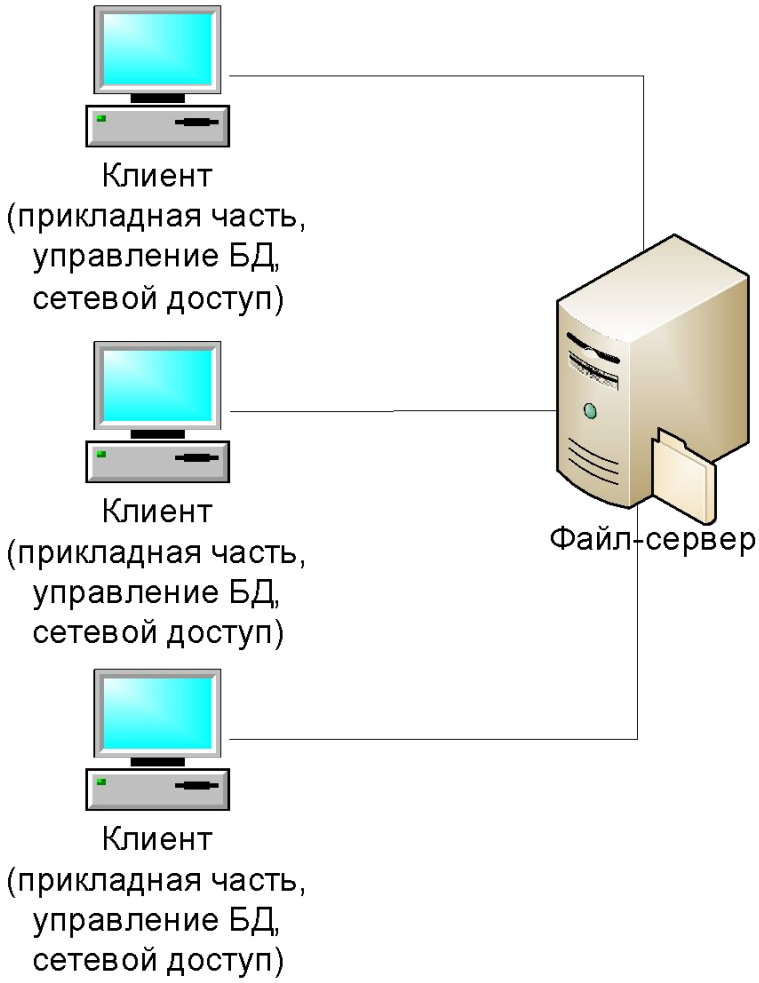
Централизованная архитектура

- **Достоинства:**
  - пользователи совместно используют дорогие ресурсы ЭВМ и дорогие периферийные устройства
  - централизация ресурсов и оборудования облегчает обслуживание и эксплуатацию вычислительной системы
  - отсутствует необходимость администрирования рабочих мест пользователей
- **Главный недостаток:**
  - пользователи полностью зависят от администратора хост-ЭВМ

Приложения клиент-сервер на основе многотерминальной системы



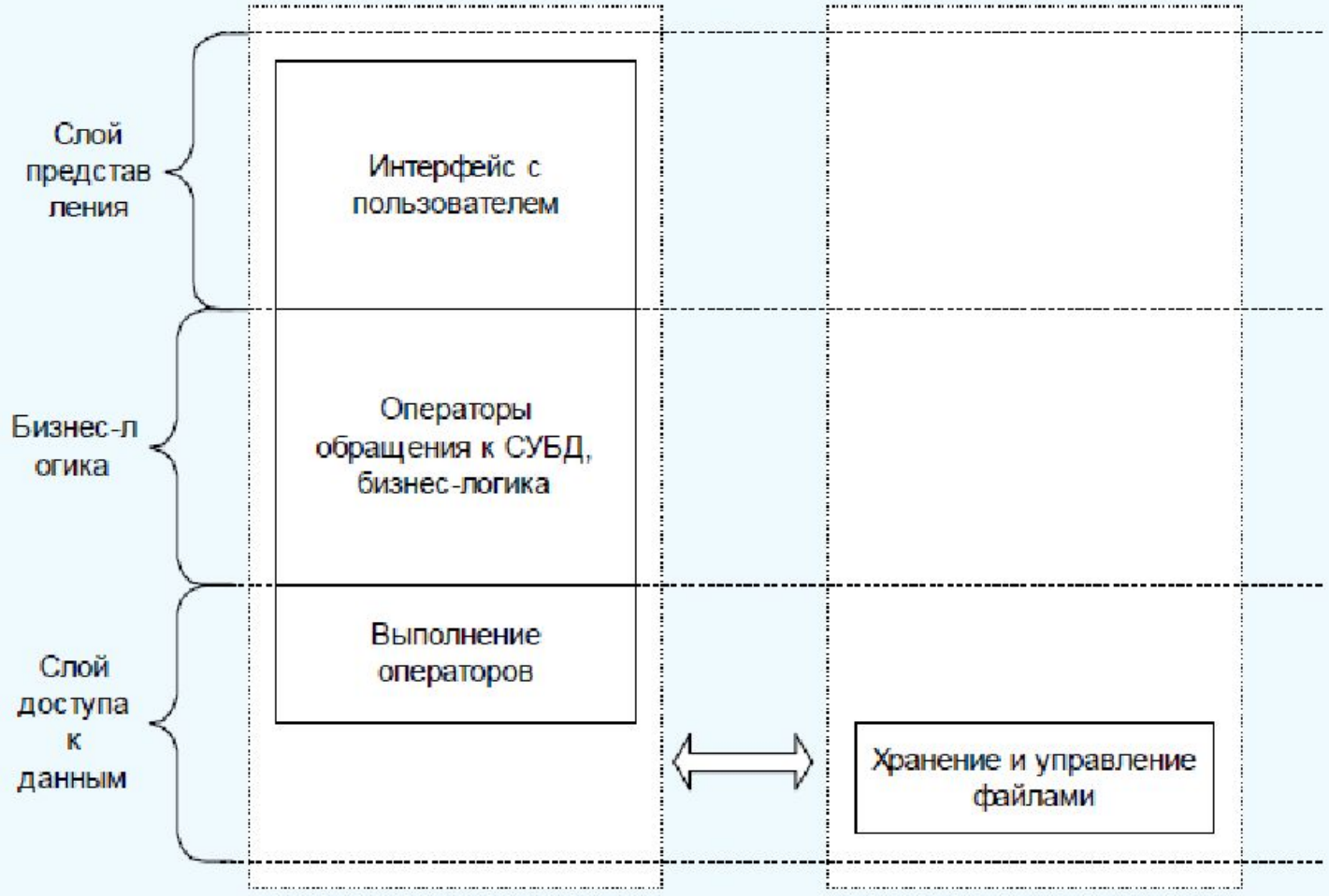
Архитектура «файл-сервер»



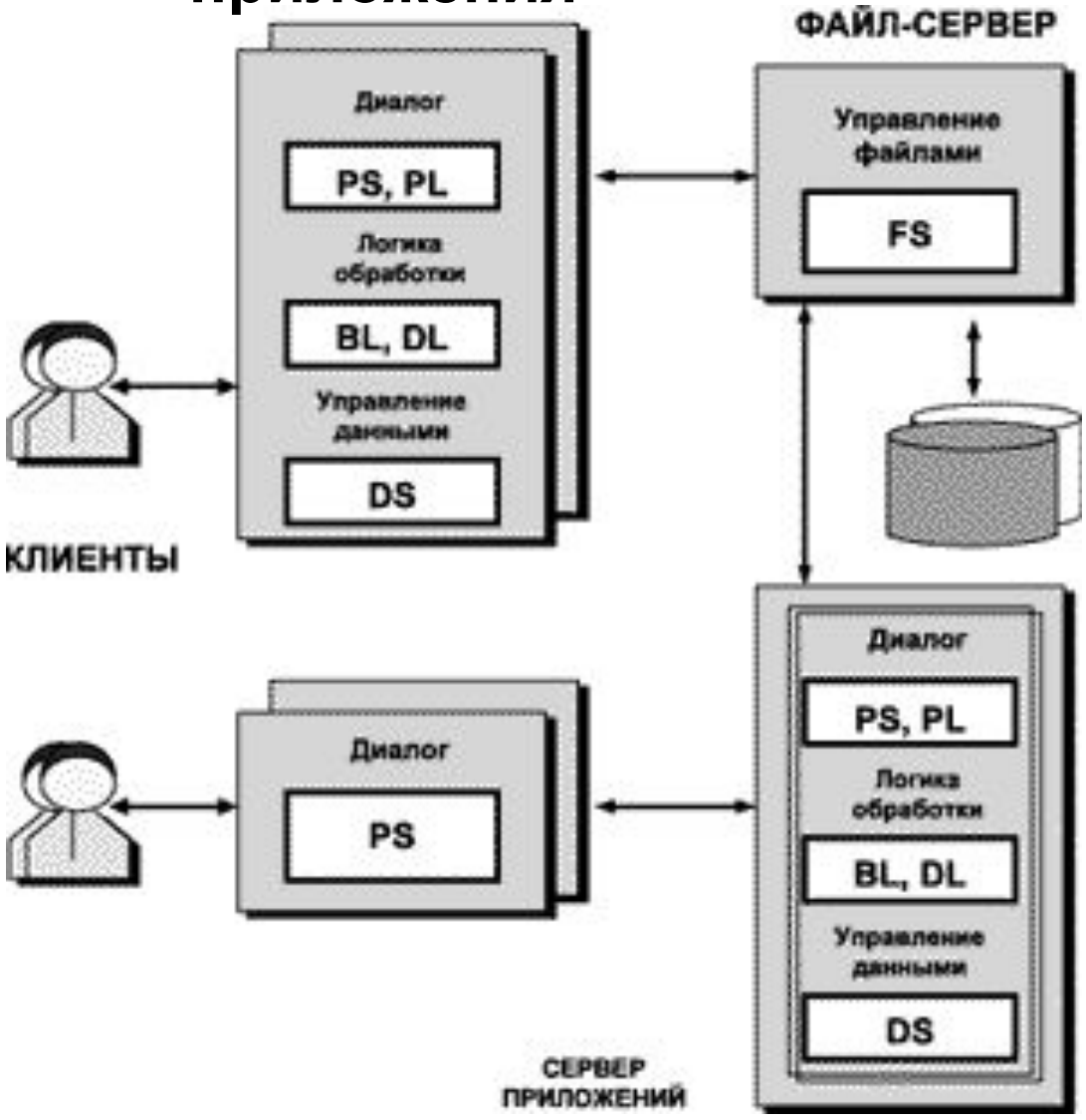
Архитектура «файл-сервер»

- **Достоинства:**
  - многопользовательский режим работы с данными
  - удобство централизованного управления доступом
  - низкая стоимость разработки
  - высокая скорость разработки
  - невысокая стоимость обновления и изменения ПО
- **Недостатки:**
  - проблемы многопользовательской работы с данными
  - низкая производительность
  - плохая возможность подключения новых клиентов

Модель файлового сервера



Файл-серверные приложения



Инструменты для создания файл-серверных приложений

Для разработки файл-серверных приложений традиционно используется инструментальное окружение «персональных» СУБД класса xBase :

FoxPro, Clipper, Paradox, Clarion, Paradox, dBase и т. п.

"визуальные" инструменты этого класса:

Visual FoxPro, CA-Visual Objects, Visual dBase

Особенности средств разработки:

1)они реализованы в виде диалоговой интегрированной среды;

2)Предоставляют 3 уровня доступа:

а) программирование и создание приложений на языке, сочетающем возможности языка 3GL с некоторыми возможностями языков четвертого поколения 4GL;

б) создание и ведение структуры БД и индексов, а также интерактивная генерация макетного приложения и его компонентов (меню, форм или окон, отчетов, запросов и программных модулей);

в) использование диалоговой среды и генераторов конечными пользователями для создания, ведения и просмотра БД, а также формирования несложных запросов и отчетов.

СУБД MS Access

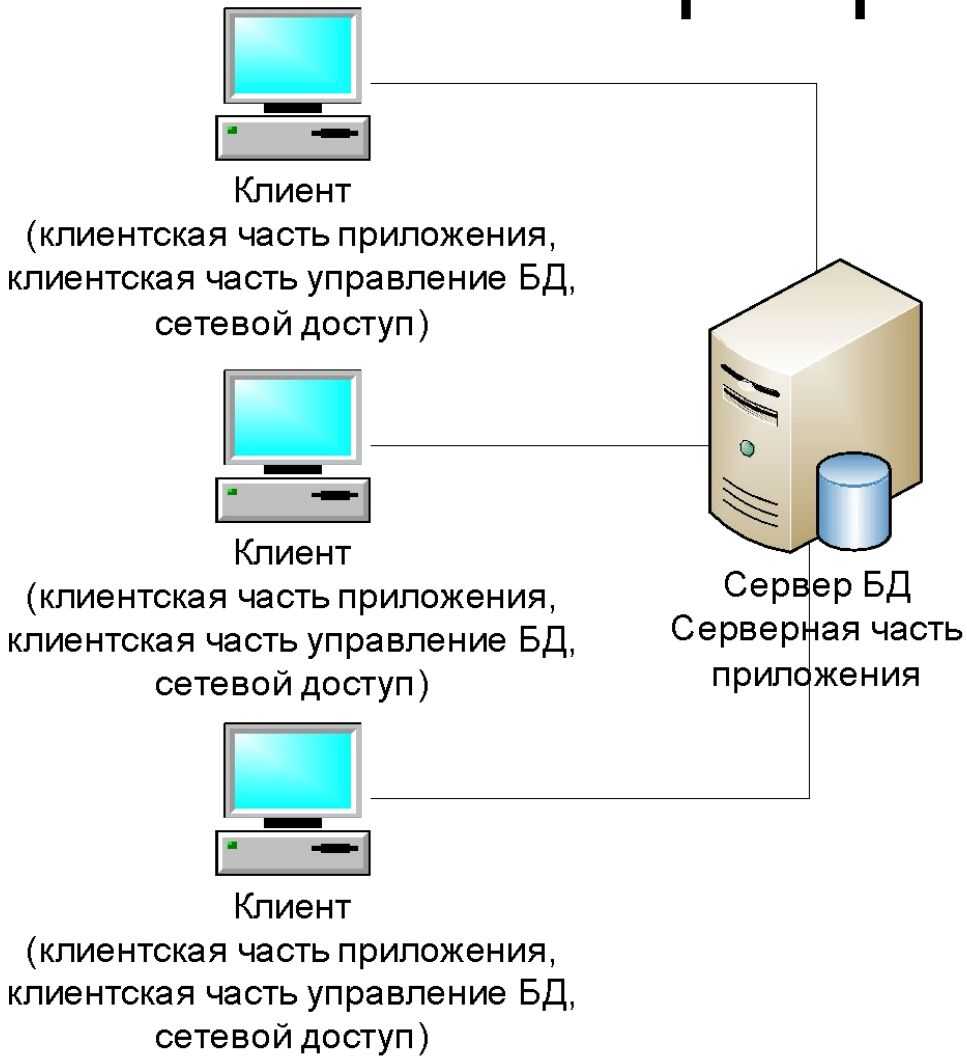
СУБД для ПК MS Access может использоваться:

- для создания масштабируемых одиночных и групповых информационных приложений;
- для разработки клиентской части приложений клиент-сервер;
- как средство автоматизации делопроизводства в составе MS-Office.

Преимущества MS Access:

- позволяет разрабатывать файл-серверные приложения с возможностью масштабирования;
- имеет удобные средства визуального конструирования, отладки;
- имеет возможности использования как Access Basic, так и SQL;
- имеет возможности переноса и адаптации приложений в архитектуру клиент-сервер с использованием интерфейса ODBC.

Двухуровневая архитектура «клиент-сервер»

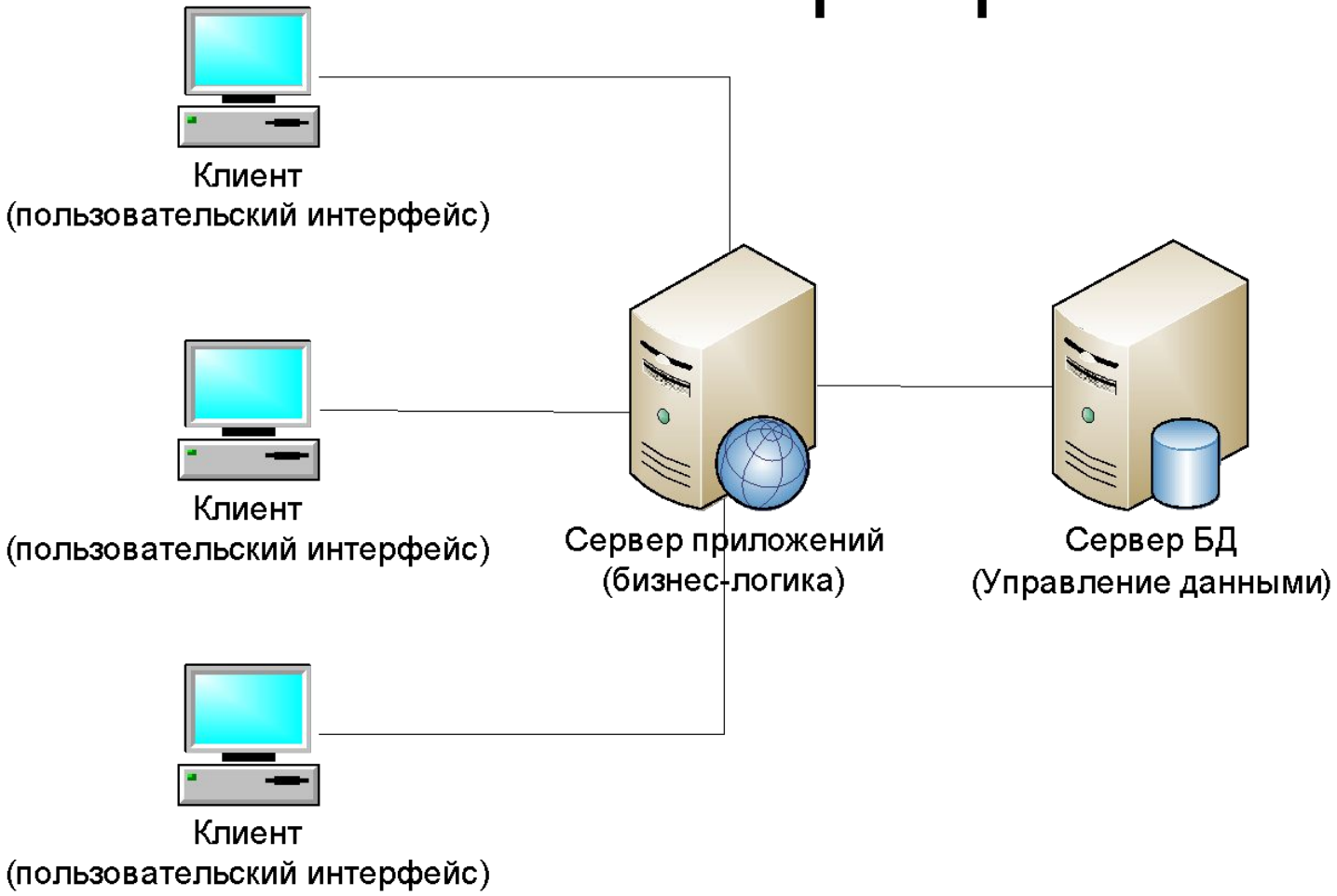


Двухуровневая архитектура «клиент-сервер»

- **Достоинства:**
  - возможность распределить функции вычислительной системы между несколькими независимыми компьютерами
  - все данные хранятся на защищенном сервере
  - поддержка многопользовательской работы
  - гарантия целостности данных
- **Недостатки:**
  - неработоспособность сервера может сделать неработоспособной всю вычислительную сеть
  - сложное администрирование
  - высокая стоимость оборудования
  - бизнес логика приложений осталась в клиентском ПО

Модель сервера СУБД

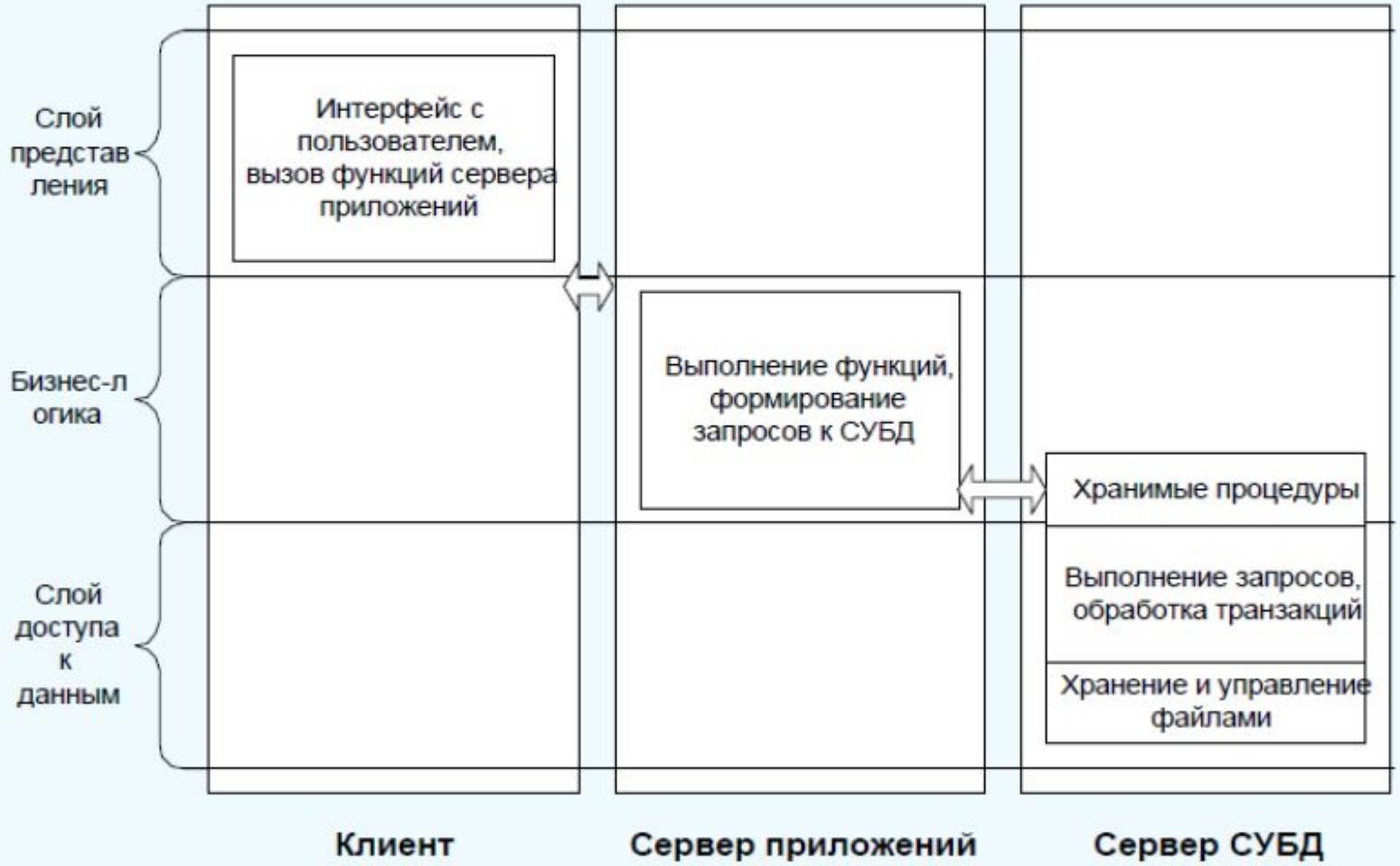
Многоуровневая архитектура «клиент-сервер»



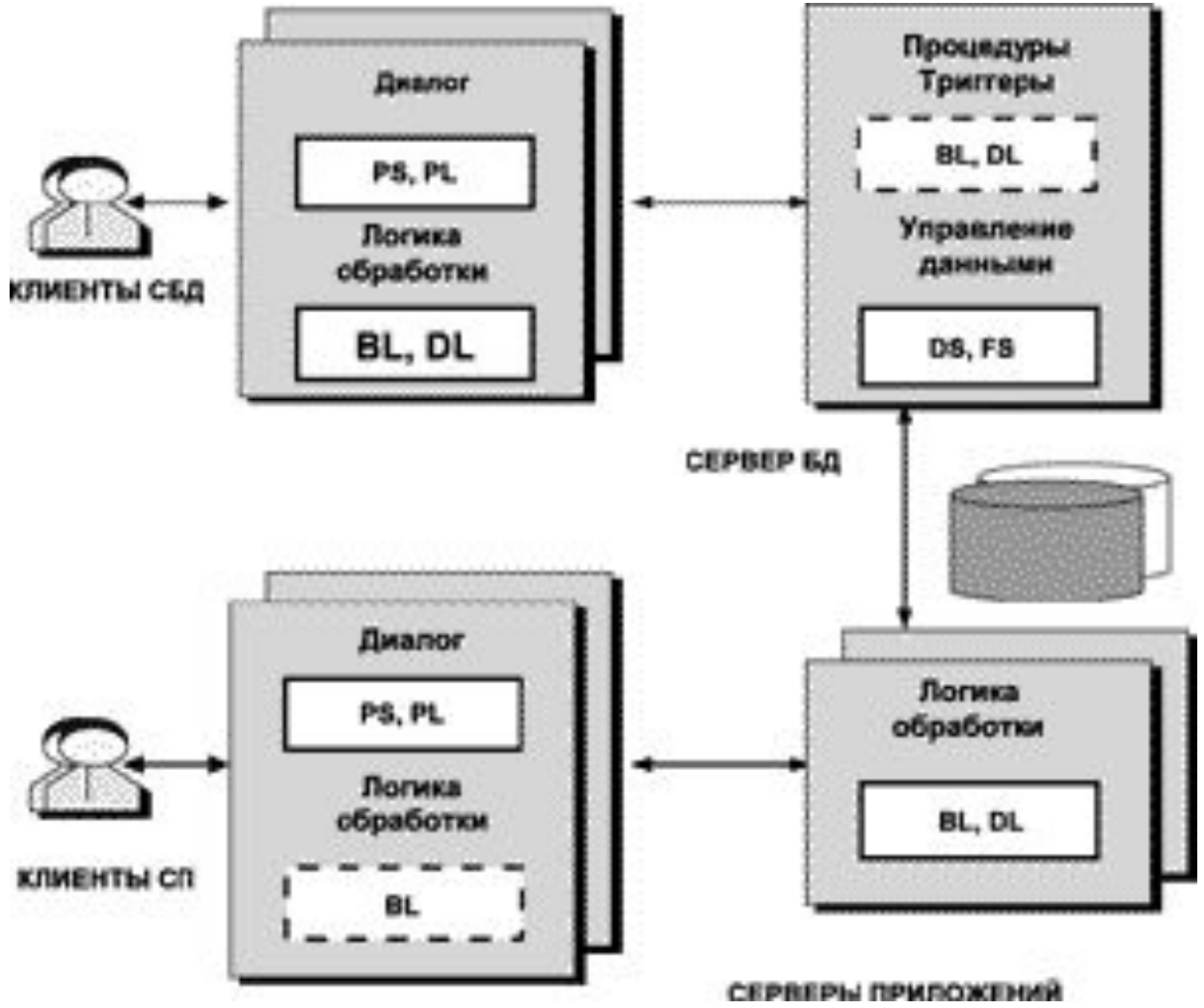
Многоуровневая архитектура «клиент-сервер»

- **Достоинства:**
  - клиентское ПО не нуждается в администрировании
  - масштабируемость
  - конфигурируемость
  - высокая безопасность и надежность
  - низкие требования к скорости канала между терминалами и сервером приложений
  - низкие требования к производительности и техническим характеристикам терминалов
- **Недостатки:**
  - сложность администрирования и обслуживания
  - более высокая сложность создания приложений
  - высокие требования к производительности серверов приложений и сервера базы данных
  - высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений

Модель сервера приложений



Варианты построения приложений клиент-сервер



Средства разработки приложений клиент-сервер

Средства для создания информационных приложений с архитектурой клиент-сервер делятся на следующие подгруппы:

- среды разработки приложений для серверов баз данных,
- независимые от СУБД инструменты для создания приложений клиент-сервер,
- средства поддержки распределенных информационных приложений.

Инструментальные средства для разработки приложений клиент-сервер необходимо выбирать, исходя из следующих критериев:

1) наличие объектно-ориентированной инфраструктуры,
2) возможности распределения приложений между клиентом и сервером,
3) обеспечена ли поддержка мониторов транзакций,
4) доступность CASE-репозитария,
5) возможность переноса приложений и контроль версий.

Среды разработки приложений для серверов баз данных

К таким средствам относят системы программирования четвертого поколения 4GL или инструментальные средства быстрой разработки приложений RAD (Rapid Application Development).

Особенности этих средств:

1) реализация удаленного доступа к СУБД по двухзвенной схеме клиент-сервер;
2) связь клиентских приложений с серверами БД с помощью непроцедурного языка структурированных запросов SQL (кроме серверов Btrieve);
3) обеспечение целостности БД, включая целостность транзакций;
4) поддержка хранимых процедур на серверах БД;
5) реализация клиентских и серверных триггеров-процедур;
6) генерация элементов диалогового интерфейса и отчетов.

Примеры среды разработки SQL-серверов БД (Informix NewEra и Oracle Power Objects).

Независимые инструментальные средства

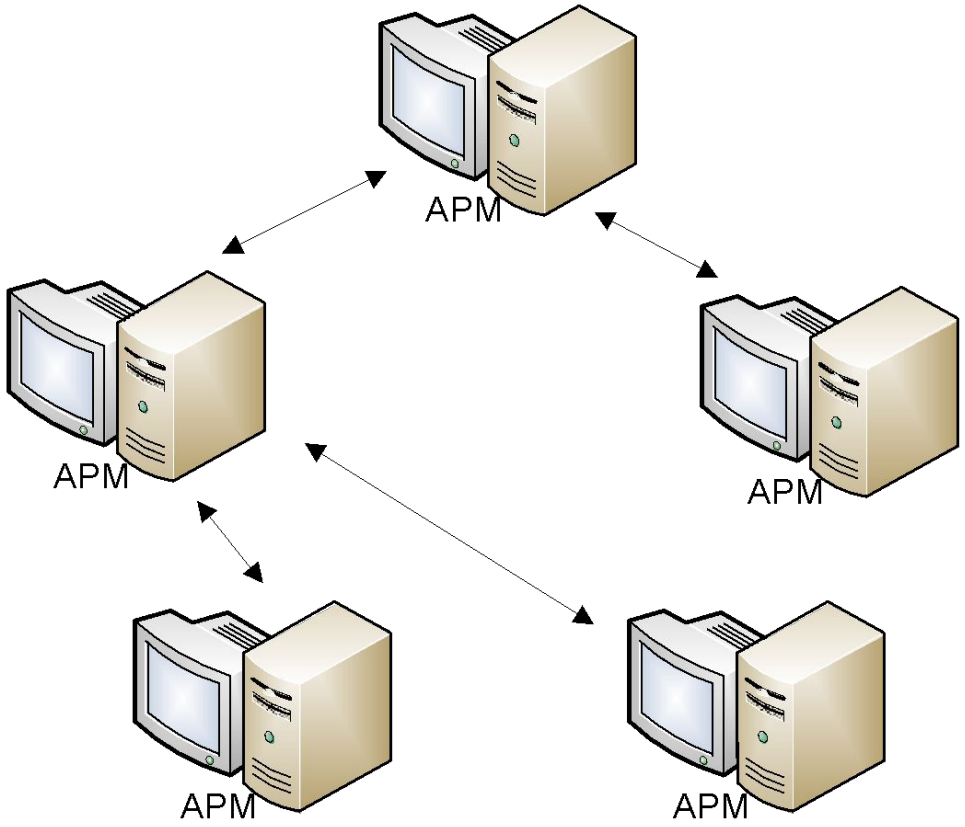
Эти средства ориентированы на многие платформы СУБД (то есть не зависят от СУБД) и представлены в виде средств быстрой разработки приложений RAD.

Для таких средств создания приложений клиент-сервер характерны следующие особенности:

- возможность распределения приложения на клиентах и/или серверах;
- создание приложений для разных серверов БД;
- поддержка спецификации ODBC для доступа к различным серверам БД, включая СУБД для ПК;
- связь с мониторами транзакций для организации трехзвенной архитектуры приложений клиент-сервер;
- объектно-ориентированное программирование приложений;
- визуальный характер генерации приложения;
- ведение репозитария объектов и их свойств, что облегчает интеграцию со средствами автоматизации проектирования программ CASE;
- управление проектами и версиями приложений;
- интеграция приложения с электронной почтой и средствами офисной автоматизации.

Примеры средств разработки: **SQLWindows, PowerBuilder, JAM и**

Архитектура распределенных систем



Организация распределенной системы

Средства поддержки распределенных приложений

Средства поддержки распределенных приложений относятся к категории промежуточного программного обеспечения middleware *для организации серверов приложений.*

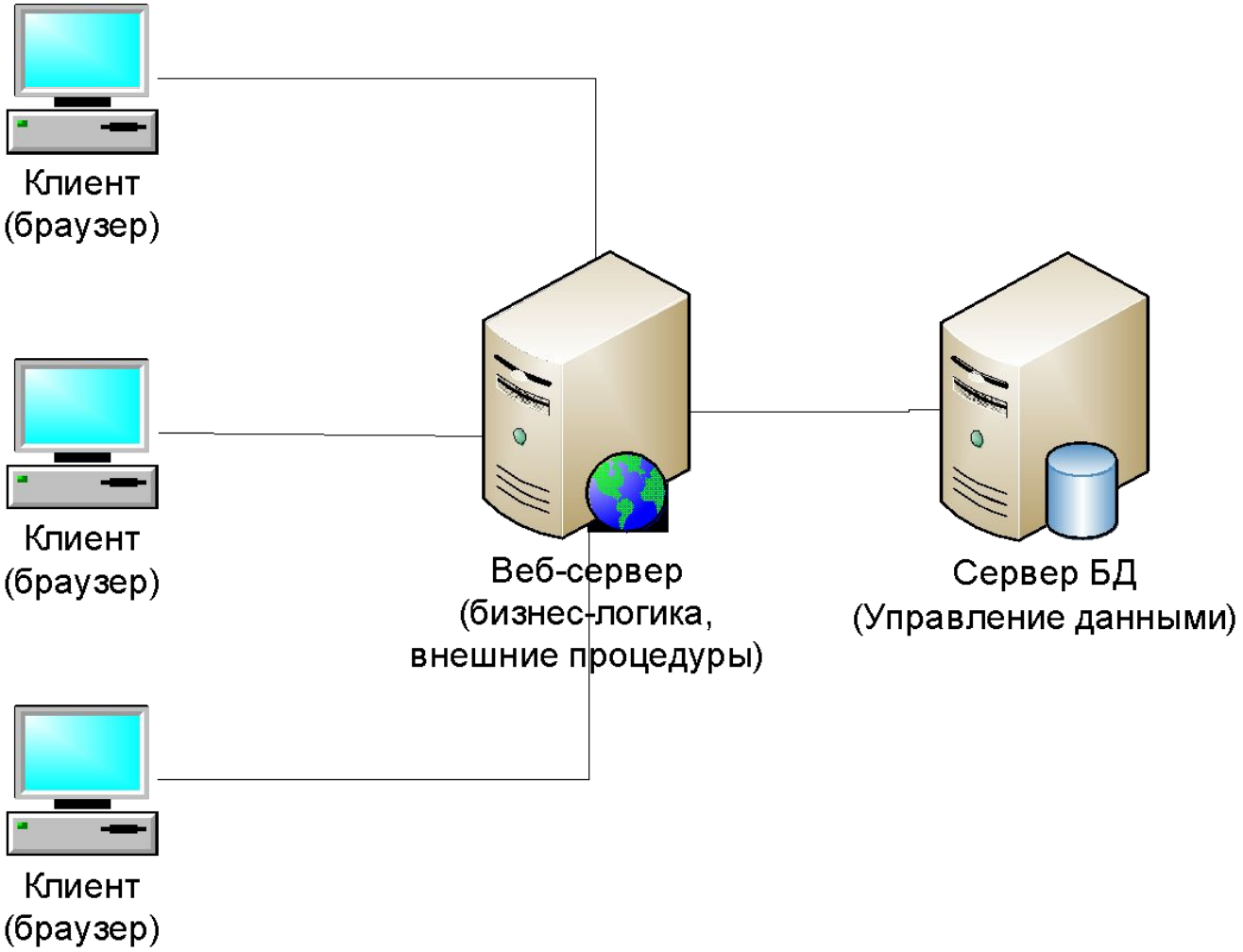
К этим средствам относят:

- интерфейсы доступа к базам данных ODBC и IDAPI;
- шлюзы для систем управления базами данных;
- протоколы и команды мониторов обработки транзакций;
- почтовые интерфейсы MAPI, VIM, MHS, X.400 и EDI;
- средства обмена сообщениями MOM;
- протоколы связывания и включения объектов OLE и динамического обмена данными DDE;
- протоколы удаленного вызова процедур RPC,
- средства коммуникационного ввода-вывода BSD Sockets и WinSock.

Таблица 1

Сх	Описание схемы	Клиент	Сервер - 1	Сервер - 2	Пример реализации
1	Централизованная многотерминальная система	PS	PL, BL, DL, DS, FS	-	Сервер Sun с X-терминалами в среде ОС Solaris
2	Локальная сеть ПК с файл серверными приложениями	PS, PL, BL, DL, DS	FS	-	Локальная сеть ПК в среде NetWare, программы на FoxPro, Clipper и др
3	Удаленный доступ к данным на сервере БД	PS, PL, BL, DL	DS, FS	-	Система клиент-сервер с доступом ПК к серверу БД Informix (NetWare)
4	Удаленный доступ к БД с использованием хранимых процедур	PS, PL, DL	BL, DS, FS	-	Система клиент-сервер, доступ ПК к серверу ORACLE в среде SCO Unix
5	Удаленный доступ к БД с разделением логики приложения	PS, PL, BL, DL	BL, DL, DS, FS	-	Система клиент-сервер, доступ ПК к серверу ORACLE на Sun (Solaris)
6	Удаленное представление данных с доступом к Unix-системе	PS, PL	BL, DL, DS,FS	-	Сеть ПК/станций, приложения на мониторе транзакций и СУБД в Unix
7	Удаленное управление файл-серверным приложением в сети	PS	PL, BL, DL, DS	FS	Связь удаленных ПК с сервером доступа WinView в сети для работы с СУБД FoxPro
8	Многотерминальный сервер приложений для доступа к СУБД	PS	PL, BL, DL	DS, FS	Сервер приложений на SCO Unix, доступ терминалов к ORACLE на HP
9	3-х звенная система на Unix с монитором транзакций	PS, PL	BL, DL	DS, FS	Сеть ПК, сервер приложений на TUXEDO и СУБД ORACLE в среде Solaris на Sun
10	3-х звенная система с монитором транзакций и разделением логики	PS, PL, BL	BL, DL	DS, FS	Аналогично предыдущему, но контроль данных выполняется на клиентских узлах

Архитектура Веб-приложений



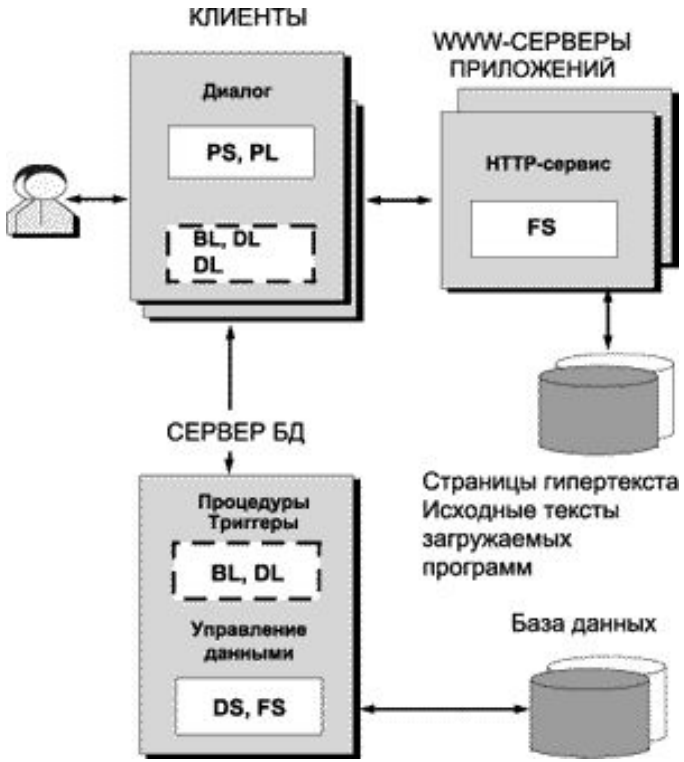
Архитектура Веб-приложений

- Отсутствие необходимости использовать дополнительное ПО на стороне клиента
- Возможность подключения практически неограниченного количества клиентов
- Централизованное место хранения данных
- Недоступность при отсутствии работоспособности сервера или каналов связи
- Достаточно низкая скорость Веб-

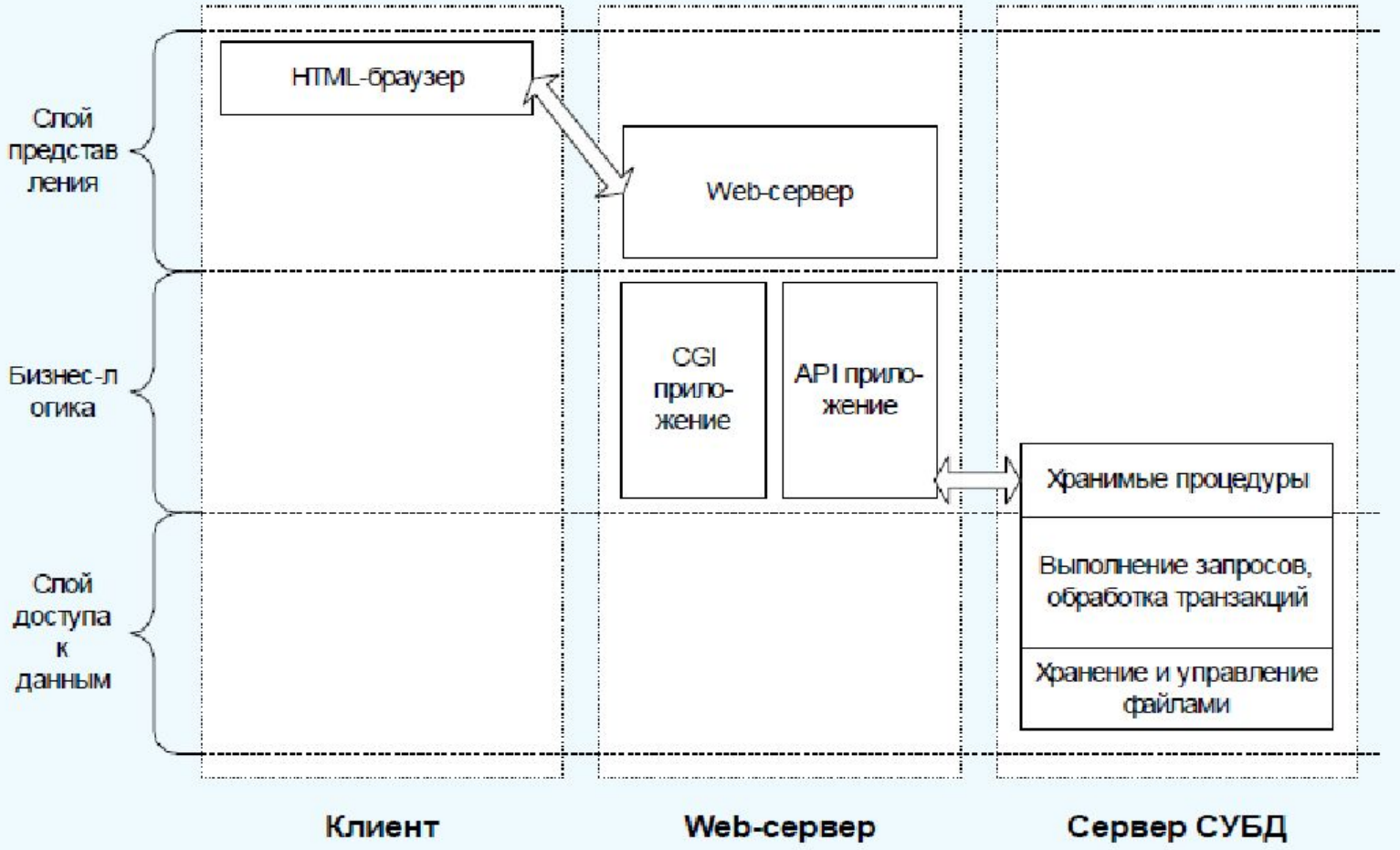
Системы на основе Internet/Intranet-технологий

Способ построения Internet/Intranet-приложений с использованием CGI-процедур

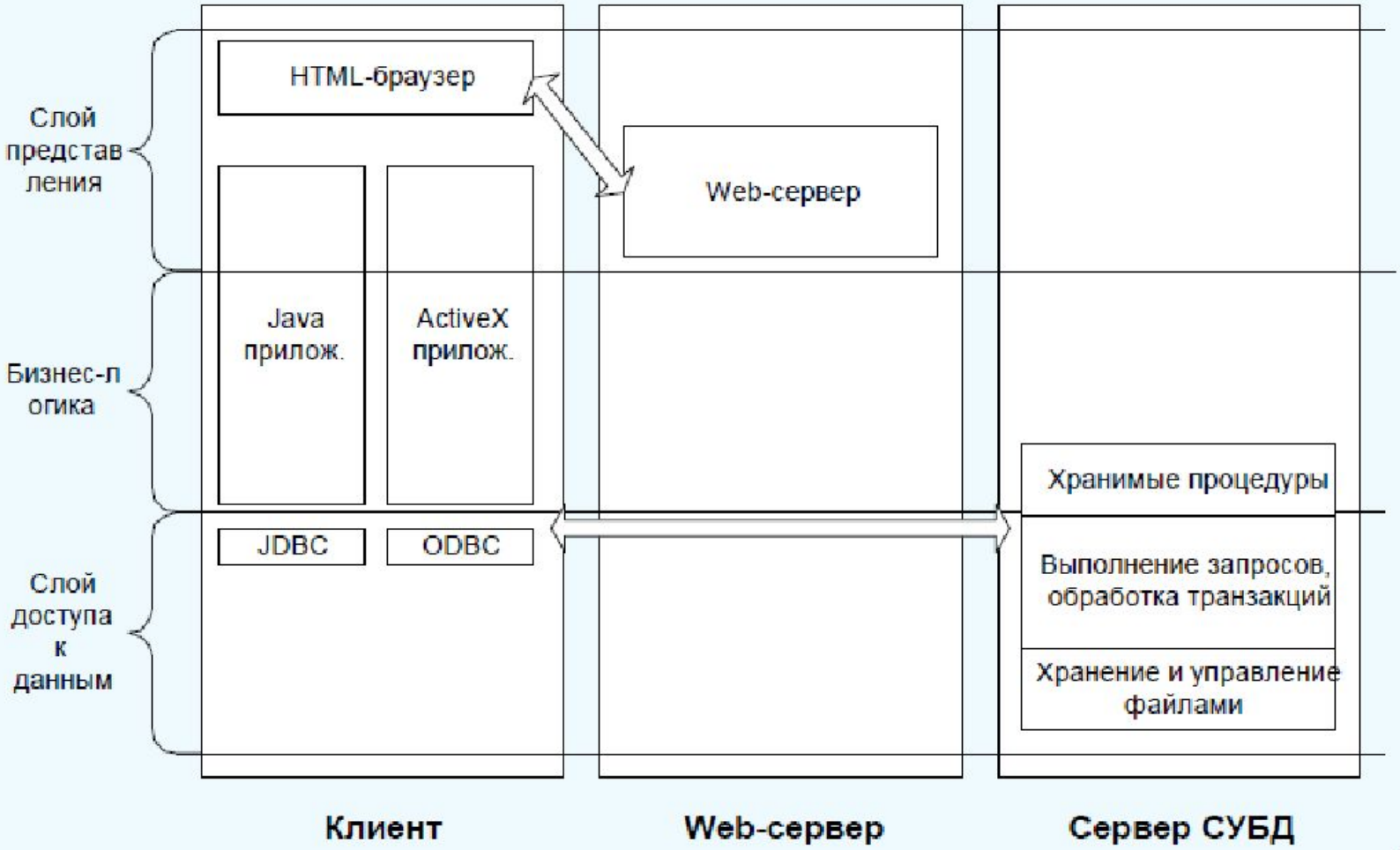
Способ построения Internet/Intranet-приложений с использованием загружаемых интерпретируемых программ



Модель доступа через Intra-/Internet & CGI/API



Модель Inter-/Intra- net с мигрирующими программами



Сервис-ориентированная архитектура

- **Сервис-ориентированная архитектура (SOA)** – модульный подход к разработке программного обеспечения, основанный на использовании сервисов со стандартизированными интерфейсами
- **Принципы SOA:**
  - архитектура не привязана к какой-то определенной технологии
  - независимость организации системы от используемой вычислительной платформы
  - независимость организации системы от применяемых языков программирования
  - использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним
  - организация сервисов как слабосвязанных компонентов для построения систем

Офисные системы

Средства автоматизации делопроизводства и документооборота

Выделяют следующие группы средств:

- средства автоматизации учрежденческой деятельности Office Automation;
- системы управления электронными документами EDMS;
- средства обеспечения коллективной работы Groupware;
- средства автоматизации документооборота Workflow.

Средства автоматизации учрежденческой деятельности

- текстовые редакторы для подготовки и корректировки документов;
- процессоры электронных таблиц для расчетов, анализа и графического представления данных;
- программы генерации запросов по образцу из различных БД;
- сетевые планировщики для назначения рабочих встреч и совещаний;
- средства разработки и демонстрации иллюстративных материалов для презентаций;
- словари и системы построчного перевода;
- программы посылки и приема факсов;
- локальную электронную почту для обмена сообщениями и пересылки файлов.

Эти средства могут представлять собой:

-отдельные пакеты (WinWord, WordPerfect, Excel, Lotus 1-2-3 и т. п.),

- интегрированные пакеты программ (**MS Works**) или согласованные наборы пакетов (**Microsoft Office** или **Corel Perfect Office**).

Для создания приложений на основе пакетов MS используют **макроязыки Basic** (Word Basic, Excel Basic и др.) или единый язык для расширения приложений, например **Visual Basic for Applications**.

Системы управления электронными документами EDMS

Выполняют следующие функции:

- создание электронной копии документа путем сканирования бумажного оригинала или ввода его текста в ПК;
- построение полнотекстового индекса документа;
- создание учетной карточки документа и заполнение атрибутов;
- организация тематических папок и/или гипертекстовых ссылок;
- блокировка документов при коллективном использовании;
- поиск документов по атрибутам и ключевым словам.

IBM Lotus - Notes,
Microsoft SharePoint
DOCS Open
Excalibur EFS

СЭД, популярные в России:
СЭД Дело,
Directum,
DocsVision,
1С: Документооборот,
CompanyMedia

Средства обеспечения коллективной работы

Эти средства ориентированы на автоматизацию работы небольшого коллектива и поддерживают корректное совместное использование информации группой пользователей.

Microsoft SharePoint

Lotus Notes

Средства автоматизации документооборота

Эти средства служат для автоматизации деятельности корпорации и поддерживают многопользовательскую обработку нескольких задач одновременно в синхронном или в асинхронном режиме.

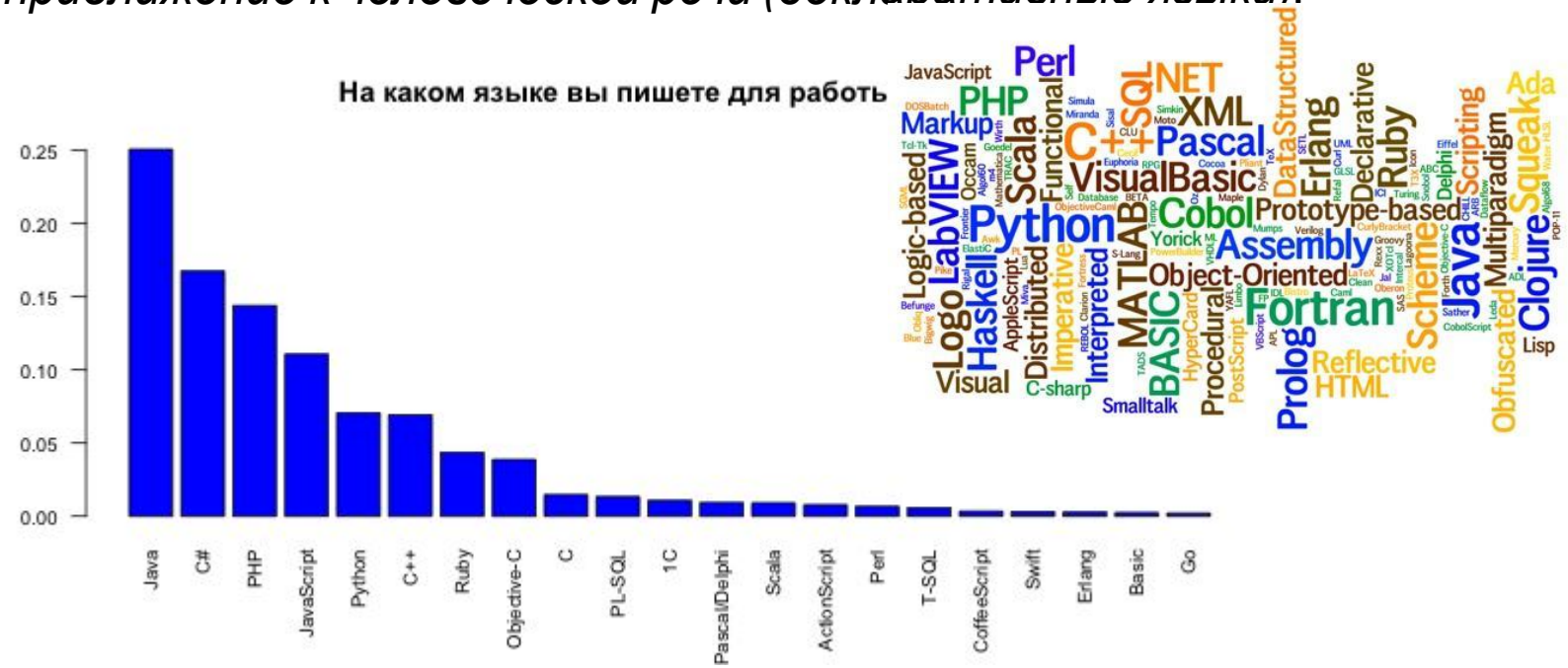
Для автоматизации документооборота и контроля исполнительской дисциплины применяются методы теории Workflow.

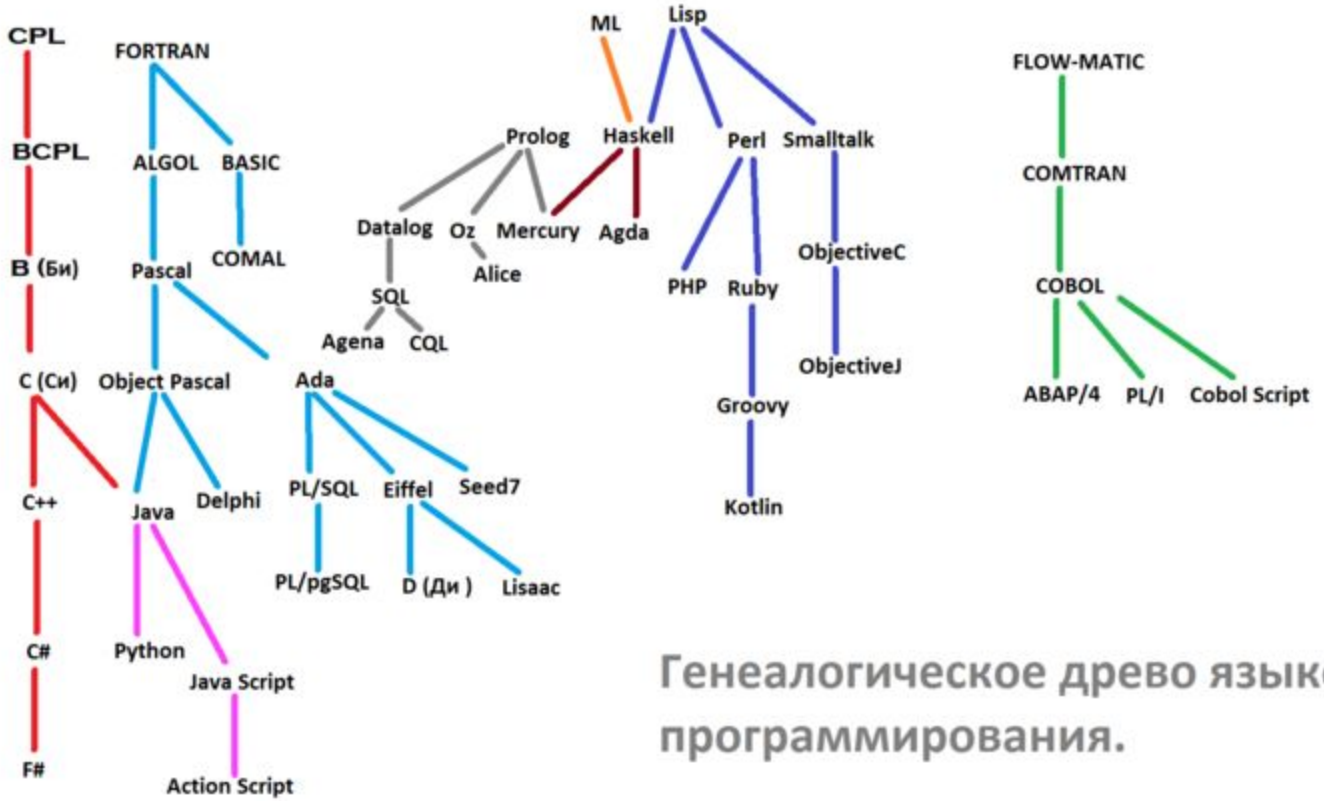
Традиционные системы программирования представлены средствами создания приложений на языках третьего поколения 3GL: C, Pascal, Basic и др (Примеры: ***Фортран, Кобол, Алгол, Pascal, Basic, C, C++, C#, Objective C, Java, Smalltalk, Delphi)***.

Четвертое поколение языков программирования

Проблемно-ориентированные языки, оперирующие конкретными понятиями узкой области. Часто относят: ***SQL, SGML (HTML, XML), Prolog, и многие другие узкоспециализированных декларативных языков***.

Основная отличительная особенность языка четвертого поколения: *приближение к человеческой речи (декларативные языки)*.





Генеалогическое древо языков программирования.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

PYTHON JAVA C PHP C++

JAVASCRIPT C# RUBY OBJECTIVE-C

Java является одним из самых популярных языков для разработки современных корпоративных приложений. Для Java создано много фреймворков, и потому разработчики могут создавать крутые приложения для широкого круга пользователей.
2Язык **C/C++**. Это универсальный выбор для разработки настольного программного обеспечения, игр с функцией аппаратного ускорения, а также приложений, требующих большого объёма памяти для работы.+
3**Objective C** – этот язык пригодится вам, если вы собираетесь заняться разработкой приложений для Apple Mac OS X, а также для айфонов, айпадов. Этот язык стал весьма востребованным еще со времен выхода первого айфона в 2007-ом году.
4Язык **C#** (Си шарп). Это самый популярный язык сейчас для разработки приложений для Windows, и очень популярный для мобильных устройств. А еще движок для разработки игр (Unity 3D) также использует C# в качестве одного из основных языков.
5Язык **PHP** является важным инструментом для создания современных веб-приложений. На PHP разработано большинство сайтов, ориентированных на большой объём данных. Например, системы управления контентом, типа WordPress.
6Каждый современный сайт использует **JavaScript**. Это ключевой язык для создания интерактивности сайта или построения пользовательских интерфейсов , создано много популярных JavaScript-фреймворков. + есть серверная часть (NodeJS)
7**Ruby on Rails.** Этот язык набирает популярность среди начинающих компаний, поскольку лучше подходит для быстрой разработки веб-приложений (по сравнению с Java или .Net)
8**Python.** Веб-приложения, статистика, анализ данных, пользовательские интерфейсы — для каждой бы задачи найдётся подходящий фреймворк в Python.
Перспективные языки программирования 2015-2025
Erlang. Главная фишка - параллельность! Крупные банки с миллионами пользователей используют Erlang.
Язык **R.** Широко используется для разработки статистического программного обеспечения.
Swift. Язык программирования Swift - новый, более быстрый и легкий путь разрабатывать под Mac и iOS, по сравнению с Objective-C.
Go. Этот язык разработан Google. Правильный подход к построению системного программного обеспечения на многоядерных компьютерах..

Система программирования ***Visual Basic*** предназначена для написания программ, работающих под управлением операционной системы Windows. Специальная версию Visual Basic - Visual Basic for Application (VBA) интегрирована во все компоненты пакета Microsoft Office, Microsoft Project и некоторые другие программы. VB используется :

- для создания простых автономных приложений;
- для создания макросов в офисных пакетах (Word, Excel, Access);
- как средство программирования для расширения систем документооборота и для создания утилит администрирования.

Система программирования ***Delphi*** применяется

- для создания расчетно-аналитических программ,
- для разработки динамических библиотек DLL, ***DLL*** (***dynamic-link library*** — «библиотека динамической компоновки», «динамически подключаемая библиотека»)
- для сопровождения и развития разработок, выполненных на Turbo и Borland Pascal, а также для быстрого прототипирования будущих приложений.

C++ применяется

- для расширения системного программного обеспечения,
- для разработки *крупных проектов*, *специальных приложений*,
- создания *библиотек и классов* для предметной области,
- разработки динамических библиотек *DLL*,
- создания *ПО для серверов приложений.*

Средства автоматизации проектирования приложений (CASE-средства)

Эти средства служат для анализа предметной области, для проектирования и генерации программ информационных приложений.
Характерными особенностями CASE-систем являются:

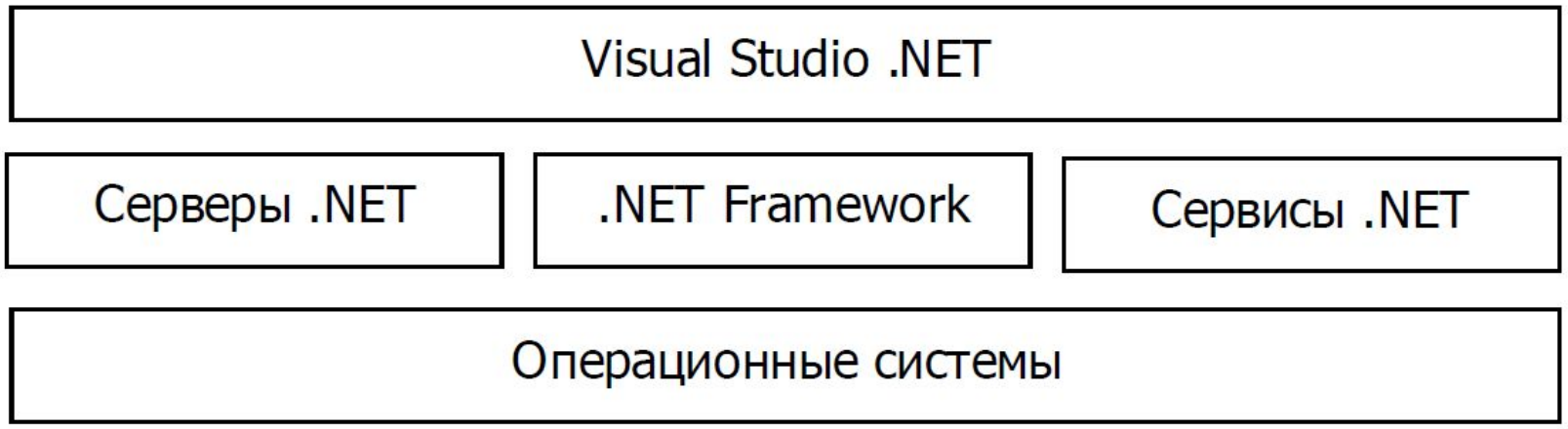
- наличие графических редакторов схем проекта,
- хранение описаний проектов в репозитарии объектов и генерация описания структуры и свойств БД, а также модулей приложения.

Могут существовать в виде отдельных утилит или интегрированной среды проектирования.
Различают следующие виды систем автоматизации проектирования приложений:

- независимые CASE-системы (например IDEF/Design);
- системы, интегрированные с СУБД (WestmountI-CASE for Informix и Oracle Designer/2000);
- системы проектирования БД (SILVERRUN и ERWin/ERX).

Платформа Microsoft.NET — интегрированная система (*инфраструктура*) средств разработки, развертывания и выполнения сложных (как правило, распределенных) программных систем (приложений), позволяющая использовать в разработке различные языки программирования – C#, Visual Basic .NET, C++ и т.п..

Состав платформы Microsoft.NET



− *операционные системы* корпорации Microsoft (Windows 2000/XP/ME/7/8/10),

− *серверы MS.Net* (.Net Enterprise Servers) — Exchange Server 2013, SQL Server и др.,

− *сервисы MS.Net* (.Net Building Block Services) — представляют собой готовые "строительные блоки" сложных программных систем, которые могут быть использованы через Интернет как сервисные услуги (Microsoft Passport),

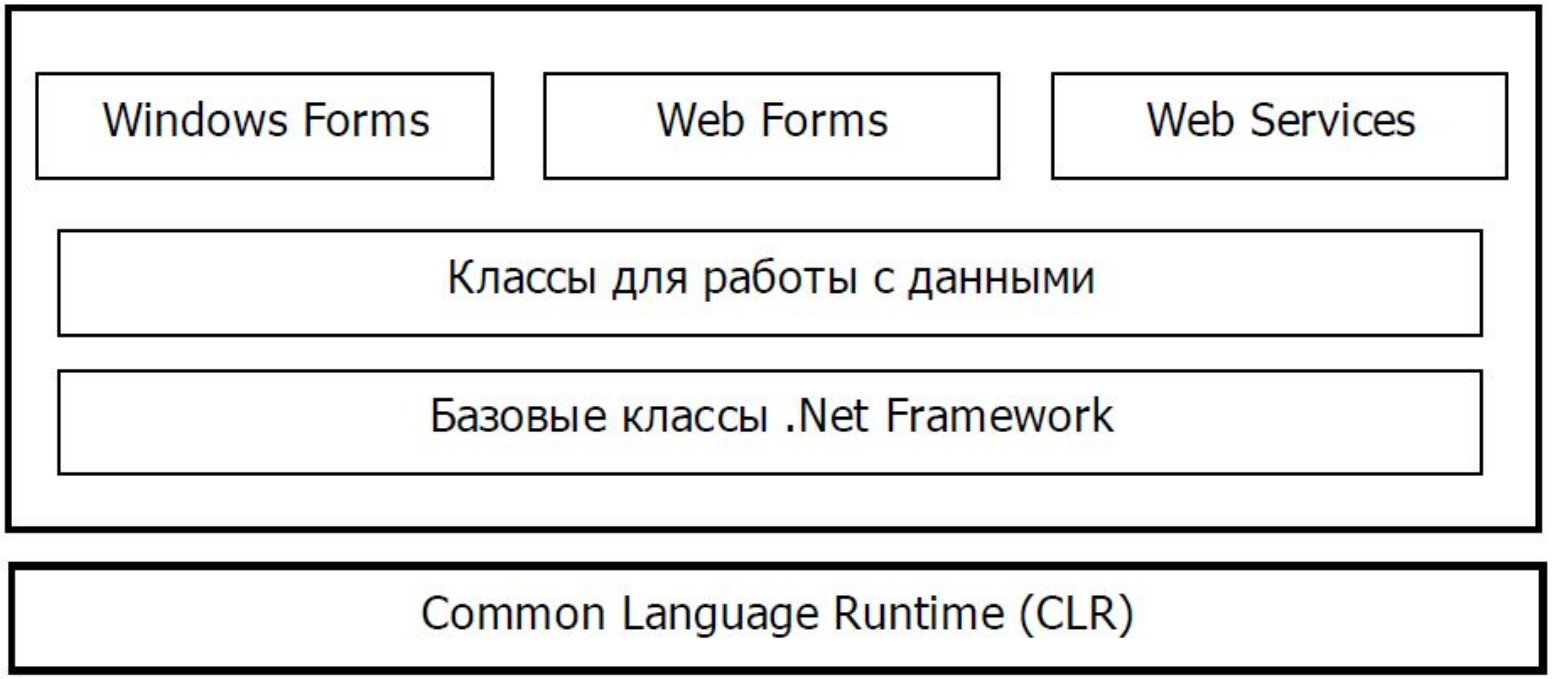
− *интегрированная среда разработки* приложений Visual Studio.NET (VS.Net) – верхний уровень MS.Net; обеспечивает возможность создания сложного ПО на основе платформы;

– *подсистема Microsoft.Net Framework* — центральная часть платформы MS.Net, обеспечивает возможность построения и исполнения .Net приложений.

Структура Microsoft.NET Framework

- ***общеязыковая среда выполнения CLR*** (*Common Language Runtime*);
- ***библиотеки базовых классов BCL*** (*Base Class Library*).

Архитектура MS.NET Framework



Ядро среды выполнения CLR - C:\Windows\system32\mscoree.dll

Библиотека классов FCL

главная сборка BCL представлена файлом:
C:\Windows\Microsoft.NET\Framework\версия\mscorlib.dll

По своему функциональному назначению в составе библиотек классов ***FCL*** могут быть выделены:

− набор *базовых классов*, обеспечивающих, например, работу со строками, ввод-вывод данных, многопоточность и т.п.,

− набор *классов для работы с данными*, предоставляющих возможность использования SQL-запросов, ADO.Net и обработки XML данных,

– набор *классов Windows Forms*, позволяющих создавать обычные Windows-приложения, в которых используются стандартные элементы управления Windows,

− набор *классов Web Forms*, обеспечивающих возможность быстрой разработки Web-приложений, в которых используется стандартный графический интерфейс пользователя,

− набор *классов Web Services*, поддерживающих создание распределенных компонентов-сервисов, доступ к которым может быть организован через Интернет.

Основные пространства имен FCL

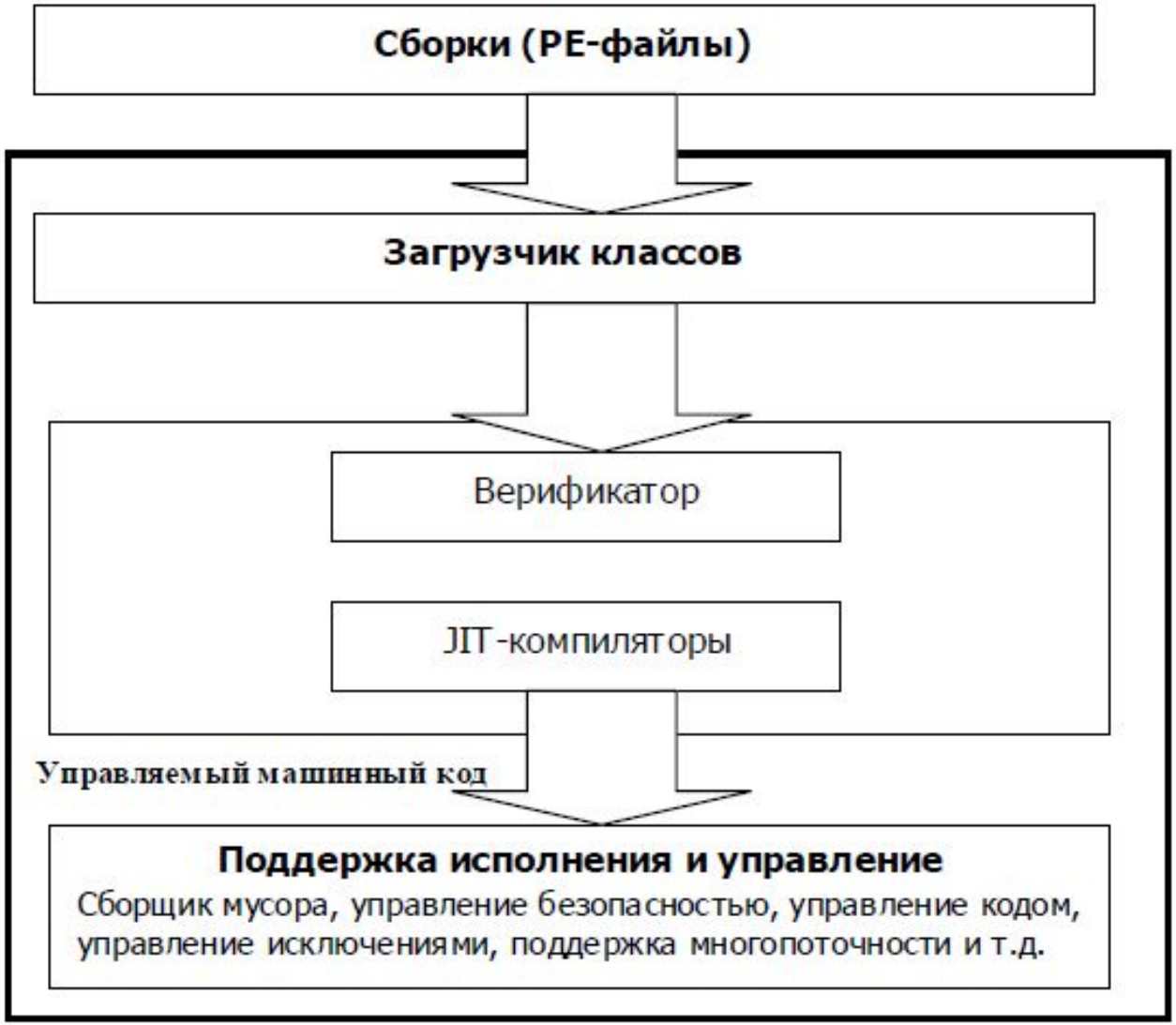
Пространство имен	Содержание
System	Фундаментальные типы данных и вспомогательные классы
System.Collections	Хэш-таблицы, массивы переменной размерности и другие контейнеры
System.Data	Классы ADO .NET для доступа к данным
System.Drawing	Классы для вывода графики (GDI+)
System.IO	Классы файлового и потокового ввода/вывода
System.Net	Классы для работы с сетевыми протоколами, например с HTTP
System.Reflection	Классы для чтения и записи метаданных
System.Runtime.Remoting	Классы для распределенных приложений
System.ServiceProcess	Классы для создания служб Windows
Sustem.Threading	Классы для создания и управления потоками
System.Web	Классы для поддержки HTTP
System.Web.Services	Классы для разработки web-сервисов
System.Web.Services.Protocols	Классы для разработки клиентов web-сервисов
System.Web.UI	Основные классы, используемые ASP .NET
System.Web.UI.WebControls	Серверные элементы управления ASP .NET
System.Windows.Forms	Классы для приложений с графическим интерфейсом пользователя
System.Xml	Классы для чтения и ввода данных в формате XML

Механизм исполнения приложений .NET

1 этап. Исходный код компилируется в код на промежуточном языке **MCIL** (*Microsoft Common Intermediate Language* — промежуточный язык Microsoft).
Программные файлы на языке IL наз-ся **сборками** (**assembly**), или — **переносимыми исполняемыми файлами** (Portable Executable или **PE**).

2 этап. ***Оперативная компиляция*** (Just-In-Time — JIT). Код *IL* компилируется в *низкоуровневый машинный (native) код* платформы исполнения.

Общая схема исполнения сборок



Реляционные базы данных

База данных (БД) – совместно используемый набор логически связанных данных (и их описание), предназначенный для удовлетворения информационных потребностей организации.

СУБД (система управления базами данных) – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также получать к ней контролируемый доступ.

Реляционная база данных (РБД) *представляет собой тело связанной информации, сохраняемой в двухмерных таблицах* **(строка и столбец)**. ***Строки РБД называют записями,*** *так как не находятся в какой-либо определенной последовательности.*

В отличие от строк (записи), столбцы таблицы (называемые также полями) упорядочиваются и именуются.

Уникальный столбец (или уникальная группа столбцов), используемый для идентификации каждой строки и хранения всех строк по отдельности, называется ***первичным ключом таблицы****.*

Имя	Телефон	Адрес
Gerry Farish	(415)365-8775	127 Primrose Ave.,SF
Celia Brock	(707)874-3553	246 #3rd St.,Sonoma
Yves Grillet	(762)976-3665	778 Modernas,Barcelona

Пациент	Доктор	Страховка	Баланс
Farish	Drume	B.C./B.S.	$272.99
Grillet	Halben	None	$44. 76
Brock	Halben	Health,Inc.	$9077.47

Реляционные связи между таблицами баз данных

Выделяют три разновидности связи между таблицами базы данных:

"один–ко–многим" - одной записи родительской таблицы может соответствовать несколько записей дочерней;

"один–к–одному" - одной записи в родительской таблице соответствует одна запись в дочерней;

"многие–ко–многим". Отношение "многие–ко–многим" применяется в следующих случаях:

- одной записи в родительской таблице соответствует более одной записи в дочерней;
- одной записи в дочерней таблице соответствует более одной записи в родительской.

Всякую связь "многие–ко–многим" в реляционной базе данных необходимо заменить на связь "один–ко–многим" (одну или более) с помощью введения дополнительных таблиц.

Язык SQL

SQL (Structured Query Langauge - язык структурированных запросов) является стандартным языком, используемым для работы с реляционными базами данных.

Основные категории команд языка SQL:

DDL – язык определения данных;

DML – язык манипулирования данными;

DQL – язык запросов;

DCL – язык управления данными;

команды администрирования данных;

команды управления транзакциями

Определение структур базы данных (DDL)

Язык определения данных (Data Definition Language, DDL) позволяет создавать и изменять структуру объектов базы данных, например, создавать и удалять таблицы. Основными командами языка DDL являются следующие: CREATE TABLE, ALTER TABLE (изменить таблицу), DROP TABLE (удалить таблицу), CREATE INDEX, ALTER INDEX, DROP INDEX.

Манипулирование данными (DML)

Язык манипулирования данными (Data Manipulation Language, DML) используется для манипулирования информацией внутри объектов реляционной базы данных посредством трех основных команд: INSERT (вставить), UPDATE (обновить), DELETE.

Выборка данных (DQL)

Язык запросов DQL включает всего одну команду SELECT. Эта команда вместе со своими многочисленными опциями и предложениями используется для формирования запросов к реляционной базе данных.

Язык управления данными (DCL - Data Control Language)

Команды управления данными позволяют управлять доступом к информации, находящейся внутри базы данных. Как правило, они используются для создания объектов, связанных с доступом к данным, а также служат для контроля над распределением привилегий между пользователями. Команды управления данными следующие: GRANT (*предоставить*), REVOKE *(аннулировать)*.

Команды администрирования данных

С помощью команд администрирования данных пользователь осуществляет контроль за выполняемыми действиями и анализирует операции базы данных ; они также могут оказаться полезными при анализе производительности системы.

Команды управления транзакциями

Существуют следующие команды, позволяющие управлять транзакциями базы данных:

SET TRANSACTION – начинает транзакцию и устанавливает ее базовые характеристики.

COMMIT – заканчивает текущую транзакцию сохранением изменений в базе данных и начинает новую транзакцию

ROLLBACK – заканчивает текущую транзакцию отменой изменений в базе данных и начинает новую транзакцию

SAVEPOINT – устанавливает контрольные точки (точки прерывания) для транзакции, разрешая неполный откат.

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных

CREATE SCHEMA - создать схему базы данных
DROP SHEMA - удалить схему базы данных
CREATE TABLE - создать таблицу
ALTER TABLE - изменить таблицу
DROP TABLE - удалить таблицу
CREATE DOMAIN - создать домен
ALTER DOMAIN - изменить домен
DROP DOMAIN - удалить домен
CREATE COLLATION - создать последовательность
DROP COLLATION - удалить последовательность
CREATE VIEW - создать представление
DROP VIEW - удалить представление
CREATE INDEX, ALTER INDEX, DROP INDEX.

Операторы DML (Data Manipulation Language) - операторы манипулирования данными

INSERT - добавить строки в таблицу
UPDATE - изменить строки в таблице
DELETE - удалить строки в таблице
COMMIT - зафиксировать внесенные изменения
ROLLBACK - откатить внесенные изменения

Операторы выборки данных (DQL)

SELECT - отобрать строки из таблиц

Операторы защиты и управления данными (DCL - Data Control Language)

CREATE ASSERTION - создать ограничение
DROP ASSERTION - удалить ограничение
GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами
REVOKE - отменить привилегии пользователя или приложения

Типы данных языка SQL

Тип данных	**Объявления**
Символьный	CHAR | VARCHAR | NCHAR | NVARCHAR
Текстовый	TEXT | NTEXT
Целочисленный	INTEGER | SMALLINT | TINYINT | BIGINT
Десятичный нецелочисленный	NUMERIC | DECIMAL
Приблизительный нецелочисленный	FLOAT | REAL
Дата/время	DATE | TIME | DATETIME | SMALLDATETIME | DATETIME2 | DATETIMEOFFSET
Денежный	MONEY | SMALLMONEY
Двоичный	BINARY | VARBINARY | BIT
Пространственный	GEOGRAPHY | GEOMETRY
Специальные	TIMESTAMP | UNIQUEIDENTIFIER | SYSNAME | SQL_VARIANT | TABLE | CURSOR

Символьные типы данных

CHAR (длина),
VARCHAR (длина),
NCHAR (длина),
NVARCHAR (длина).

Текстовые типы данных

TEXT и NTEXT

Целочисленные типы данных

INT (INTEGER) — 4 байта (диапазон от -231 до 231-1),
SMALLINT — 2 байта (диапазон от -215 до 215-1),
TINYINT — 1 байт (диапазон от 0 до 255)
BIGINT — 8 байт (диапазон от -263 до 263-1).

Десятичные нецелочисленные типы данных

DECIMAL [(точность[,масштаб])] или DEC,
NUMERIC [(точность[,масштаб])].

Приблизительные нецелочисленные типы данных

FLOAT (точность до 15 цифр, 8 байт),
REAL (точность до 7 цифр, 4 байта).

Типы данных Дата/Время

DATE — для представления даты используется 3 байта,

DATETIME — для представления даты и времени используется 8 байт,

SMALLDATETIME — для представления даты используется 4 байта,

DATETIME2 [(доли секунды)] — используется 6 байт для представления точности меньше 3 цифр, 7 байт — для точности в 3 и 4 цифры; для представления любых других значений точности требуется 8 байт,

DATETIMEOFFSET [(доли секунды)] — используется 10 байт, по умолчанию используется фиксированная точность 100 нс.

TIME [(доли секунды)] — используется 5 байт, по умолчанию используется фиксированная точность 100 нс.

Денежные типы данных

MONEY

SMALLMONEY

Двоичные типы данных

BINARY [(n)] — двоичные данные фиксированной длины размером в n байт, где n — значение от 1 до 8000. Размер при хранении составляет n байт.

VARBINARY [(n | max)] — двоичные данные с переменной длиной. n может иметь значение от 1 до 8000. max указывает, что максимальный размер при хранении составляет 2^31-1 байт. Размер хранения — это фактическая длина введенных данных плюс 2 байта. Введенные данные могут иметь размер 0 символов.

BIT — тип данных позволяет хранить один бит, который принимает значения 0 или 1.

Пространственные типы данных

GEOGRAPHY — тип данных SQL Server сохраняет эллиптические данные (в системе координат круглой земли), такие как координаты широты и долготы в системе GPS. Географический пространственный тип данных GEOGRAPHY в SQL Server реализуется как тип данных среды .NET CLR.

GEOMETRY — этот тип представляет данные в евклидовом пространстве (плоской системе координат). Плоский пространственный тип данных GEOMETRY в SQL Server реализуется как тип данных среды CLR.

Специальные типы данных

Тип данных **TIMESTAMP** применяется в качестве индикатора изменения версии строки в пределах базы данных.

Тип данных **UNIQUEIDENTIFIER** используется для хранения глобальных уникальных идентификационных номеров.

Тип данных **SYSNAME** предназначен для идентификаторов объектов.

Тип данных **SQL_VARIANT** позволяет хранить значения любого из поддерживаемых SQL Server типов данных за исключением TEXT, NTEXT, IMAGE и TIMESTAMP.

Тип данных **TABLE**, подобно временным таблицам, обеспечивает хранение набора строк, предназначенных для последующей обработки.

Тип данных **CURSOR** нужен для работы с такими объектами, как курсоры, и может быть востребован только для переменных и параметров хранимых процедур. Курсор позволяет клиентским приложениям работать не с полным набором данных, а лишь с одной или несколькими строками.

Создание пользовательского типа данных

```
sp_addtype [@typename=]type,[@phystype=]
system_data_type
[,[@nulltype=]'null_type']
```

Тип данных system_data_type выбирается из следующей таблицы

image	smalldatetime	decimal	bit
text	real	'decimal[(p[,s])]'	'binary(n)'
uniqueidentifier	datetime	numeric	'char(n)'
smallint	float	'numeric[(p[,s])]'	'nvarchar(n)'
int	'float(n)'	'varbinary(n)'	
	ntext	'varchar(n)'	'nchar(n)'

Пример :

Создание пользовательского типа данных bir.

```
EXEC sp_addtype bir, DATETIME, 'NULL'
```

Использование пользовательского типа данных bir при создании таблицы.

```
CREATE TABLE tab
(id_n  INT IDENTITY(1,1) PRIMARY KEY,
names  VARCHAR(40),
birthday  BIR)
```

Удаление пользовательского типа данных происходит в результате выполнения процедуры

```
sp_droptype type: EXEC sp_droptype 'bir'
```

Получение информации о типах данных

```
SELECT * FROM  systypes
```

Преобразование типов

```
CAST(выражение AS тип_данных)
CONVERT(тип_данных[(длина)],
    выражение [,стиль])
```

Пример:

```
DECLARE @d DATETIME
DECLARE  @s CHAR(8)
SET @s='29.10.01'
SET @d=CAST(@s AS DATETIME)
```

Запись SQL-операторов

Оператор SQL состоит из ***зарезервированных слов***, а также из слов, ***определяемых пользователем***.

Зарезервированные слова являются постоянной частью языка SQL и имеют фиксированное значение.

Слова, определяемые пользователем, задаются им самим (в соответствии с синтаксическими правилами) и представляют собой идентификаторы или имена различных объектов базы данных.

На формат идентификатора накладываются следующие ограничения:

- идентификатор может иметь длину до 128 символов;
- идентификатор должен начинаться с буквы;
- идентификатор не может содержать пробелы.

Символ	Обозначение
::=	Равно по определению
|	Необходимость выбора одного из нескольких приведенных значений
<…>	Описанная с помощью метаязыка структура языка
{…}	Обязательный выбор некоторой конструкции из списка
[…]	Необязательный выбор некоторой конструкции из списка
[,…n]	Необязательная возможность повторения конструкции от нуля до нескольких раз

Основные объекты структуры базы данных SQL-

Tables	Таблицы базы данных, в которых хранятся собственно данные
Views	Просмотры (виртуальные таблицы) для отображения данных из таблиц
Stored Procedures	Хранимые процедуры
Triggers	Триггеры – специальные хранимые процедуры, вызываемые при изменении данных в таблице
User Defined function	Создаваемые пользователем функции
Indexes	Индексы – дополнительные структуры, призванные повысить производительность работы с данными
User Defined Data Types	Определяемые пользователем типы данных
Keys	Ключи – один из видов ограничений целостности данных
Constraints	Ограничение целостности – объекты для обеспечения логической целостности данных
Users	Пользователи, обладающие доступом к базе данных
Roles	Роли, позволяющие объединять пользователей в группы
Rules	Правила базы данных, позволяющие контролировать логическую целостность данных
Defaults	Умолчания или стандартные установки базы данных

Создание Запросов

Запрос (query) – это средство выбора необходимой информации из базы данных.

Запросы могут находиться:

-на стороне клиентского приложения - запрос прописывается внутри объекта связи. В этом случае клиентское приложение не зависит от файла данных. Файл данных содержит только таблицы. Поэтому, мы легко можем модифицировать клиентское приложение, не затрагивая файл данных, но в этом случае запрос передается серверу через сеть, что может вызвать проблемы с безопасностью.

-на стороне сервера - сам запрос выступает в качестве компонента БД, вся передача информации происходит внутри файла данных, т.е. внутри самого сервера, клиентскому приложению только передаются результаты выполнения запроса. В этом случае обеспечивается высокая защита данных, но в случае изменения запроса придется менять сам файл данных.

1. Статические запросы – их структура неизменна в ходе работы с программой;
2. Динамические запросы - могут меняться в зависимости от ситуации. Обычно динамические запросы могут быть реализованы только при помощи запросов, выполняющихся на стороне клиента. На стороне сервера они реализуются в виде *хранимых процедур*.

```
SELECT [ALL|DISTINCT]
[TOP|PERCENT n]
<Список полей>
[INTO <Имя новой таблицы>]
[FROM <Имя таблицы >]
[WHERE <Условие>]
[GROUP BY <Поле>]
[ORDER BY <Поле > [ASC|DESC]]
[COMPUTE AVG|COUNT|MAX|MIN|SUM(<Выражение>)]
```

Примеры

```
SELECT TOP 20 PERCENT *.Сотрудники, Должность.Должности
FROM Сотрудники, Должности
WHERE Код.Сотрудники = Код.Должности
COMPUTE COUNT (ФИО.Сотрудники)
```

```
SELECT ALL Операция, Сумма
INTO [Сделки за Май]
FROM Операции
WHERE Месяц = 'Май'
GROUP BY Операция
ORDER BY Сумма
COMPUTE SUM (Сумма)
```

Вычисления с помощью SELECT

```
SELECT AVG(возраст) FROM Студенты

SELECT COUNT(ФИО) FROM Студенты

SELECT Top 100 * FROM Студенты
```

Вычисления с помощью SELECT

Математические функции:

ABS (numeric) - модуль числа;

· ACOS/ASIN/ATAN (Float) - арккосинус, арксинус, арктангенс в радианах;

· COS/SIN/TAN/COT (Float) - косинус, синус, тангенс, котангенс;

· CEILING (Numeric) - наименьшее целое, большее или равное параметру в скобках;

· DEGREES (Numeric) - преобразует радианы в градусы;

· EXP(Float) - экспонента, ex;

· FLOOR(Numeric) - наибольшее целое меньшее или равное выражению **numeric ;**

· LOG(Float) - натуральный логарифм ln;

· LOG10(Float) - десятичный логарифм log10;

· PI () - число пи;

· POWER (Numeric,y) - возводит выражение **Numeric в степень y;**

· RADIANS (Numeric) - преобразует градусы в радианы;

· RAND () - генерирует случайное число типа данных **Float, расположенное** между нулем и единицей;

· ROUND (Numeric, Длина) - округляет выражение **Numeric до заданной Длины (количество знаков после запятой);**

· SIGN (Numeric) - выводит знак числа +/- или ноль;

· SQUARE (Float) - вычисляет квадрат числа **Float;**

· SQRT (Float) - вычисляет квадратный корень числа **Float.**

Примеры использования математических функций:

Вычисления с помощью SELECT

Строковые функции

Строковые функции позволяют производить операции с одной или несколькими строками.

· 'Строка1'+ 'Строка2' присоединяет **Строку1 к Строке2;**

· ASCII(Char) - возвращает **ASCII код с самого левого символа выражения Char ;**

· CHAR(Int) - выводит символ соответствующий **ASCII коду в выражении Int;**

· CHARINDEX(Образец, Выражение) - выводит позицию **Образца выражения, то есть где находится Образец в Выражении;**

· DIFFERENCE(Выражение1, Выражение2) - сравнивает два выражения, выводит числа от 0 до 4: 0 - выражения абсолютно различны; 4 - выражения абсолютно идентичны. Оба выражения типа данных **Char;**

· LEFT(Char, Int) - выводит из строки **Char Int символов слева;**

· RIGHT(Char, Int) - выводит из строки **Char Int символов справа;**

· LTRIM(Char) - удаляет из строки **Char пробелы слева;**

· RTRIM(Char) - удаляет из строки **Char пробелы справа;**

· WCHAR(Int) - выводит выражение **Int в формате Unicode;**

· REPLACE(Строка1, Строка2, Строка3) - меняет в **Строке1 все элементы Строка2 на элементы Строка3;**

· REPLICATE(Char, Int) - повторяет строку **Char Int раз;**

· REVERSE(Char) - производит инверсию строки **Char, то есть располагает** символы в обратном порядке;

· SPACE(Int) - выводит **Int пробелов;**

Хранимые процедуры

Хранимые процедуры - SQL запрос, хранимый на стороне сервера и этот запрос имеет параметры, которые подставляются внутрь SQL кода.

В основном хранимая процедура либо реализует:

-связь между таблицами,

- осуществляет фильтрацию данных,

- производит вычисления.

В случае связей между таблицами одна таблица всегда выступает первичной, а другая вторичной, связь происходит при помощи полей связи. При связи сопоставляются записи с одинаковыми значениями полей связи.

Хранимые процедуры

Чтобы посмотреть информацию о хранимой процедуре необходимо выполнить команду:

EXEC SP_HELPTEXT <Имя процедуры>
Хранимые процедуры могут быть запущены следующей командой

EXEC <Имя процедуры> [<Параметр1>, <Параметр2>, …]

Здесь:
<Имя процедуры> - имя выполняемой процедуры.
Параметр1, Параметр2, … - значение параметров.

Пример

EXEC СрБАЛЛ 4

Пользовательские функции

1. Скалярные функции - функции, которые возвращают число или текст, то есть одно или несколько значений;

2. Табличные функции - функции, которые выводят результат в виде таблицы.

Пользовательские функции

Синтаксис:

CREATE FUNCTION <Имя функции>
([@<Параметр1> <Тип1>[=<Значение1>],
@<Параметр2> <Тип2>[=<Значение2>], . . .])
RETURNS <Тип>/TABLE
AS
RETURN([<Команды SQL>])

Здесь:

· Имя функции - имя создаваемой пользовательской функции.
· Параметр1, Параметр2, .. - параметры передаваемые в функцию.
· Значение1, Значение2, … - значения параметров по умолчанию.
· Тип1, Тип2, . .. - типы данных параметров.

Пример:

```
CREATE FUNCTION Среднее
(@X1 Int,@X2 Int,@X3 Int)
RETURNS Real
AS
BEGIN
DECLARE @Res Real
SET @Res =(@X1+@X2+@X3)/3
RETURN @Res
END
```

```
CREATE FUNCTION Среднее
(@X1 Int,@X2 Int,@X3 Int)
RETURNS Real
AS
RETURN (SELECT (@X1+@X2+@X3)/3)
```

Пользовательские функции

Созданная функция, вычисляющая среднее 6, 3 и 3, запускается следующим образом:

```
SELECT Среднее (6, 3, 3)
```

Пользовательские функции

Табличная пользовательская функция:

```
CREATE FUNCTION Возраст
(@CurDate Date)
RETURNS TABLE
AS
RETURN (SELECT ФИО, [Дата рождения], Возраст = DATEDIFF
(yy,[Дата рождения], @CurDate)
FROM Студенты)
```

Данная функция вызывается следующим образом:

```
SELECT * FROM Возраст ('12/17/2007')
```

Триггеры

Триггеры – это объекты, которые выполняют команды SQL если происходят какие-либо действия с таблицей (Например: добавление, изменение или удаление записей).
При помощи триггеров можно организовать автоматическое удаление записей из вторичной таблицы при удалении связанной с ними записи из первичной таблицы.

2 вида триггеров:

1. Триггеры выполняемые после события, произошедшего с таблицей.
   Применяется для обработки событий таблиц.
   2. Триггеры выполняемые вместо события, происходящего с таблицей. В этом случае событие (добавление, изменение или удаление записей) не выполняется, а вместо него выполняются SQL команды заданные внутри триггера.
   Применяется для обеспечения целостности данных, то есть удаление записей из вторичной таблицы при удалении связанной с ними записи из первичной таблицы.

Триггеры создаются для конкретной таблицы и выполняются автоматически, если с таблицей, для которой они были созданы, происходит событие (добавление, изменение или удаление записей).

Триггеры

Синтаксис:

CREATE TRIGGER <Имя триггера>
ON <Имя таблицы>
FOR <INSERT|UPDATE|DELETE>
[WITH ENCRYPTION]
AS <Команды SQL>

Пример:

CREATE TRIGGER Добавление
ON Студенты
FOR AFTER INSERT
AS PRINT 'Запись добавлена'

Здесь:
· Имя триггера - это имя создаваемого триггера.
· Имя таблицы - имя таблицы, для которой создаётся триггер.
· Если используется параметр AFTER, то триггер выполняется после события, а если параметр INSTEAD OF, то выполняется вместо события.
· Параметры INSERT, UPDATE и DELETE определяют событие, при котором (или вместо которого) выполняется триггер.
· Параметр WITH ENCRYPTION - предназначен для включения шифрования данных при выполнении триггера.
· Команды SQL - это SQL команды, выполняемые при активизации триггера.

Триггеры

Примеры:

```
CREATE TRIGGER Изменение
ON Студенты
FOR AFTER UPDATE
AS PRINT 'Запись изменена‘

CREATE TRIGGER Удаление
ON Студенты
FOR AFTER DELETE
AS PRINT 'Запись удалена‘

CREATE TRIGGER УдалениеСтудента
ON Студенты
INSTEAD OF DELETE
AS
BEGIN
DELETE Оценки
FROM deleted
WHERE deleted.[Код студента]=Оценки.[Код студента]
DELETE Студенты
FROM deleted
WHERE deleted.[Код студента]=Студенты.[Код студента]
END
```

Управляющие конструкции SQL

Группировка двух и более команд в единый блок осуществляется с использованием ключевых слов BEGIN и END:

<блок_операторов>::=
BEGIN
{ sql_оператор | блок_операторов }
END

Нередко определенная часть программы должна выполняться только при реализации некоторого логического условия. Синтаксис условного оператора показан ниже:

<условный_оператор>::=
IF лог_выражение { sql_оператор | блок_операторов }
[ELSE
{sql_оператор | блок_операторов }]

Циклы организуются с помощью следующей конструкции:
<оператор_цикла>::=WHILE лог_выражение { sql_оператор | блок_операторов }
[BREAK] { sql_оператор | блок_операторов } [CONTINUE]

Управляющие конструкции SQL

Циклы организуются с помощью следующей конструкции:

<оператор_цикла>::=
WHILE лог_выражение
    { sql_оператор | блок_операторов }
    [BREAK]
    { sql_оператор | блок_операторов }
    [CONTINUE]

Цикл можно принудительно остановить, если в его теле выполнить команду BREAK. Если же нужно начать цикл заново, не дожидаясь выполнения всех команд в теле, необходимо выполнить команду CONTINUE.

Для замены множества одиночных или вложенных условных операторов используется следующая конструкция:

<оператор_поливариантных_ветвлений>::=
CASE входное_значение
WHEN {значение_для_сравнения |
    лог_выражение } THEN
    вых_выражение [,...n]
[ELSE иначе_вых_выражение]
END

Если входное значение и значение для сравнения совпадают, то конструкция возвращает выходное значение. Если же значение входного параметра не найдено ни в одной из строк WHEN...THEN, то тогда будет возвращено значение, указанное после ключевого слова

Языки HTML, XHTML, CSS. Структура HTML-документа

HTML – HyperText Markup Language
XML – eXtansible Markup Language
История:
HTML 2.0;
HTML 3.0;
HTML 3.2 — 1997;
HTML 4.0 — 1997;
HTML 4.01 — 1999;
HTML 5.0 — 2014;
HTML 5.1 — 2016.

XHTML 1.0
XHTML 1.1
XHTML Basic и XHTML MP
XHTML 2.0
DHTML (Dynamic HTML) — способ создания интерактивного веб-сайта, использующий сочетание статичного языка HTML, встраиваемого скриптового языка JavaScript, CSS (каскадных таблиц стилей) и DOM (объектной модели документа).
MHTML (MIME HTML) — архивный формат веб-страниц, используемый для комбинирования кода HTML и ресурсов, которые обычно представлены в виде внешних ссылок (изображения, анимации Flash, Java-апплеты и аудиофайлы) в один файл.

Языки HTML, XHTML, CSS. Структура HTML-документа

 Привет

Тег
Элемент

 Привет

Атрибут
Значение атрибута

CSS (Cascading Style Sheets) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.
CSS1;
CSS2;
CSS3;
CSS4.
CSS-фреймворк — фреймворк, созданный для упрощения работы верстальщика. CSS-библиотеки обычно имеюn вид внешнего css-файла, «подключаются» к проекту (добавляются в заголовок веб-страницы).

Языки HTML, XHTML, CSS. Структура HTML-документа

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>HTML Document</title>
-----
</head>
<body>
------
</body>
</html>
```