

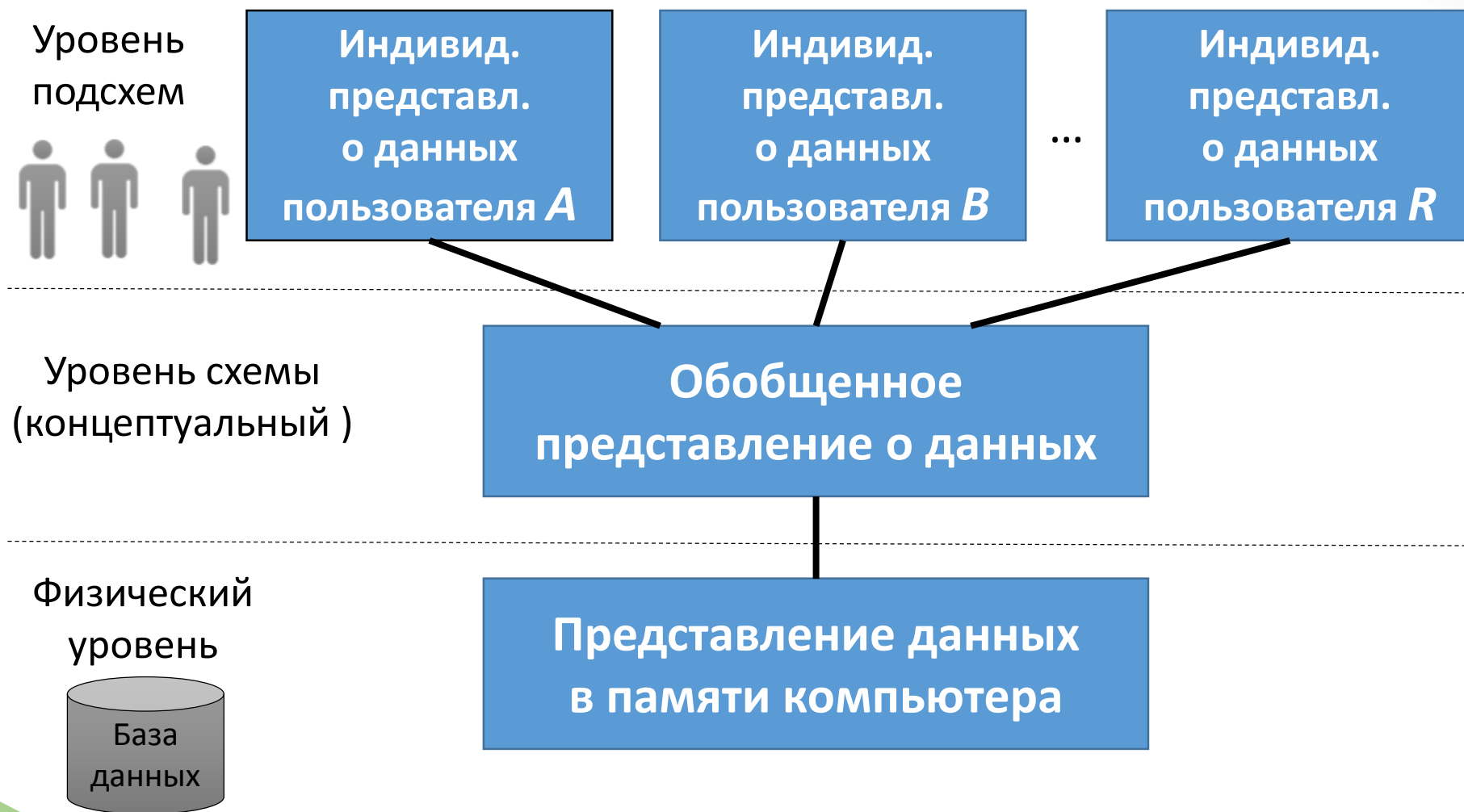
Архитектуры системы баз данных

Технологии баз данных. Лекция 2

Архитектура систем баз данных ANSI/SPARC

- Научная группа ANSI/SPARC (Study Group on Data Base Management Systems) была организована в 1972 комитетом Standards Planning and Requirements Committee (SPARC) института American National Standards Institute on Computers and Information Processing (ANSI/X3).
- Целью группы являлось определение областей технологии баз данных, нуждающихся в стандартизации, и выработка набора рекомендуемых действий в каждой из этих областей.
- В 1975-78 группа разработала архитектуру системы баз данных.

Уровни абстракции данных



Физический уровень

- Представление схемы в виде совокупности файлов определенной структуры.



Уровень схемы

- **Схема** описывает логическую структуру базы данных в терминах некоторой модели данных.

Поставщики

КодП	ИмяП	Город	...
S1	Иванов	Москва	
S2	Петров	Челябинск	
...			

Детали

КодД	ИмяД	Цена	...
P1	Болт	10.00	
P2	Гайка	40.00	
...			

Поставки

КодП	ИмяД	К-во
S1	P1	3000
S2	P2	5000
...		

Уровень подсхем

- **Подсхема** – представление базы данных с точки зрения конечного пользователя.

Поставщики			Детали	
КодП	ИмяП	Город	КодД	Цена
S1	Иванов	Москва	P1	10.00
S2	Петров	Челябинск	P2	40.00
...			...	

Диспетчер



Поставки			
ИмяП	ИмяД	К-во	Цена
Иванов	Болт	3000	10.00
Петров	Гайка	5000	40.00
...			

Экономист



Цикл работы СУБД

- 1) Пользователь формирует запрос на языке SQL.
- 2) СУБД воспринимает запрос и интерпретирует его.
- 3) Запрос к подсхеме преобразуется в эквивалентный запрос к схеме.
- 4) Запрос к схеме преобразуется в эквивалентный запрос к физическому представлению.
- 5) Отрабатывается запрос к физическому представлению.
- 6) Результат преобразуется к виду уровня схемы.
- 7) Результат преобразуется к виду уровня подсхемы и помещается в рабочую область пользователя.

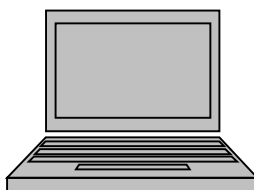
Цикл работы СУБД

1) Пользователь формирует запрос на языке SQL.

Уровень
подсхем



Экономист



```
select ИмяП  
from Поставки  
where Цена > 10;
```

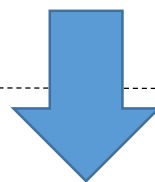
Цикл работы СУБД

- 2) СУБД воспринимает запрос и интерпретирует его.
- 3) Запрос к подсхеме преобразуется в эквивалентный запрос к схеме.

Уровень
подсхем



```
select ИмяП  
from Поставки  
where Цена > 10;
```

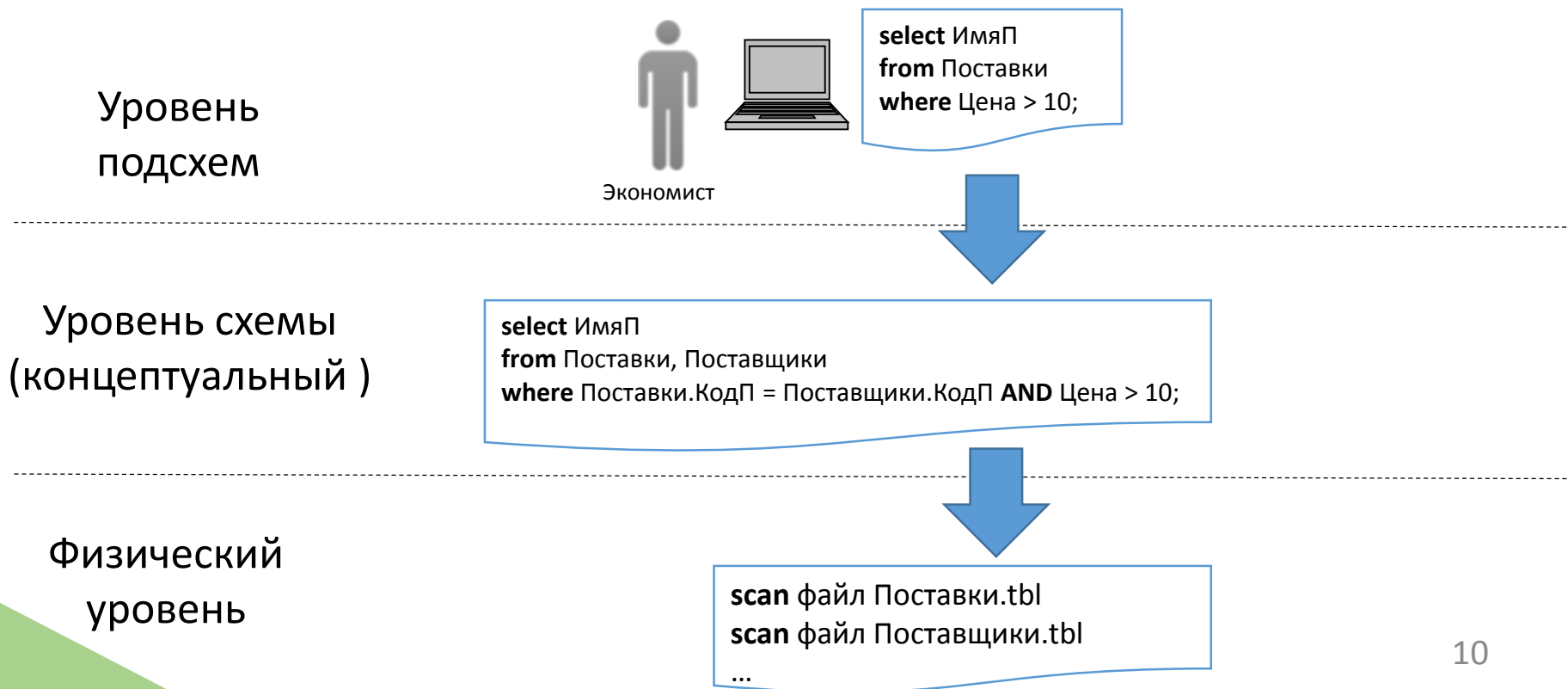


Уровень схемы
(концептуальный)

```
select ИмяП  
from Поставки, Поставщики  
where Поставки.КодП = Поставщики.КодП AND Цена > 10;
```

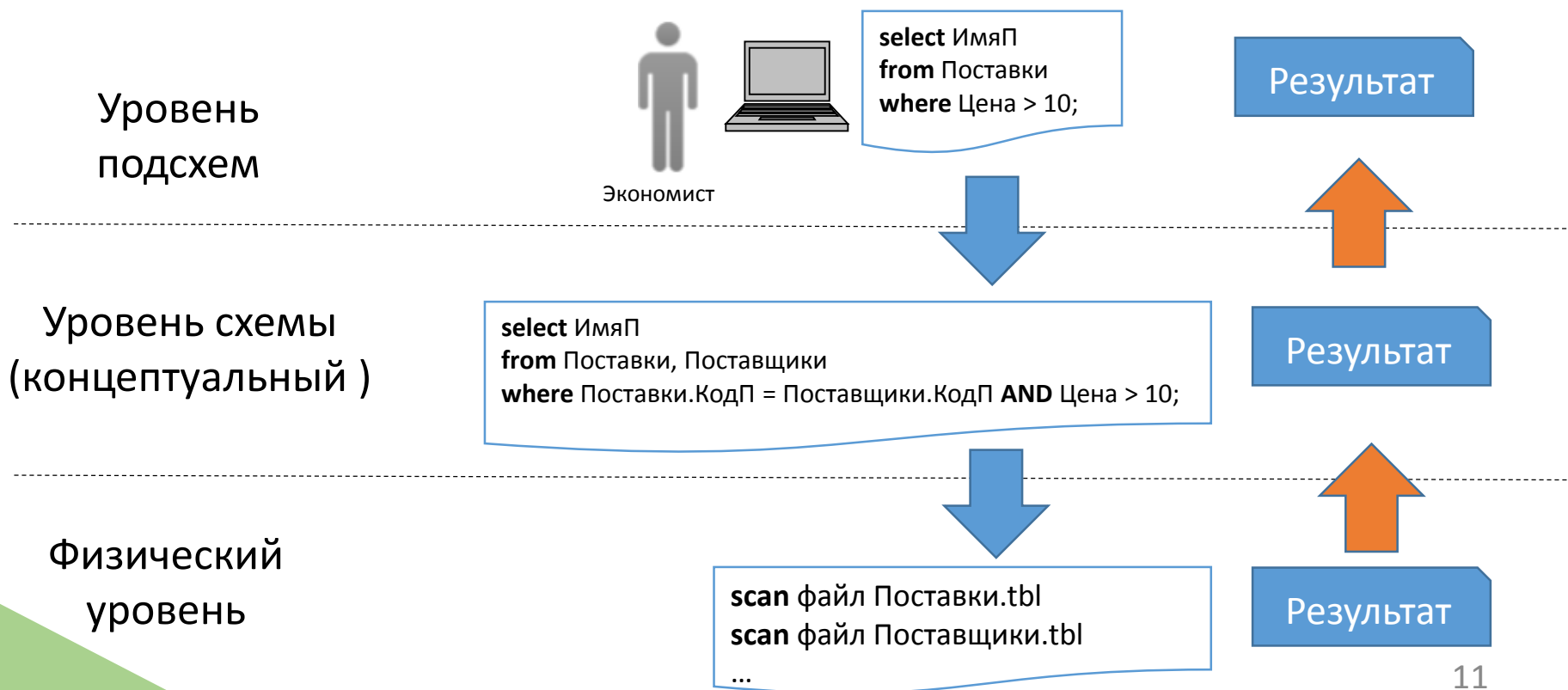
Цикл работы СУБД

- 4) Запрос к схеме преобразуется в эквивалентный запрос к физическому представлению.
- 5) Отрабатывается запрос к физическому представлению.



Цикл работы СУБД

- 6) Результат преобразуется к виду уровня схемы.
- 7) Результат преобразуется к виду уровня подсхемы и помещается в рабочую область пользователя.

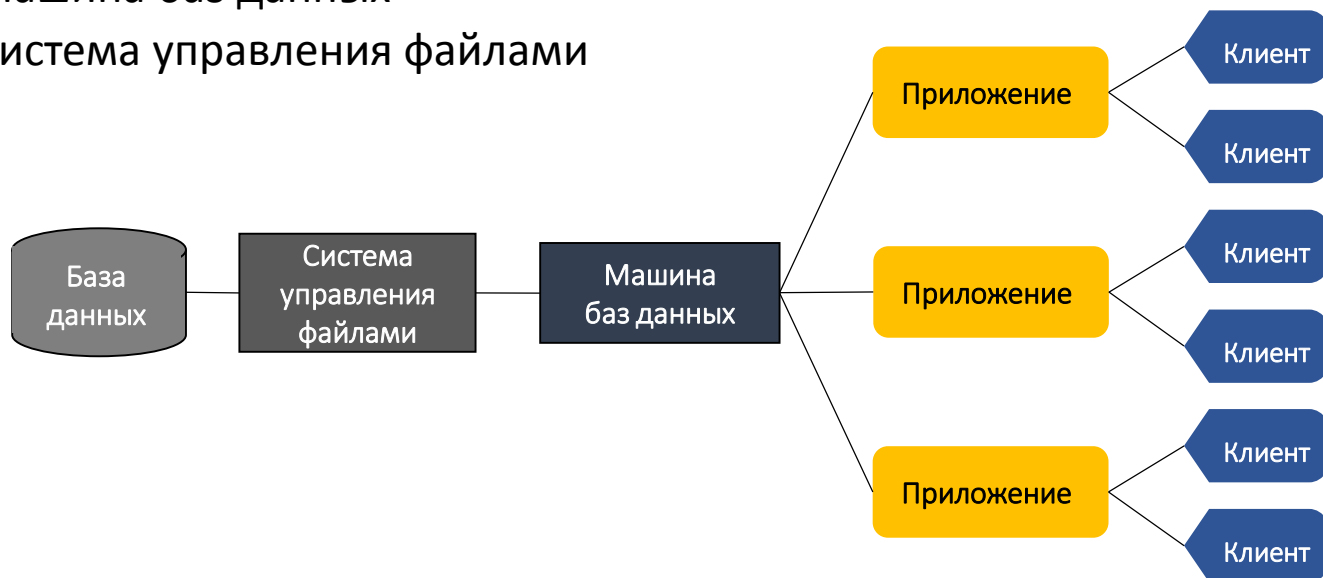


Администратор базы данных

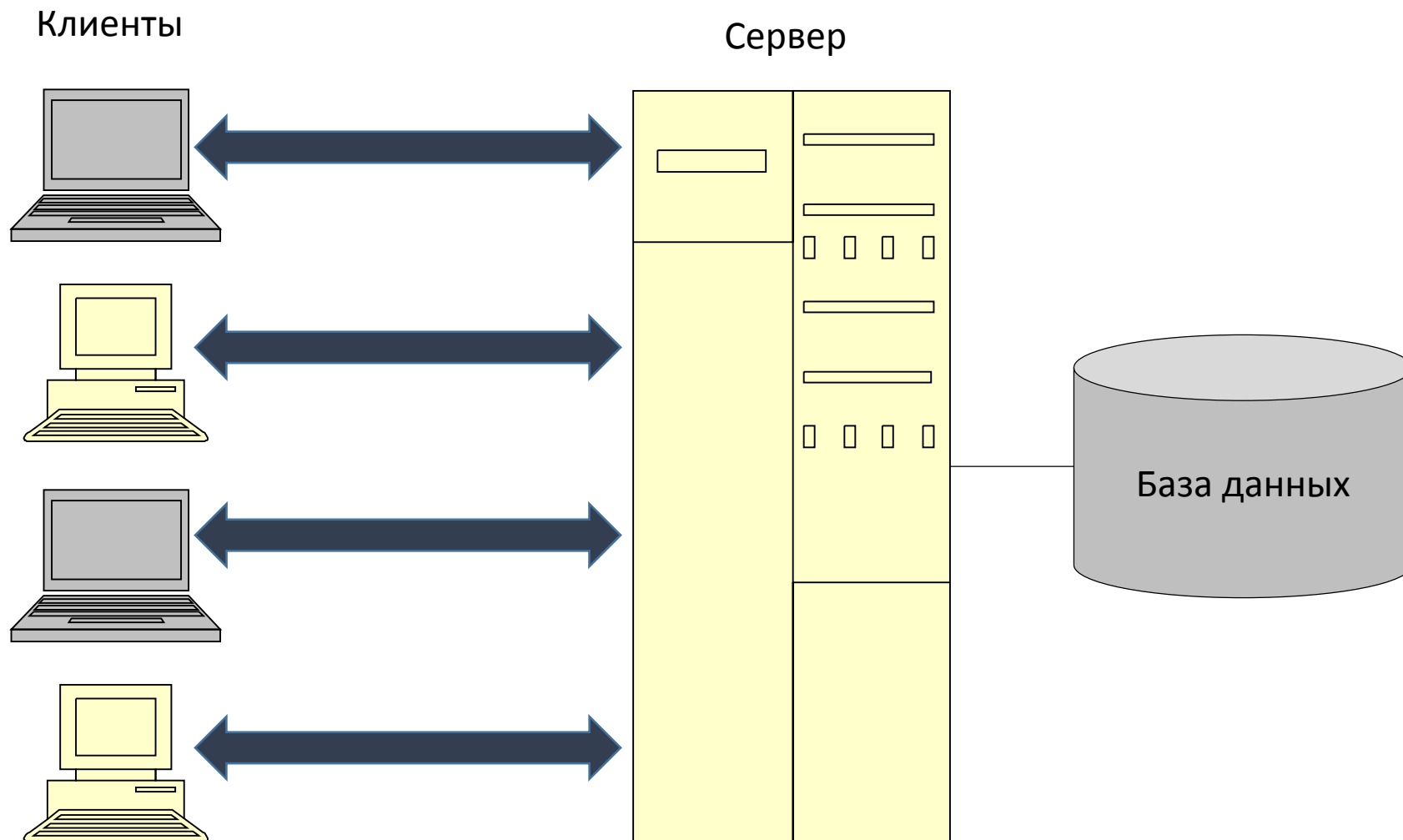
- **Администратор базы данных** – это лицо (группа лиц), ответственное за общее управление базой данных.
- Основные функции администратора базы данных
 - Описание схемы на языке описания данных и задание отображения "элементы схемы → элементы физического представления".
 - Описание подсхем на языке описания данных и задание отображений "элементы подсхемы → элементы схемы" компиляция и сохранение.
 - Регистрация пользователей и регламентация прав доступа к базе данных.
 - Задание ограничений целостности базы данных.
 - Определение стратегии резервного копирования и восстановления базы данных.

Сетевая архитектура системы баз данных

- Сетевая архитектура определяет принцип построения системы баз данных в зависимости от расположения компонент системы в компьютерной сети.
- Компоненты системы баз данных
 - Пользовательский интерфейс
 - Прикладные функции
 - Машина баз данных
 - Система управления файлами



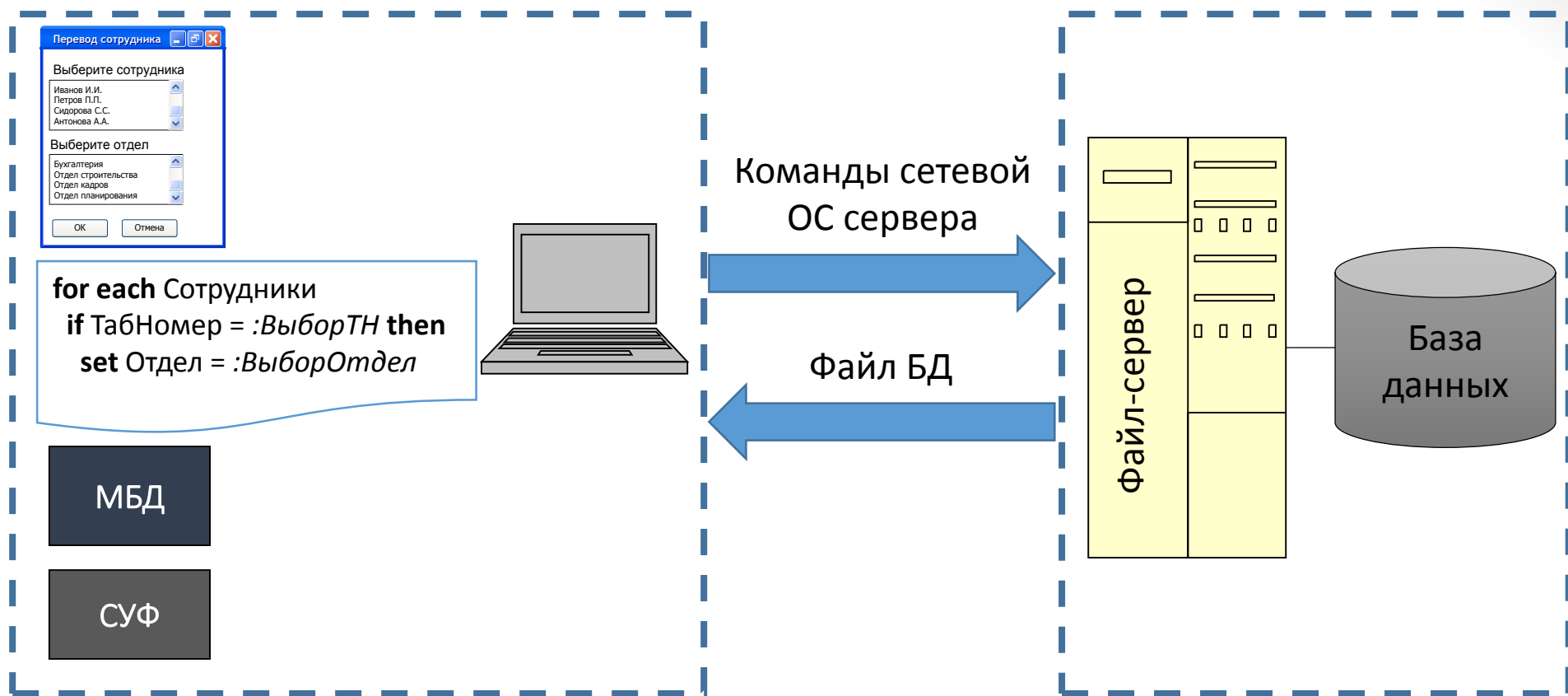
Архитектура клиент-сервер



Разновидности архитектуры клиент-сервер

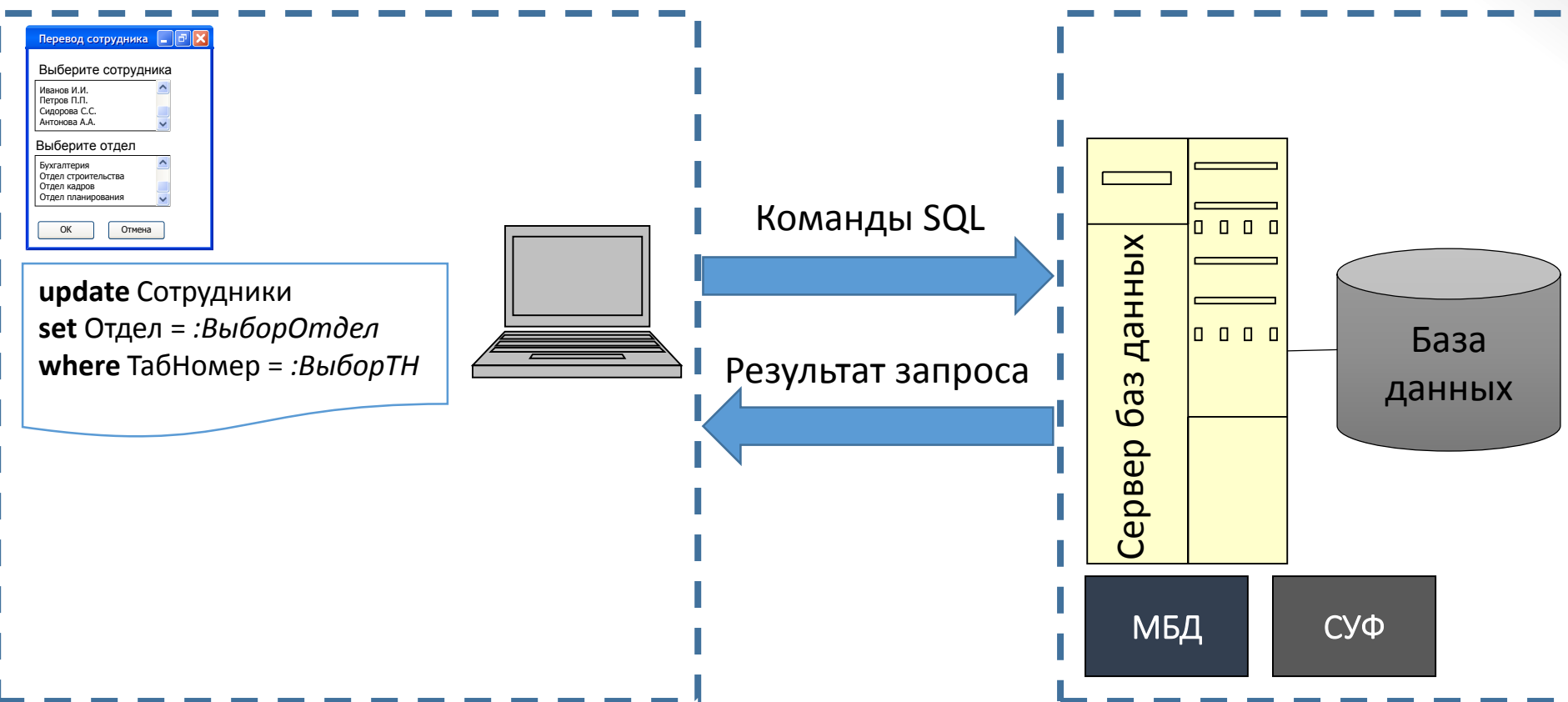
- Файл-сервер
- SQL-сервер, активный SQL-сервер
- Сервер приложений

Архитектура файл-сервер



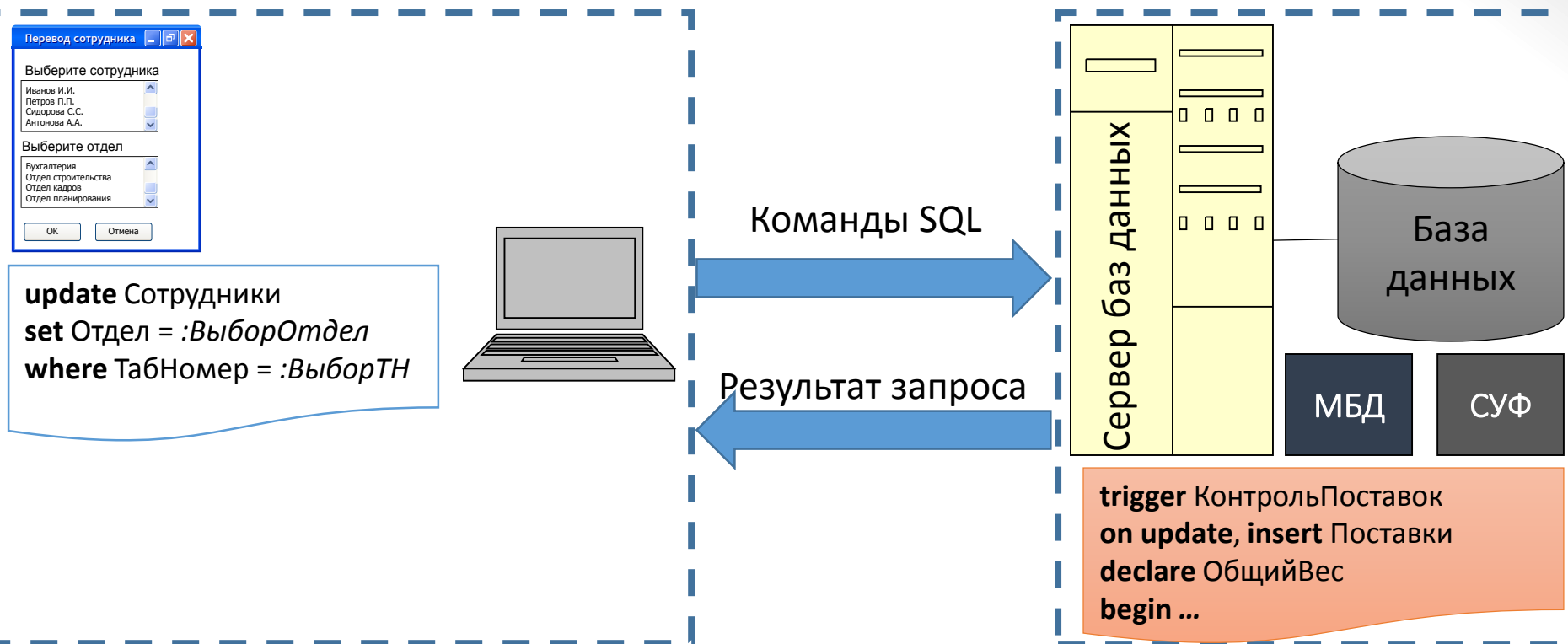
- Выполнение запроса осуществляется на клиенте; большой сетевой трафик.
- Пример: СУБД Clipper.

Архитектура сервер баз данных



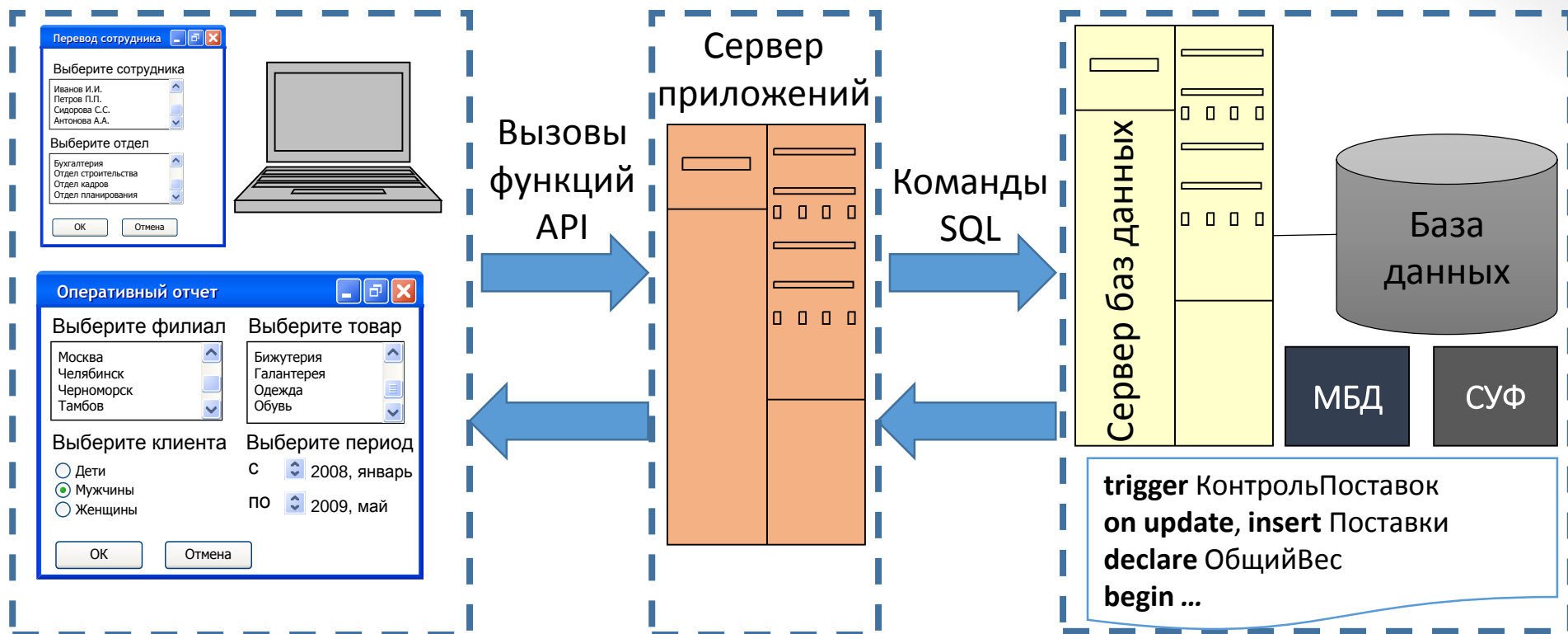
- Выполнение запроса осуществляется на сервере; клиент лишь передает серверу SQL-запрос.
- Пример: все современные корпоративные СУБД (Oracle, MS SQL Server, DB2 и др.).

Архитектура активный сервер баз данных

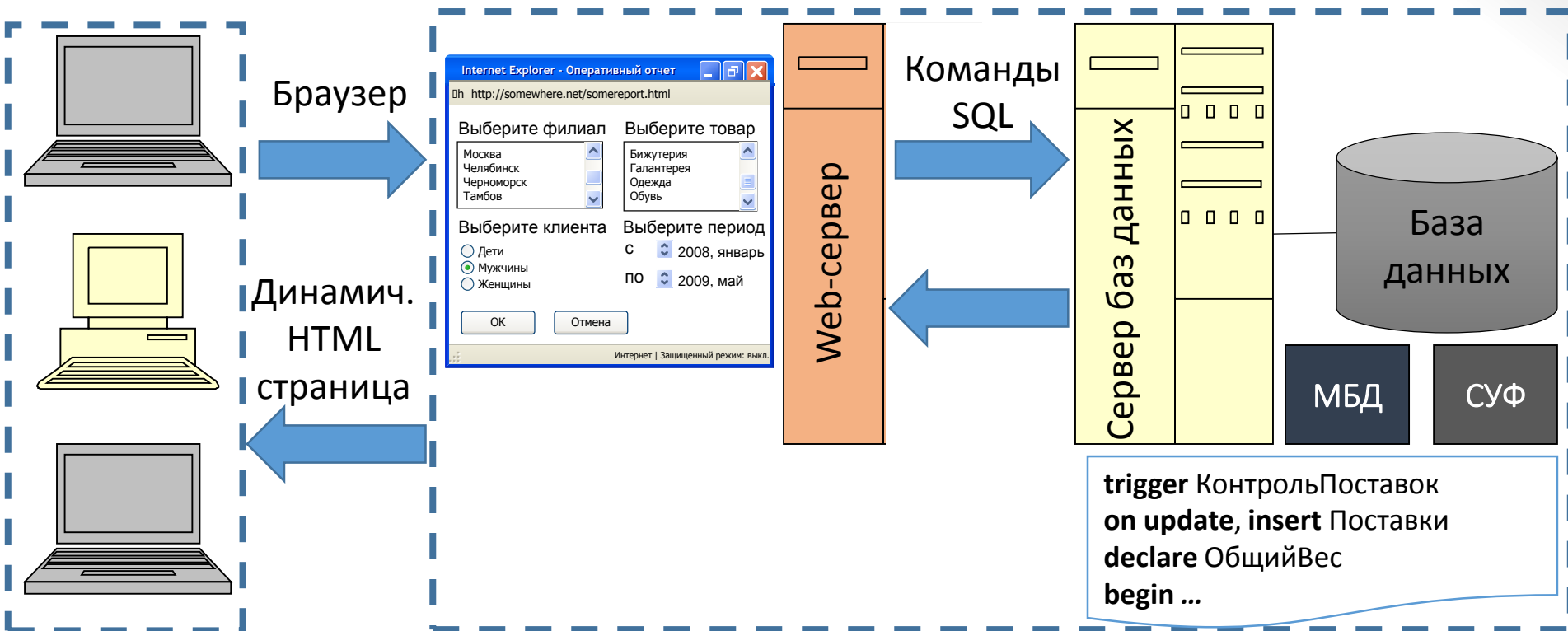


- В базе данных на сервере хранится часть прикладных функций
 - **Триггер** – {событие, действие}. Событие – удаление, добавление, обновление записи таблицы. Действие – подпрограмма, автоматически запускаемая сервером при наступлении события.
 - **Хранимая процедура** может быть исполнена (команда **exec**) клиентом.

Архитектура сервера приложений



Централизованная архитектура

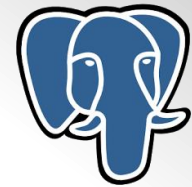




PostgreSQL

Концепция архитектуры PostgreSQL

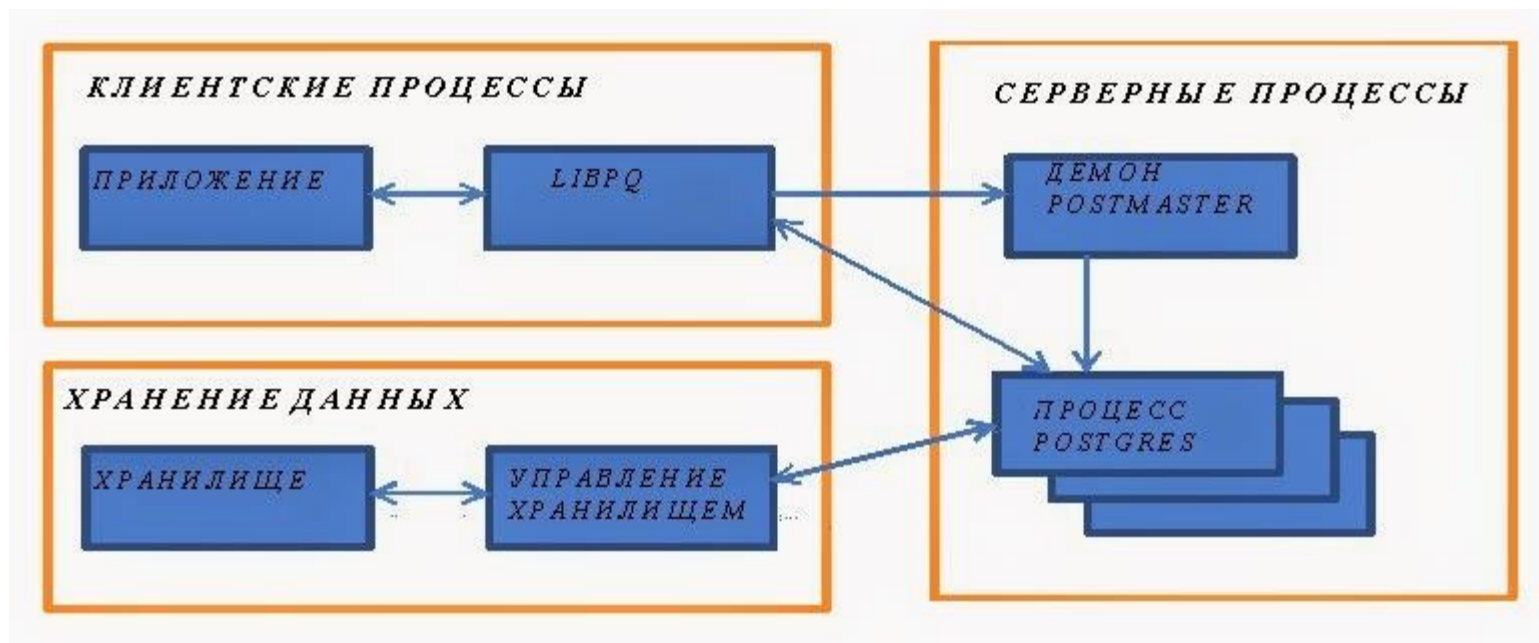
- PostgreSQL — объектно-реляционная СУБД, распространяемая под свободной лицензией.
- PostgreSQL изначально фокусировалась на использовании в сложных приложениях. Большой упор делался на надёжность, развитую функциональность.
- PostgreSQL «вырос» из некоммерческой СУБД Postgres, разработанной, как и многие открытые проекты, в Калифорнийском университете в Беркли. К разработке Postgres имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres. Название «Postgres» расшифровывалось как «Post Ingres», соответственно, при создании Postgres использовались многие уже ранее сделанные наработки.



PostgreSQL

Архитектура системы PostgreSQL

- Архитектура Postgres разбита на 3 основные подсистемы:
 - Front End или клиентская часть системы
 - Серверная часть
 - Back End

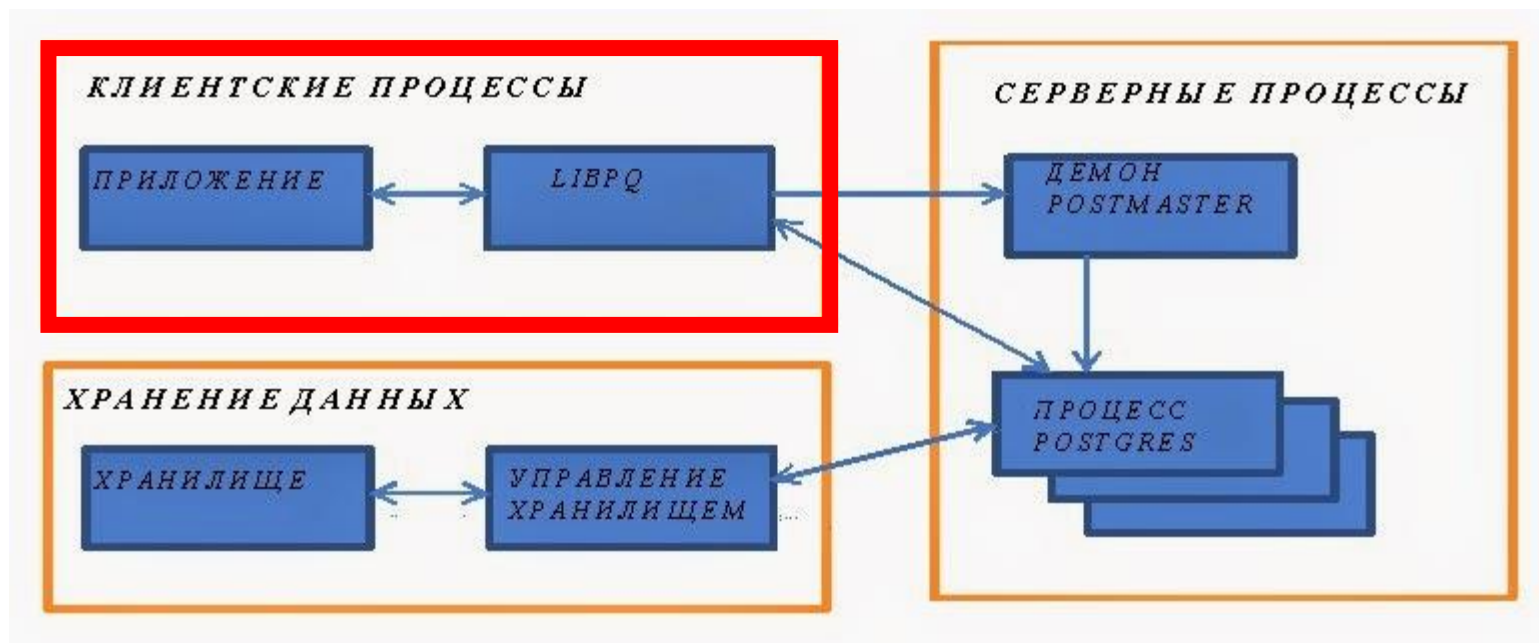




PostgreSQL

Архитектура системы PostgreSQL

- **Front End** включает в себя собственно клиентское приложение и библиотеку LIBPQ, реализующую интерфейс связи с сервером. Библиотека LIBPQ отвечает за установление соединения с сервером и передачу SQL-запросов.

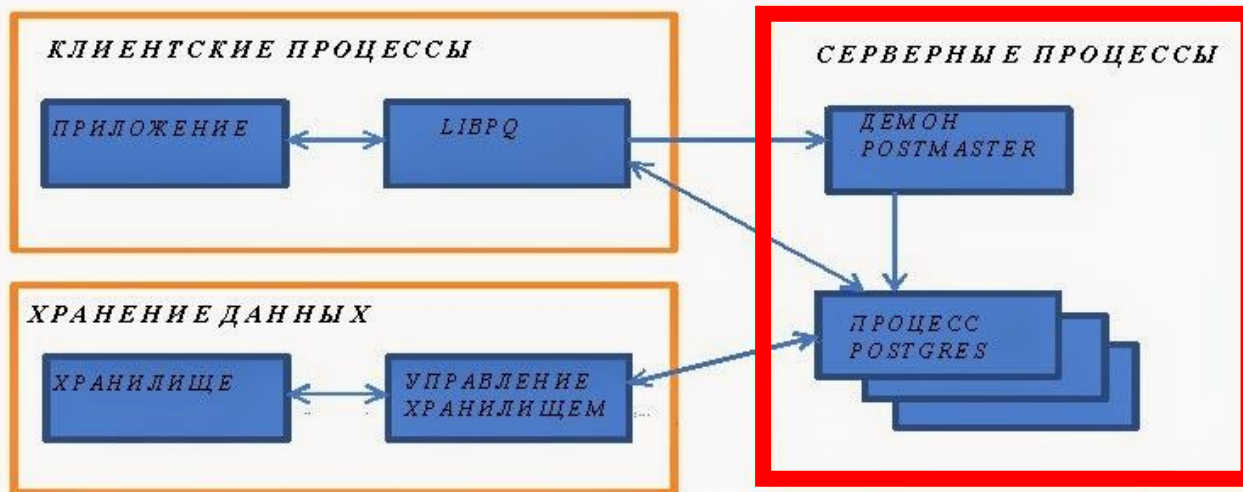


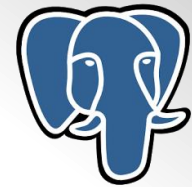


PostgreSQL

Архитектура системы PostgreSQL

- **Серверная часть** включает серверные процессы и контролирующий процесс-демон Postmaster, отвечающий за взаимодействие с клиентами.
- Демон postmaster постоянно запущен в фоновом режиме на сервере. Он авторизует и принимает запросы от клиентов и осуществляет обмен данными между клиентом и сервером. При получении запроса соединения от клиента Postmaster создаёт соответственный фоновый серверный процесс postgres, при этом используется связь один-к-одному. После того, как серверный процесс создан, клиент и сервер взаимодействуют напрямую.

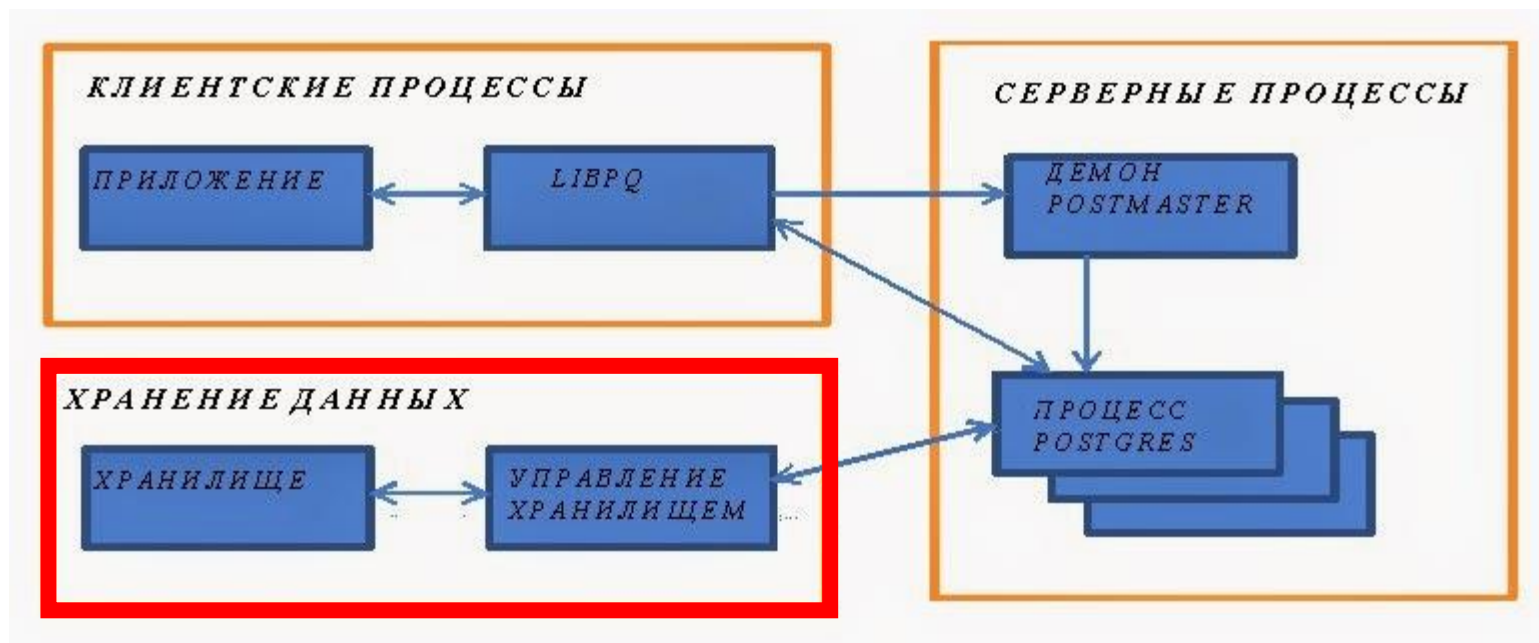




PostgreSQL

Архитектура системы PostgreSQL

- **Back End** включает хранилище данных и средства управления хранилищем. Несколько серверных процессов могут одновременно иметь доступ к информации из хранилища.

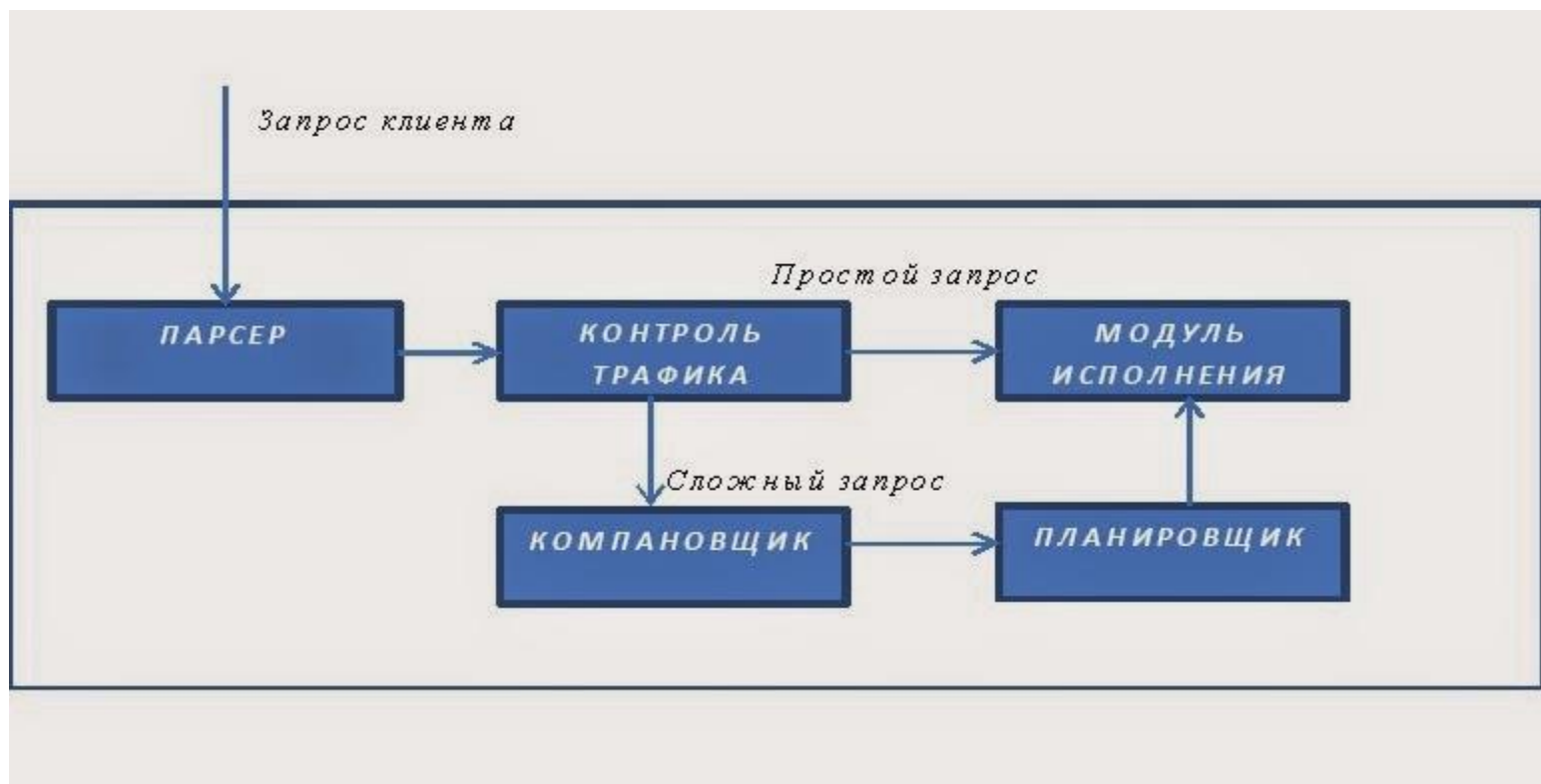


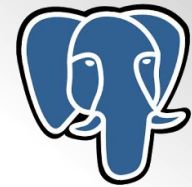


PostgreSQL

Серверная часть PostgreSQL

- После установления соединения, серверный процесс получает SQL-запросы в текстовом виде и трансформирует эти запросы в поток выходных данных.

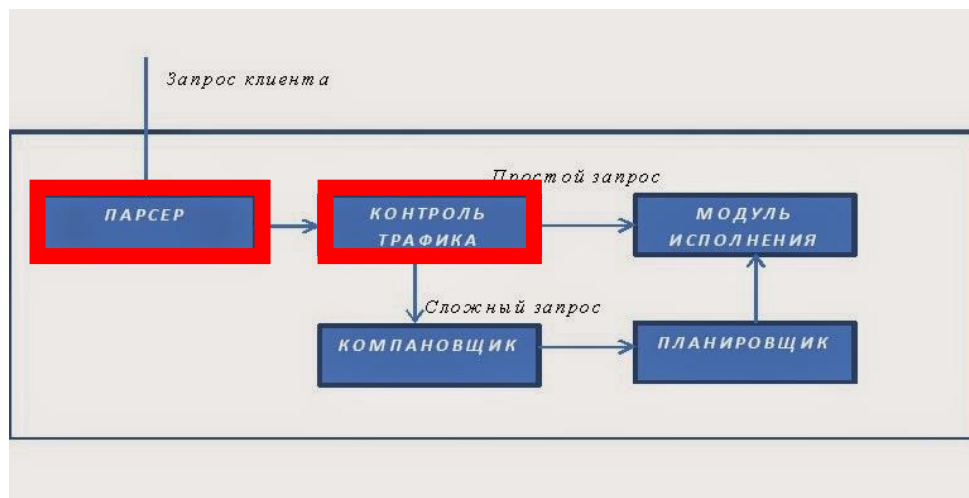


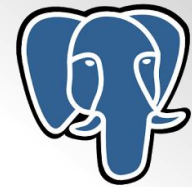


PostgreSQL

Серверная часть PostgreSQL

- 1) Обработка запроса происходит в следующем порядке:
- 2) Парсер принимает запрос в виде текста в формате ASCII и проверяет его синтаксис, распознавая идентификаторы и ключевые слова. В результате работы парсера формируется дерево разбора, либо в случае неверного синтаксиса генерируется ошибка.
- 3) Далее за дело принимается служба контроля трафика. На этом этапе запрос идентифицируется как простой или сложный. Простые запросы перенаправляются непосредственно в модуль исполнения, сложные передаются через компоновщик планировщику/оптимизатору.

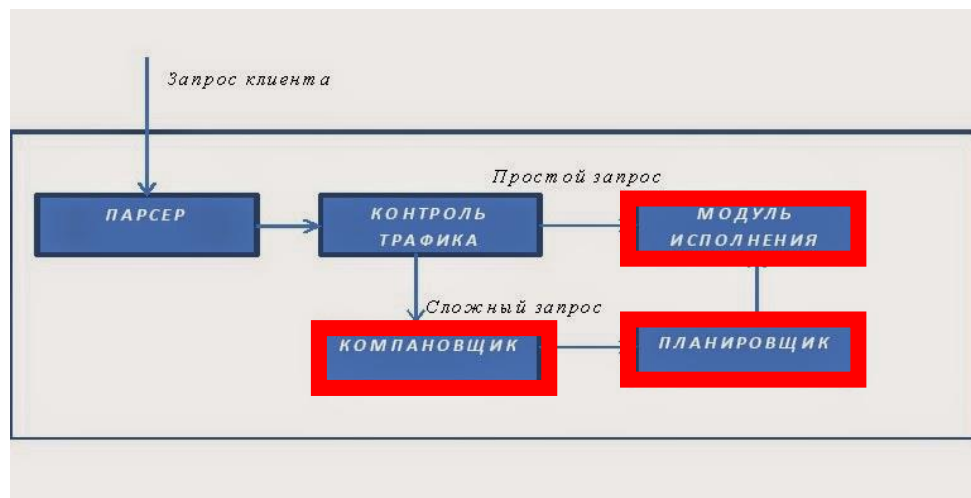


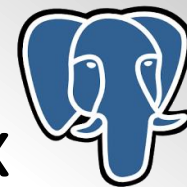


PostgreSQL

Серверная часть PostgreSQL

- 4) **Компановщик** принимает запрос от парсера и преобразует его в альтернативную вспомогательную форму.
- 5) Сложный SQL-запрос может быть выполнен несколькими разными способами. **Планировщик** определяет наиболее оптимальный путь и передаёт управление модулю исполнения.
- 6) **Модуль исполнения** выполняет запрос оптимальным образом согласно дереву разбора, предоставленному планировщиком, и возвращает вывод в текстовом виде клиенту.

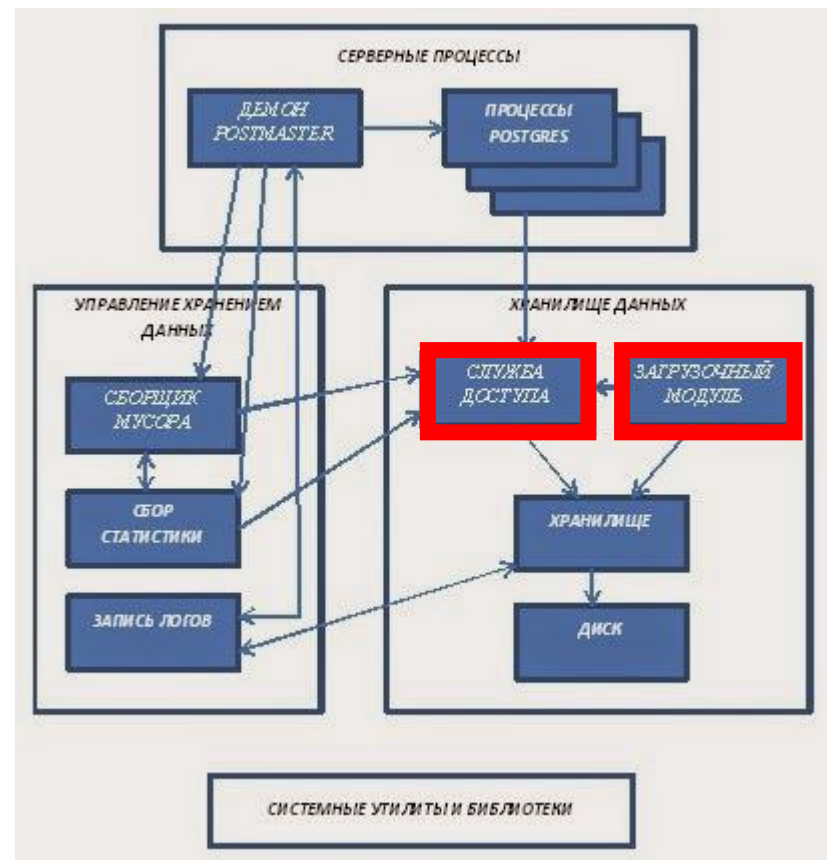


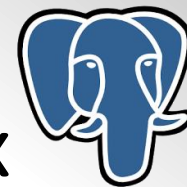


PostgreSQL

Принципы организации хранения данных

- Взаимодействие с хранилищем данных осуществляется через **службы доступа и загрузочный модуль**.

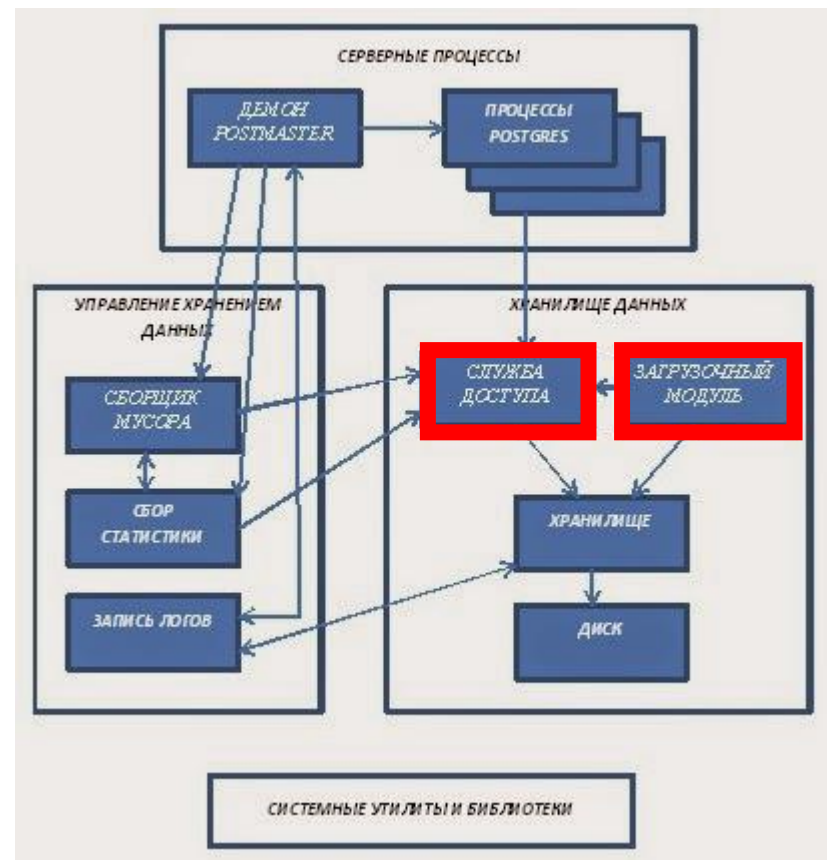


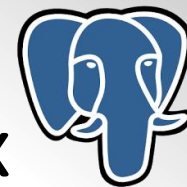


PostgreSQL

Принципы организации хранения данных

- Образец базы данных создаётся с помощью загрузочного модуля при первом запуске Postgres. На этом этапе невозможно обращение к базе данных через обычные SQL-запросы.
- Затем вся работа происходит через службу доступа к базе данных. Служба доступа отвечает за индексирование, сканирование, поиск, компиляцию и возвращение запрошенных данных.

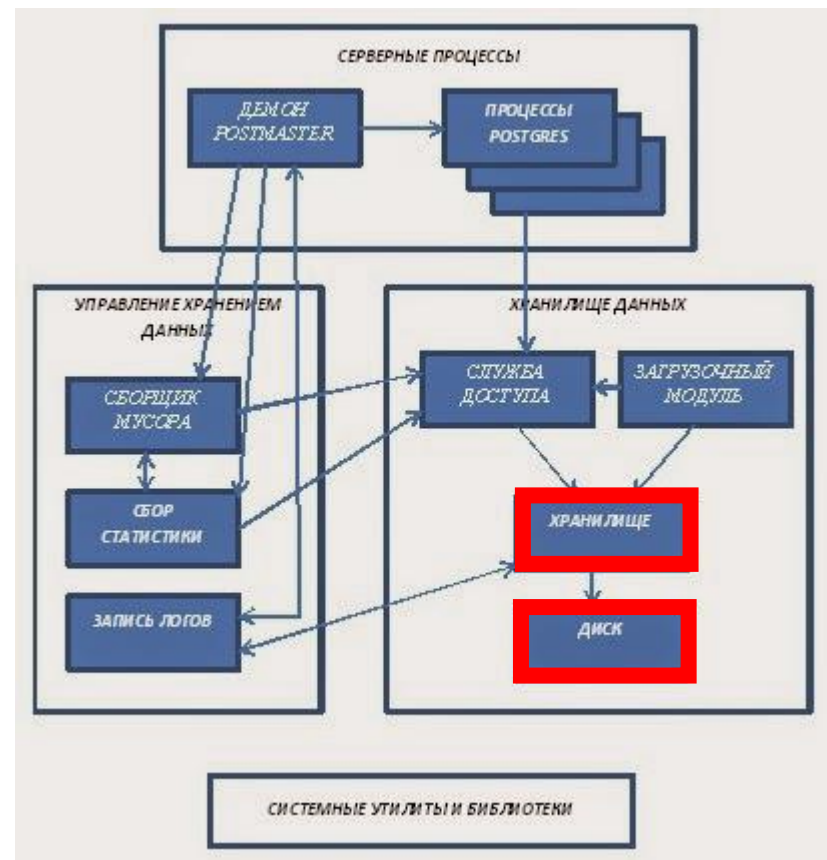


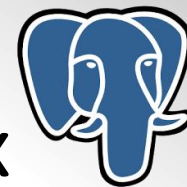


PostgreSQL

Принципы организации хранения данных

- **Хранилище** предоставляет унифицированный доступ к серверным данным на **диске**. С помощью сложной системы буферизации в хранилище возможен множественный доступ к одним и тем же таблицам параллельными запросами и процессами. Таким образом, хранилище является посредником между службами Postgres и физическим диском, а также обеспечивает семафоры и блокировки файлов. На сервере Postgres может существовать только одно хранилище.



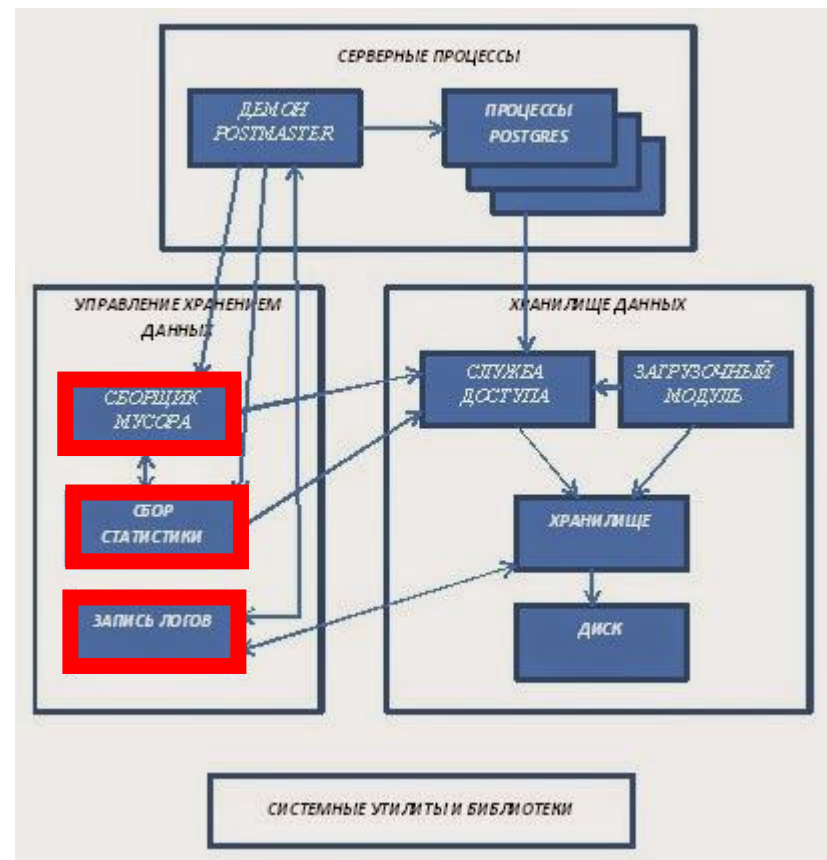


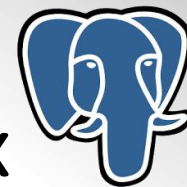
PostgreSQL

Принципы организации хранения данных

Управление хранилищем включает в себя несколько независимых, иногда необязательных, подсистем:

- **Модуль сбора статистики** накапливает информацию о доступе к таблицам и индексам, вызовах серверных функций и командах, выполненных модулем исполнения. По запросу накопленная статистика передаётся другим процессам системы.
- **Сборщик мусора Auto-Vacuum** — набор процессов для автоматического освобождения неиспользуемой памяти в таблицах. Для принятия решения об очистке неиспользуемых данных сборщик мусора опирается на данные, полученные от модуля сбора статистики.
- **Фоновый процесс записи логов** сохраняет логи произведённых операций и информацию для резервного восстановления системы в случае поломки. Все изменения, сделанные после последнего сохранения состояния системы записываются в специальных лог-файлах.





PostgreSQL

Принципы организации хранения данных

- Системные утилиты предоставляет некоторые общие функции для процессов серверной части. Системные утилиты доступны для всех подсистем сервера в фоновом режиме.

