

# **Базы данных**

## **Лекция 03 – Архитектура СУБД**

**Преподаватель: Поденок Леонид Петрович, 505а-5**

**+375 17 293 8039 (505а-5)**

**+375 17 320 7402 (ОИПИ НАНБ)**

**prep@lsi.bas-net.by**

**ftp://student:2ok\*uK2@Rwox@lsi.bas-net.by**

**Кафедра ЭВМ, 2024**

2024.02.20

## Оглавление

Трехуровневая архитектура ANSI-SPARC.....	3
Компоненты СУБД.....	5
Администратор базы данных.....	12
Архитектура многопользовательских СУБД.....	18
Низкоуровневые операции логического уровня в таблицах.....	24

# Трехуровневая архитектура ANSI-SPARC

1) **Внешний**, реализующий представления пользователя и описывающие различные внешние представления данных. **Для данной базы данных может быть много внешних схем.**

2) **Концептуальный**, реализующий способ описания того, какие данные хранятся во всей базе данных и как эти данные взаимосвязаны.

Концептуальная схема описывает все элементы данных и отношения между ними вместе с ограничениями целостности. **Для каждой базы данных существует только одна концептуальная схема.**

3) **Внутренний**, определяющий как база данных физически представлена в компьютерной системе. Внутренняя схема на самом нижнем уровне содержит определения хранимых записей, методов представления, полей данных и индексов. **Для каждой базы данных существует только одна внутренняя схема.**

	Пользователь 1	Пользователь 2	Пользователь N	
	Представление 1	Представление 2	...	Представление 3
Внешний уровень				
Концептуальный уровень	Концептуальная схема (независимость от физической структуры данных)			
Внутренний уровень	Внутренняя схема			
Физическая организация данных	Физический уровень базы данных			

Независимость от логической структуры данных

Преобразование внешняя-концептуальная

Преобразование концептуальная - внутренняя

## Концептуальная схема (Концептуальная модель)

**Концептуальная схема представляет собой единое интегрированное определение данных в рамках предприятия, которое не привязано к какому-либо отдельному применению данных и не зависит от того, как данные физически хранятся или к ним осуществляется доступ.**

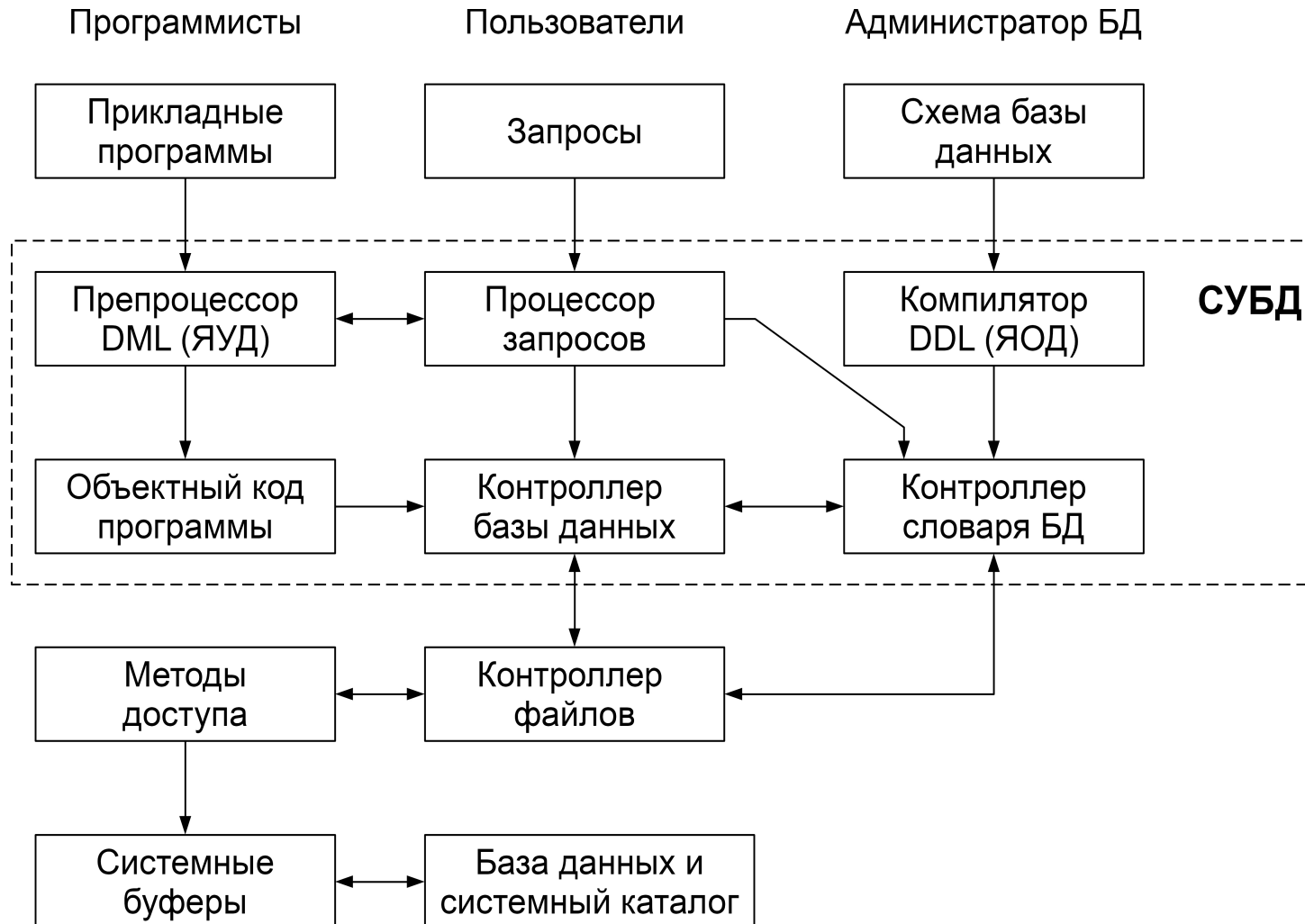
Целью этой концептуальной схемы является предоставление последовательного определения значений и взаимосвязей данных, которые можно использовать для интеграции, совместного использования и управления целостностью данных.

По мере развития практики администрирования данных и появления новых графических методов термин «схема» уступил место термину «модель».

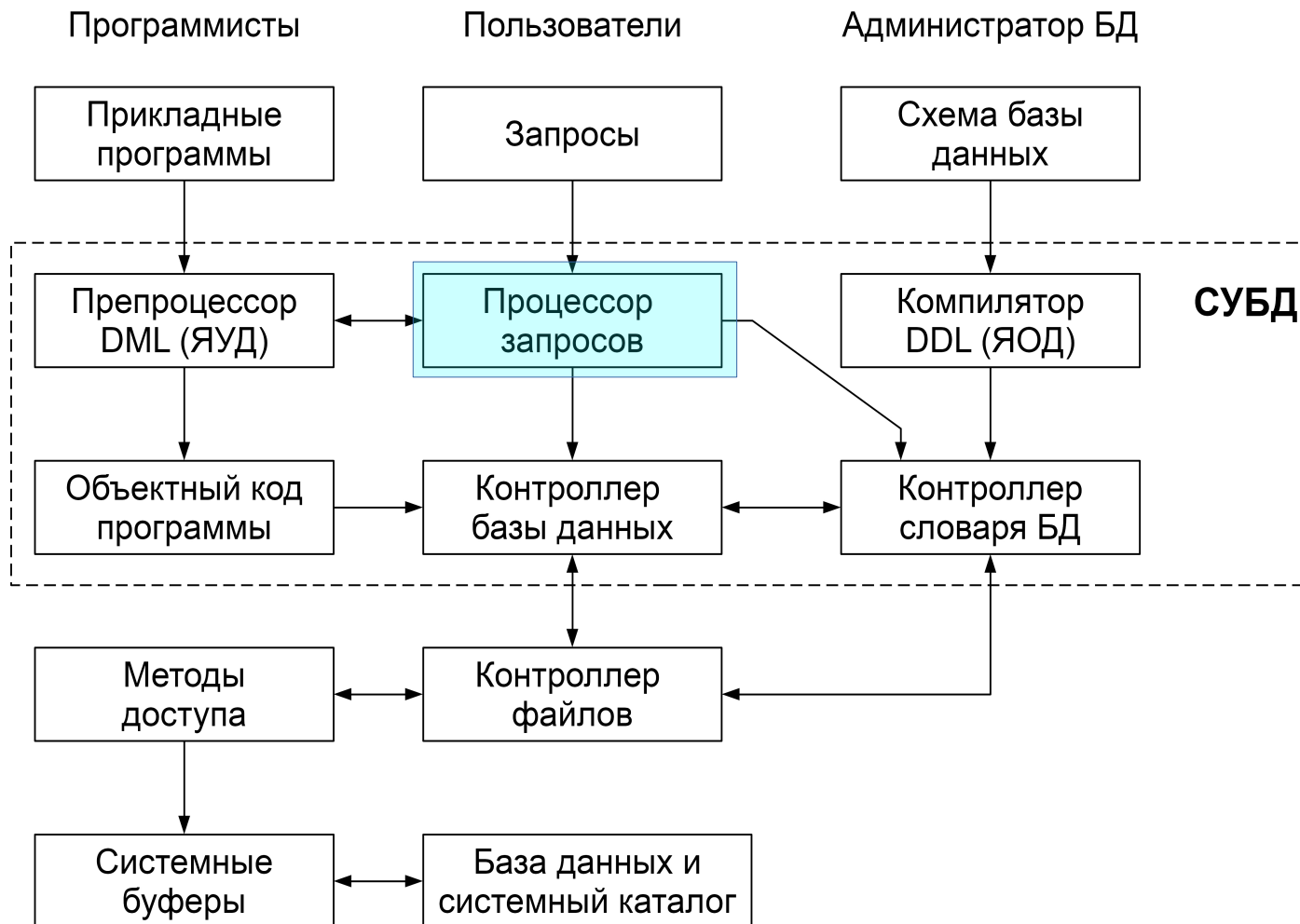
Концептуальная модель представляет собой представление данных, которое согласовывается между конечными пользователями и администраторами баз данных, охватывающее те объекты, о которых важно хранить данные, значение данных и взаимосвязь данных друг с другом.

# Компоненты СУБД

СУБД состоит из программных компонентов (модулей), каждый модуль выполняет одну специфическую операцию (часть операций может поддерживаться операционной системой, но только базовые и контролируемые СУБД (как надстройкой над ОС)).



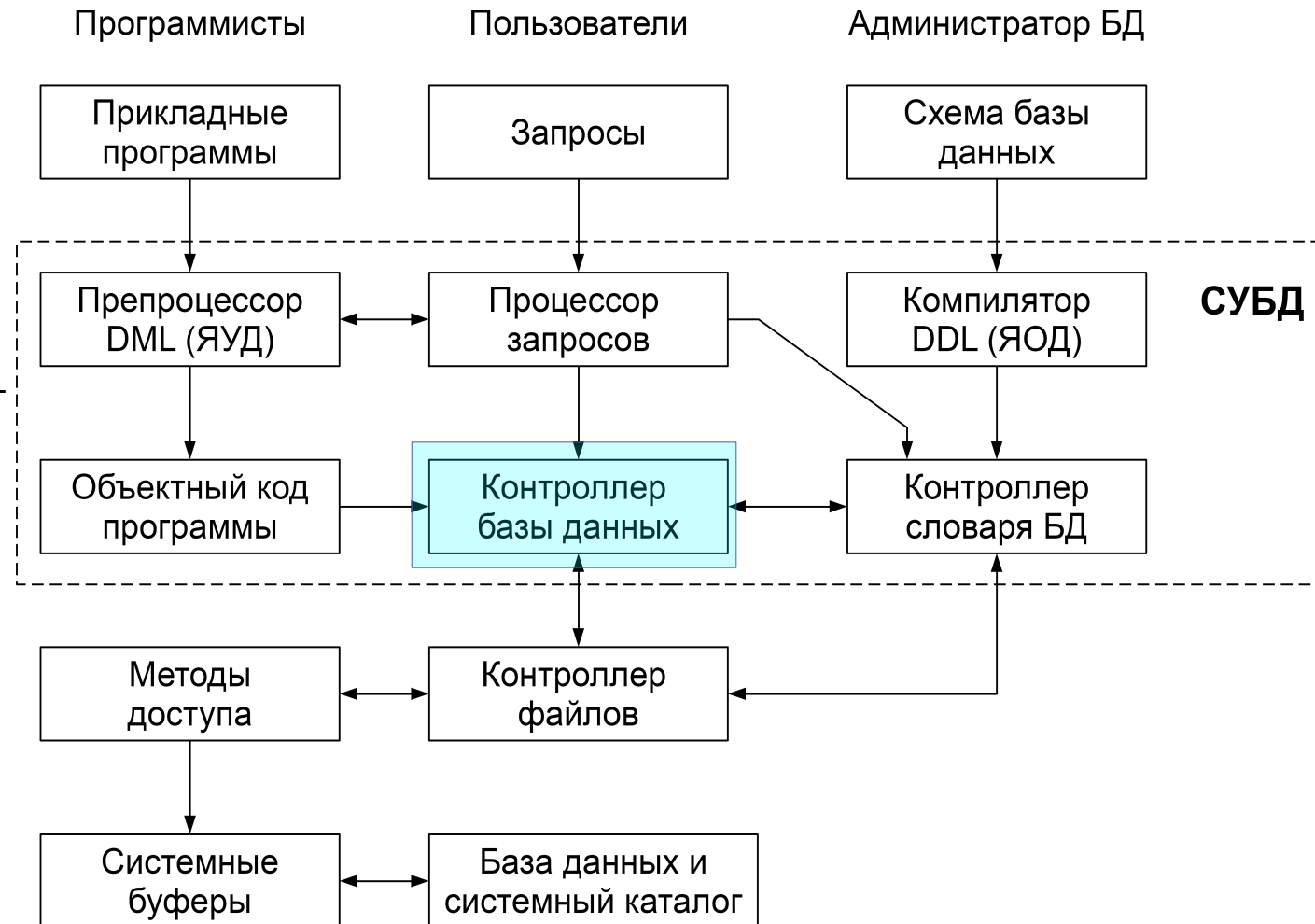
**Процессор запросов** – основной компонент СУБД – преобразует запросы в последовательность низкоуровневых инструкций для контроллера базы данных;



**Контроллер базы данных** – принимает запросы, проверяет внешние и концептуальные схемы для определения тех концептуальных записей, которые необходимы для выполнения запроса.

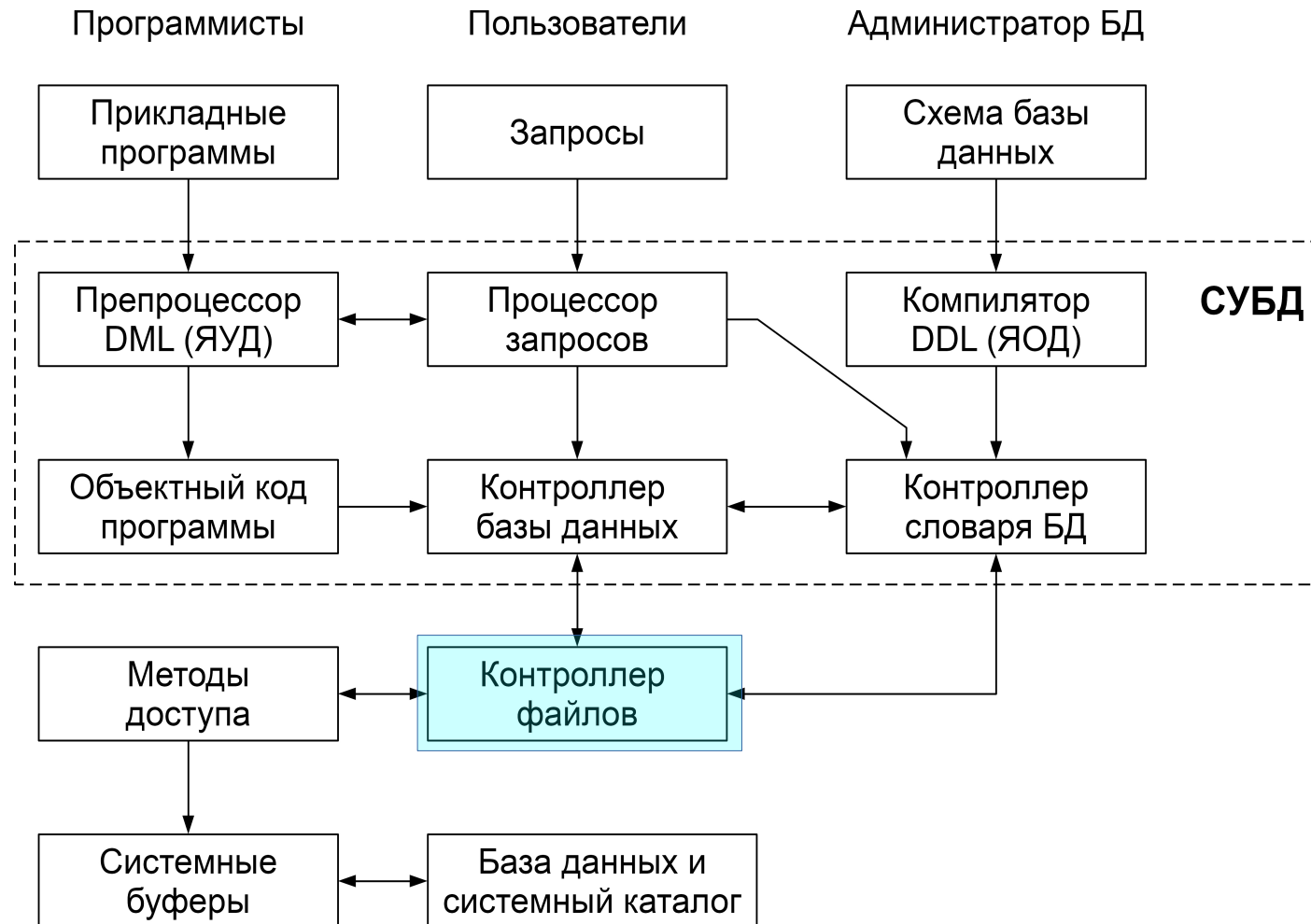
В процессе обработки запроса обычно используются следующие основные компоненты:

- 1) контроль прав доступа;
- 2) процессор команд;
- 3) средства контроля целостности;
- 4) оптимизатор запросов;
- 5) контроллер транзакций;
- 6) планировщик;
- 7) контроллер восстановления;
- 8) контроллер буферов (память-диск);



## Контроллер файлов – манипулирует файлами БД.

Создает и поддерживает список структур и индексов, определенных во внутренней схеме, но не управляет вводом-выводом данных непосредственно, а лишь передает запросы соответствующим методам доступа, которые обмениваются данными между системными буферами и диском;





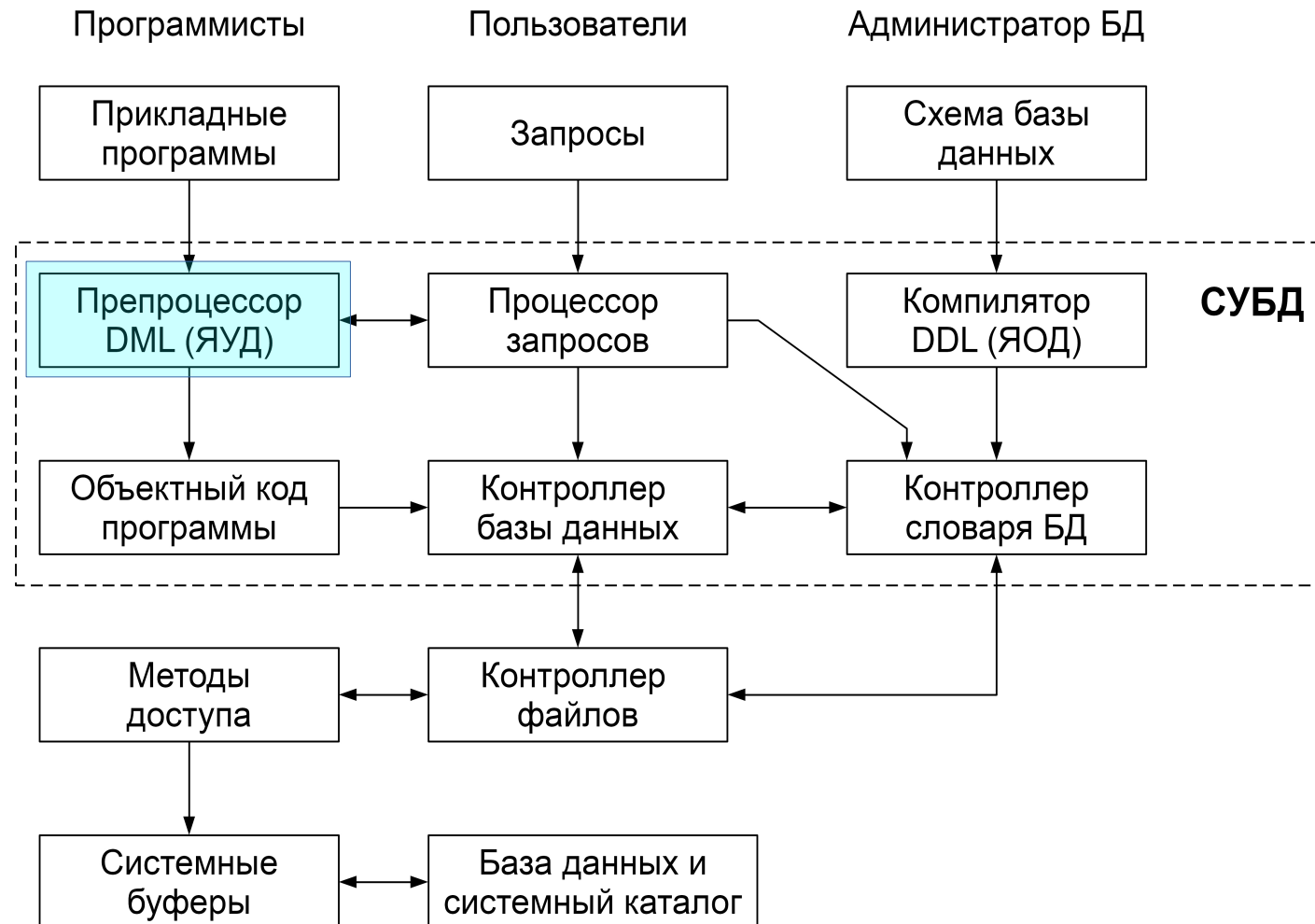
**Преппроцессор языка DML** – взаимодействует с процессором запросов и преобразует внедренные в прикладные программы DML-операторы в вызовы стандартных функций базового языка.

DML (Data Manipulation Language) – язык, содержащий набор операторов для поддержки основных операций манипулирования содержащимися в базе данными:

- вставка;
- модификация;
- извлечение данных
- удаление данных.

DML может быть двух типов:  
**процедурный** – указывает, как можно получить результат оператора DML. Обычно оперирует данными по одной записи;

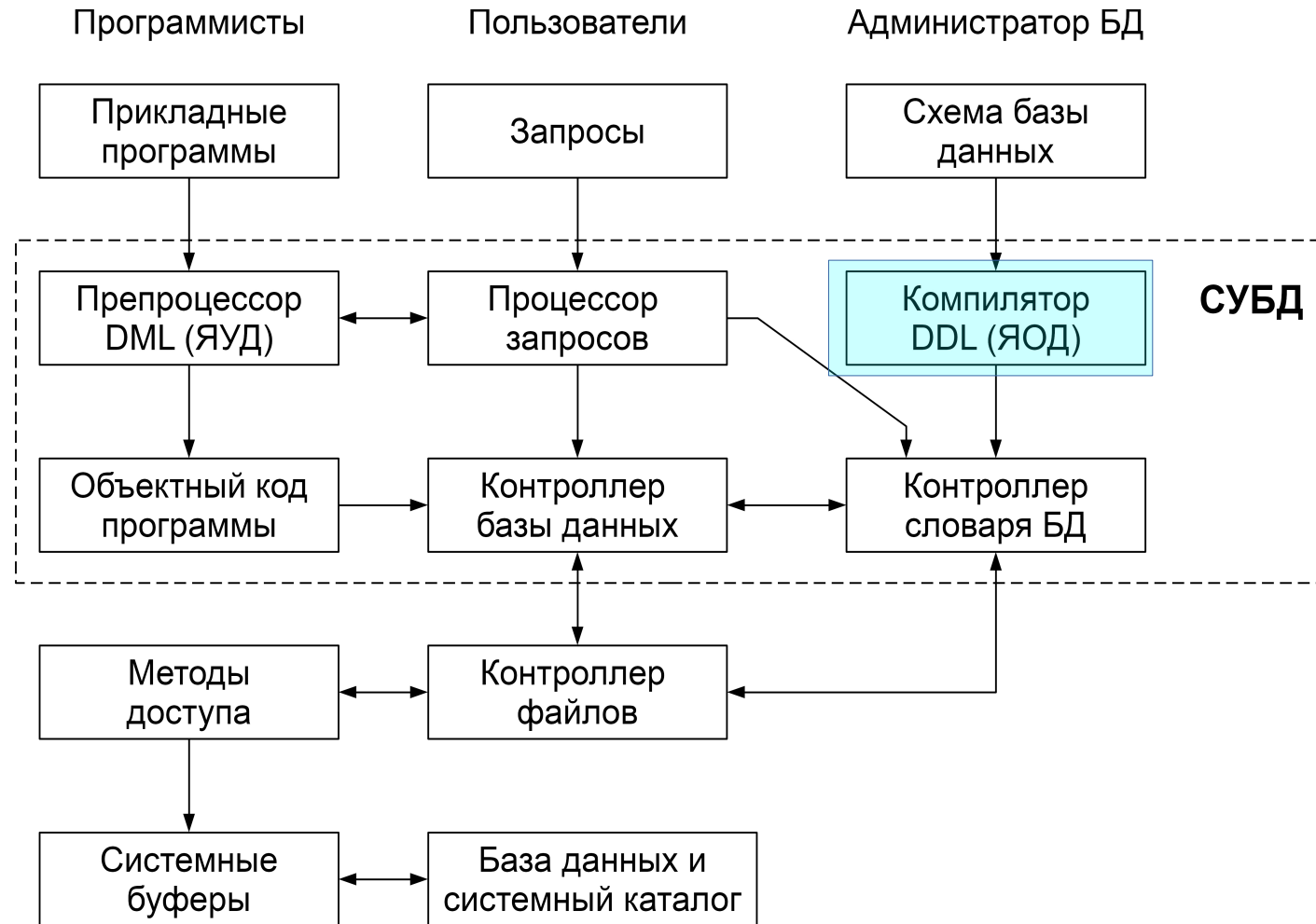
**непроцедурный** – описывает, какой результат будет получен. Обычно оперирует наборами записей (SQL).



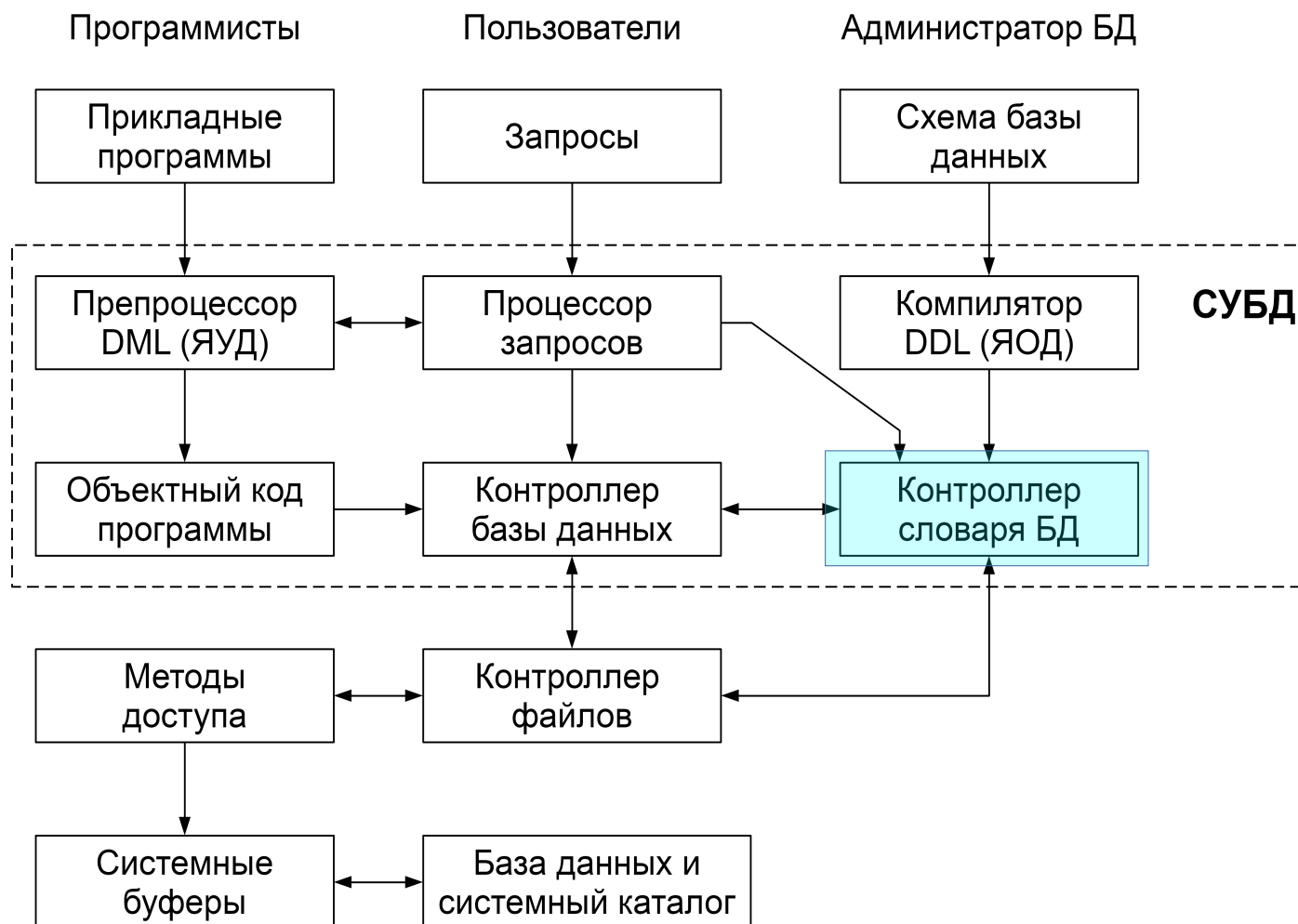
**Компилятор языка DDL** – преобразует DDL-команды в набор таблиц, содержащих метаданные, которые сохраняются в системном каталоге и частично в заголовках файлов с данными.

DDL (Data Definition Language) - описательный (декларативный) язык, который позволяет описывать сущности (объекты), необходимые для работы некоторого приложения, а также связи, имеющиеся между различными сущностями.

Теоретически, в архитектуре ANSI-SPARC, можно выделить языки DDL для каждой схемы, но на практике есть один, позволяющий задавать спецификации как минимум для внешней и концептуальной схем.



**Контроллер словаря** – управляет доступом к системному каталогу и обеспечивает работу с ним. Почти все компоненты СУБД имеют доступ к системному каталогу.



# Администратор базы данных

**Администратор данных** — человек, отвечающий за стратегию и политику принятия решений, связанных с данными предприятия.

Администратор данных решает, **какая** именно информация должна храниться в базе данных, т.е. указывает те типы сущностей, в которых заинтересовано предприятие, а также определяет, какую информацию об этих сущностях необходимо вводить в базу данных.

Этот процесс обычно называют логическим (или концептуальным) проектированием базы данных.

**Администратор базы данных (АБД)** — это человек, обеспечивающий необходимую техническую поддержку для реализации принятых решений.

АБД отвечает за общее управление системой на техническом уровне.

Функции АБД включают:

- определение концептуальной схемы;
- определение внутренней схемы.

## Определение концептуальной схемы

После того как содержимое базы данных на абстрактном уровне будет определено администратором данных, АБД создает соответствующую концептуальную схему, используя концептуальный язык определения данных (DDL).

*Объектная (откомпилированная) форма* этой схемы будет использоваться в СУБД для получения ответов на запросы, связанные с доступом к данным.

*Исходная (не откомпилированная) форма* будет играть роль справочного документа для пользователей системы.

Иногда концептуальную схему создает АД, а логическим проектированием занимается АБД.



## Определение внутренней схемы

Администратор базы данных должен также решить, **как** данные будут представлены в хранимой базе данных. Этот процесс обычно называют физическим проектированием базы данных.

После завершения физического проектирования АБД создает соответствующие структуры хранения, используя внутренний язык определения данных, т.е. описывает *внутреннюю схему*.

Также он определяет соответствующее отображение между внутренней и концептуальной схемами.

Как и концептуальная схема, внутренняя схема и соответствующее отображение будут существовать в исходной и объектной формах.

Для определения всех трех схем используются соответствующие языки определения данных. Часто языки определения данных на концептуальном и/или внутреннем уровне могут включать в себя средства определения отображений, тем не менее, две функции – создание схемы и определение отображений – должны быть четко разделены.

Итак, проектирование осуществляется в определенной последовательности — вначале необходимо установить, какие данные требуются, а затем решить, как следует представить их в памяти.

Физическое проектирование выполняется после логического.

## **Взаимодействие с пользователями**

В обязанности АБД входит также взаимодействие с пользователями для предоставления им необходимых данных и подготовки требуемых внешних схем или оказания помощи пользователям в их подготовке с применением прикладного внешнего языка определения данных.

Также функции АБД включают:

- консультации по разработке приложений;
- обеспечение требуемого технического обучения;
- предоставление помощи в выявлении и устранении возникающих проблем;
- прочие виды профессионального обслуживания.

## **Определение требований защиты и обеспечения целостности данных**

Требования защиты и поддержки целостности данных рассматриваются как часть определения концептуальной схемы.

Концептуальный язык определения данных обычно включает в себя средства описания этих требований.

## Определение процедур резервного копирования и восстановления

Предприятие, хранящее свои данные в СУБД полностью зависимо от бесперебойного функционирования этой СУБД.

В случае повреждения какой-либо части базы данных вследствие ошибки человека, отказа оборудования или сбоя программ операционной системы очень необходимо иметь возможность восстановить утраченные данные с минимальной задержкой и с наименьшим воздействием на остальную часть системы.

В идеале, данные, которые не были повреждены, совсем не должны затрагиваться в процессе восстановления.

АБД должен разработать и реализовать, во-первых, подходящую *схему восстановления*, включающую периодическую выгрузку базы данных на устройство резервного копирования (получение дампа), а во-вторых, процедуры загрузки базы данных из последнего созданного дампа в случае необходимости.

Требование *быстрого восстановления* поврежденных данных является одной из тех причин, по которым желательно организовать хранение данных не в каком-либо одном месте, а распределить их по нескольким отдельным базам данных.

Каждая из таких баз данных будет представлять собой оптимальный объект выгрузки или перезагрузки.

Существуют системы, хранящие десятки триллионов байт данных и имеет место тенденция к их росту. Такие системы очень больших баз данных (Very Large DataBase – VLDB) требуют тщательного и продуманного администрирования, особенно если необходимо обеспечить для пользователей постоянный доступ к базе данных.



## **Управление производительностью и реагирование на изменяющиеся требования**

АБД отвечает за такую организацию системы, при которой можно поддерживать производительность, оптимальную для всего предприятия в целом, а также за корректировку работы системы (настройку) в соответствии с изменяющимися требованиями.

Например, может возникнуть необходимость в периодической реорганизации хранимой базы данных для обеспечения того, чтобы производительность системы всегда поддерживалась на приемлемом уровне.

Изменения на уровне физического хранения данных (внутреннем уровне) должны сопровождаться соответствующими изменениями в определении его отображения на концептуальный уровень, чтобы по возможности сохранять концептуальную схему неизменной.

# Архитектура многопользовательских СУБД

Различают следующие архитектуры (по мере развития):

## Телеобработка

В данном случае используется один компьютер, соединенный с терминалами пользователей.

*Терминал* – оконечное устройство, предназначенное только для ввода и отображения информации.

Пользователь через терминал обращается к пользовательскому приложению, а то в свою очередь – к СУБД, затем результат идет в обратном порядке к пользователю. Основной пользовательский интерфейс на мейнфреймах.

## Файловый сервер

Обработка данных распределена в сети. Это обычно локальная вычислительная сеть (ЛВС).

Пользовательские приложения и сама СУБД размещены и функционируют на рабочих станциях и обращаются к файловому серверу только при необходимости доступа к данным, которые располагаются в файлах.

Недостатки:

- большой объем сетевого трафика;
- необходимость в наличии СУБД на каждой рабочей станции;
- управление параллельностью, восстановлением и целостностью данных усложняется из-за доступа к данным от нескольких экземпляров СУБД.

## **Технология «клиент/сервер»**

Единая двухуровневая система, состоящая из:

- Процесс-клиент (толстый или интеллектуальный клиент). Выполняет запросы на доступ к некоторым ресурсам. Внешний компонент, или внешний интерфейс.
- Процесс-сервер, предоставляющий некоторые ресурсы. Внутренний компонент, или машина базы данных.

Процессы совсем необязательно должны быть размещены на одном компьютере.

Обычно сервер располагается на одном узле ЛВС, а клиенты – на других узлах.

**Сервер**— это сама СУБД.

Он поддерживает все основные функции СУБД:

- определение данных;
- манипулирование данными;
- защиту данных
- поддержание целостности данных и т.д.

В частности, он предоставляет полную поддержку внешнего, концептуального и внутреннего уровней. Поэтому сервер в этом контексте — это просто другое название для СУБД.

Сервер принимает и обрабатывает запросы к базе данных, включая проверку полномочий клиента, обеспечение требований целостности, поддержку системного каталога, поддержку параллельного доступа, выполнение запроса и обновление данных, а затем передает полученные результаты обработки обратно клиенту.

**Клиенты** — это различные приложения, которые выполняются с помощью СУБД.

Таковыми являются как приложения, написанные пользователями, так и встроенные приложения, предоставляемые поставщиками СУБД или некоторыми сторонними поставщиками программного обеспечения.

В самом сервере разница между встроенными приложениями и приложениями, написанными пользователем, не обнаруживается, поскольку все эти приложения используют один и тот же интерфейс сервера, а именно ***интерфейс внешнего уровня***.

Некоторые специальные "служебные" приложения могут оказаться исключениями. Они иногда работают непосредственно на внутреннем уровне системы. Подобные утилиты правильнее относить к встроенным компонентам СУБД, а не к приложениям в обычном смысле.

Клиент управляет пользовательским интерфейсом и логикой приложения. Обычно это рабочая станция с пользовательским приложением, которое:

- принимает от пользователя запрос;
- проверяет синтаксис;
- генерирует запрос к базе данных;
- передает сообщение серверу;
- ожидает поступление ответа;
- форматирует полученные данные для визуализации.

## **Приложения**

- приложения, написанные пользователями;
- приложения, предоставляемые поставщиками.

## Приложения, написанные пользователями

В основном, это обычные прикладные программы, чаще всего написанные либо на языке программирования общего назначения, таком как C++ или Java (COBOL), либо на одном из специализированных языков четвертого поколения (4G):

- общего использования (Cliper, Clarion, dBase, 4GL Cobol/PLI generator, ...);
- запросов к базам данных (SQL, Informix-4GL, ...);
- генераторы отчетов (Oracle Reports, Crystal Reports..);
- языки манипулирования данными (Clarion, IDL, PL/SQL, R, ...)
- ...

## Приложения, предоставляемые поставщиками

Их часто называют *инструментальными средствами*.

В целом, назначение таких средств — упрощать процесс создания и выполнения других приложений, т.е. приложений, которые разрабатываются специально для решения некоторой конкретной задачи.

Часто эти приложения могут выглядеть вовсе не так, как приложения в общепринятом смысле.

Назначение инструментальных средств также состоит в предоставлении пользователям, особенно конечным, возможности создавать приложения без написания традиционных программ.

Например, одно из предоставляемых поставщиком СУБД инструментальных средств может быть генератором отчетов, с помощью которого конечный пользователь сможет получить отформатированный отчет, выполнив обычный запрос к системе.

Каждый такой запрос является по существу небольшим специальным приложением, написанным на языке очень высокого уровня (с конкретным назначением), а именно — на языке формирования отчетов.

Отдельно поставляемые инструментальные средства делятся на несколько самостоятельных классов:

- процессоры языков запросов;
- генераторы отчетов;
- графические бизнес-подсистемы;
- электронные таблицы;
- процессоры команд на естественных языках;
- статистические пакеты (R);
- средства управления копированием или средства извлечения данных;
- генераторы приложений, включая процессоры языков четвертого поколения;
- другие средства разработки приложений, включая CASE-инструменты (CASE, или Computer-Aided Software Engineering – автоматизированное проектирование и создание программ);
- инструментальные средства интеллектуального анализа данных и визуализации.

### **Преимущества технологии «клиент/сервер»**

- более широкий доступ к существующим базам данных;
- повышение производительности, по сравнению с телеобработкой и/или ФС;
- снижение стоимости аппаратного обеспечения (серверная машина – более мощная, чем клиентские);
- снижение сетевого трафика (по сети идут только необходимые данные);
- повышается уровень непротиворечивости данных (проверка целостности данных выполняется централизованно на сервере, а не у клиентов).

## **Расширение двухуровневой технологии «клиент/сервер» трехуровневая структура.**

- тонкий (неинтеллектуальный) клиент, управляющий только пользовательским интерфейсом;
- средний уровень (клиент), управляющий логикой пользовательского приложения;
- сервер базы данных.

# Низкоуровневые операции логического уровня в таблицах

Обычно они реализованы, как операторы 4GL языков в СУБД на основе файловой системы. Мнемоника обобщенная.

```
TABLE LIST (ID, NAME, DATE, ...)
```

```
TABLE ITEM (ID, LIST_ID, PART_NO, NAME, RPOVIDER, QUANT, ...)
```

```
PRIMKEY LIST:BY_ID (ID)
```

```
PRIMKEY ITEM:BY_ID (ID)
```

```
INDEX LIST:BY_NAME (NAME)
```

```
INDEX LIST:BY_DATE (DATE)
```

```
INDEX ITEM:BY_NAME (NAME)
```

```
INDEX ITEM:BY_NAME_INLIST (LIST_ID, NAME)
```

```
INDEX ITEM:BY_PNO_INLIST (LIST_ID, PART_NO) # кандидат в PrimKey
```

## Простейшие сериальные операции

SET(TABLE) – установка курсора (точки доступа) в «сыром» порядке записей

NEXT(TABLE) – просмотр таблицы в направлении от начала к концу

PREV(TABLE) – просмотр таблицы в направлении с конца к началу

INSERT(RECORD, TABLE) – вставка в таблицу без обновления индексов (APPEND)

## Прямой доступ по первичному ключу

GET(TABLE, VALUE) – получить строку (запись) из таблицы по значению перв. ключа

UPDATE(RECORD, TABLE) – обновить строку (запись) в таблице по значению перв. ключа



## **Добавление/удаление**

ADD(RECORD, TABLE) — добавить строку (запись) в таблицу и обновить все индексы  
DEL(TABLE, RECORD) — удалить строку (запись) из таблицы и обновить все индексы  
DEL(TABLE, KEY) — удалить строку (запись) из таблицы и обновить все индексы

## **Прямой доступ по вторичному ключу**

GET(TABLE, INDEX, VALUE) — получить строку по значению ключа

## **Сериальные операции в порядке индексов**

SET(TABLE, INDEX, KEY\_VALUE) — установить курсор доступа в порядке индекса  
NEXT(TABLE, INDEX) — просмотр таблицы в направлении с начала к концу  
PREV(TABLE, INDEX) — просмотр таблицы в направлении с конца к началу