Функции промышленных СУБД

ЛЕКЦИЯ №3

- *1 Эффективное хранение данных и доступ к ним*
- *2 Поддержка системного каталога*
- *3 Обеспечение ссылочной целостности данных*
- *4 Управление буферами*
- *5 Обеспечение транзакционной целостности*
- *6 Организация обмена данными*
- *7 Оптимизация запросов*
- *8 Резервное копирование и восстановление данных*
- *9 Защиты данных и информационной безопасности*
- *10 Поддержка независимости программ от данных*
- *11 Обеспечение средств взаимодействия различных категорий пользователей со средой*

1 Эффективное хранение данных и доступ к ним

Инструменты эффективной организации данных:

- ***Кластеризация*** *- ЛОГИЧЕСКИ СВЯЗАННЫЕ СТРОКИ РАЗЛИЧНЫХ ТАБЛИЦ ГРУППИРУЮТСЯ В ОДНОМ ФИЗИЧЕСКОМ БЛОКЕ ДАННЫХ*
- ***Индексация*** *- ОСНОВНОЙ СПОСОБ СНИЖЕНИЯ КОЛИЧЕСТВА ДИСКОВЫХ ОПЕРАЦИЙ ВВОДА-ВЫВОДА*
- **Хеширование данных** *- ИСПОЛЬЗОВАНИЕ ХЕШ-КЛАСТЕРОВ ПОЗВОЛЯЕТ УСКОРИТЬ ПОИСК НЕОБХОДИМЫХ СТРОК*

Кластеризация

Блок 10

tab1

Id1	D1
101	216

tab2

Tab1_id	id2	d2
101	001	fp
101	005	dp
101	006	cp

Блок 11

tab1

Id1	D1
102	410

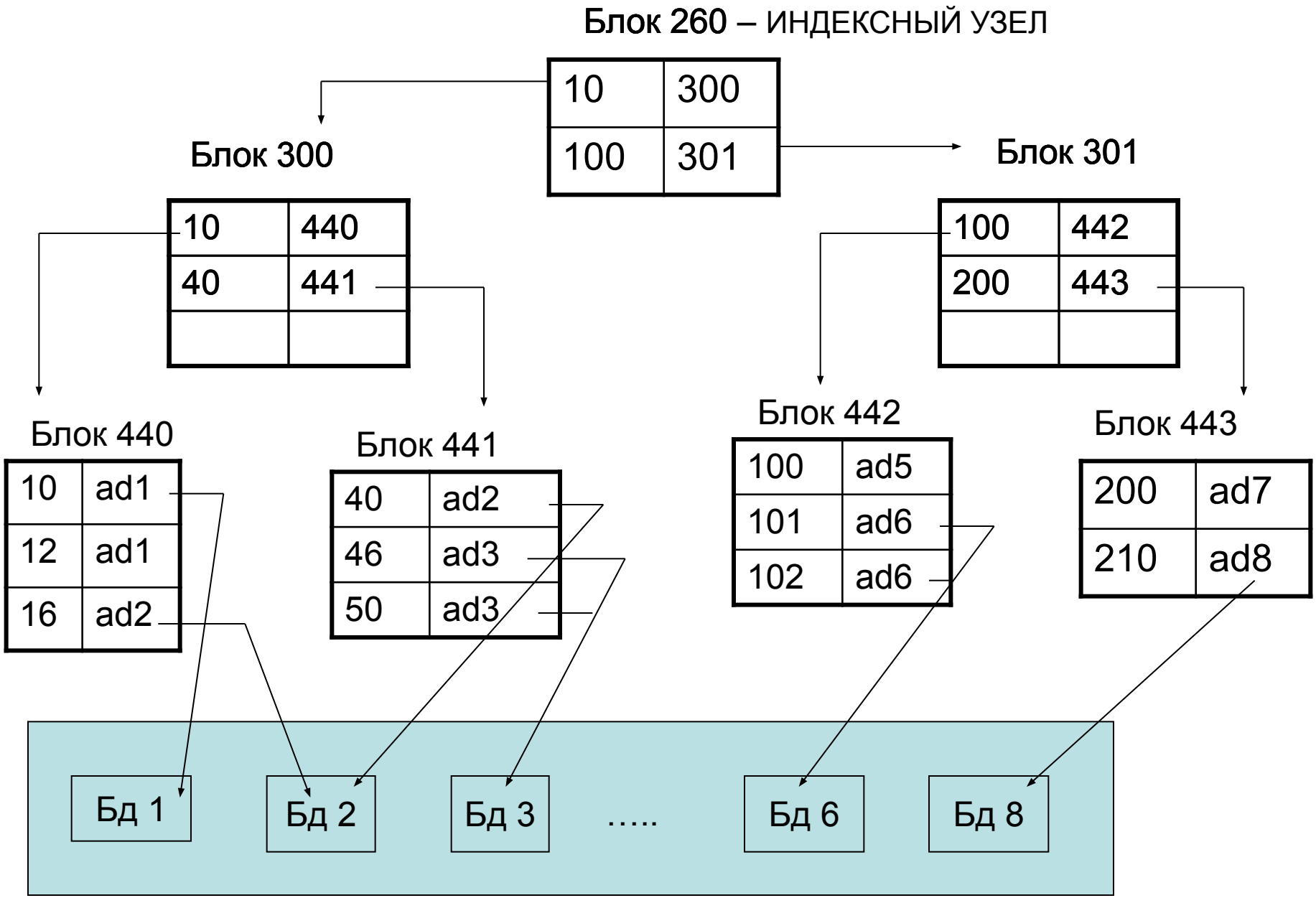
tab2

Tab1_id	id2	d2
102	010	fp
102	020	cp
102	021	kp

Каждому блоку устанавливается кластерный ключ. На рисунке кластерным ключом является первичный ключ Id1 – это физический адрес на диске

кластеры

Индексация


Блок 260 – ИНДЕКСНЫЙ УЗЕЛ
10 300
100 301
Блок 300
10 440
40 441
Блок 301
100 442
200 443
Блок 440
10 ad1
12 ad1
16 ad2
Блок 441
40 ad2
46 ad3
50 ad3
Блок 442
100 ad5
101 ad6
102 ad6
Блок 443
200 ad7
210 ad8
Бд 1
Бд 2
Бд 3
…..
Бд 6
Бд 8

Индексы в виде **B+дерева** целесообразно создавать для атрибутов, у которых значения являются *уникальными.*

Если атрибут содержит значительное количество *неуникальных* значений, необходимо создавать индексы **битовой карты**

	Новочеркасск	**Ростов**	**Шахты**
Иванов	1	0	0
Петров	0	1	0
Сидоров	1	0	0
Ковалев	0	0	1

Этот индекс представляет собой двумерную таблицу, в которой каждый столбец соответствует одному значению индексируемого атрибута, а каждая строка – экземпляру записи в базе данных

В схеме **Sotr,** представленной ниже, атрибут **addres** содержит неуникальные значения, по этому атрибуту создается битовая карта

Sotr

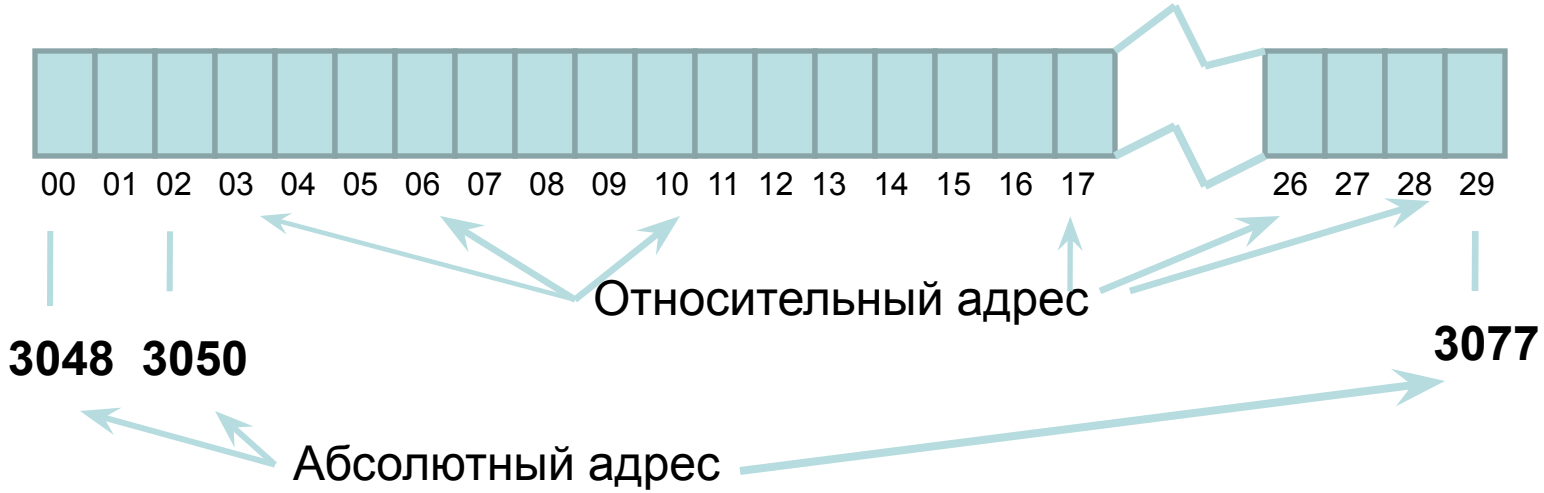
Tab_nom	**fio**	**dol**	**addres**	**data**

Хеширование данных

Недостаток индексных схем – необходимость при поиске записи дважды обращаться к вспомогательному запоминающему устройству: для чтения индекса,

для доступа к записи.

ПРИНЦИП ХЕШИРОВАНИЯ: Заменить затраты времени на поддержание и просмотр индексов на время ЦП для выполнения алгоритма хеширования, генерирующего физический адрес блока размещения записи.



ПРИМЕР, ИЛЛЮСТРИРУЮЩИЙ ЭТОТ МЕТОД:

Требуется записать на диск, позволяющий хранить в блоке до 2000 байт, 500 записей ведомости зарплаты, каждая из которых имеет длину 100 байт. Для этого необходимо 25 блоков. Округляем до 30 блоков.

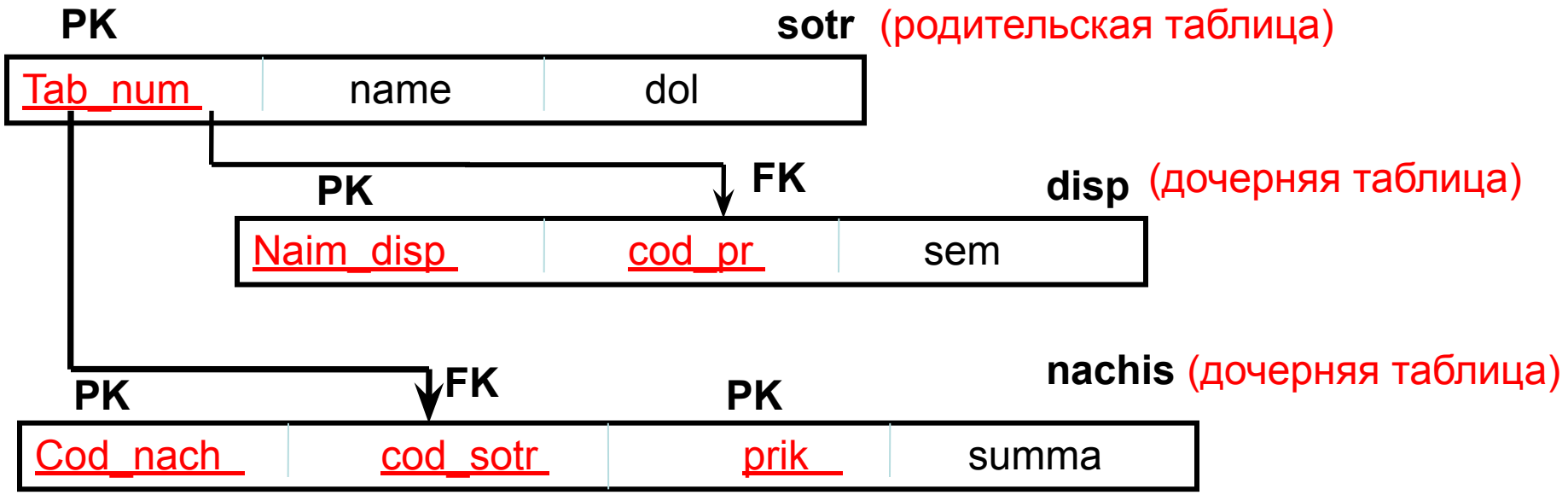
Предположим, что эта последовательность блоков начинается с адреса 3048 и заканчивается 3077. Первая запись, которую необходимо разместить, имеет значение ключа 1562, тогда:

1. *Делим значение ключа на количество блоков. Остаток от деления дает относительный адрес записи:*

1562 : 30 остаток равен 2

2. *Относительный номер блока =* ***2***
3. *Прибавляя результат пункта первого к адресу начального блока, получаем абсолютный адрес блока, в котором будет размещена запись:* ***2 + 3048 = 3050***

3 Обеспечение целостности данных



- Tab_num INTEGER **PRIMARY KEY NOT NULL –** в таблице sotr
- Name_disp INTEGER **PRIMARY KEY NOT NULL**
- **FOREIGN KEY** (cod_pr) **REFERENCES** sotr(tab_num)

} в таблице disp

- **PRIMARY KEY**(cod_nach, prik)
- **FOREIGN KEY** (cod_sotr) **REFERENCES** sotr(tab_num)

} в таблице nachis

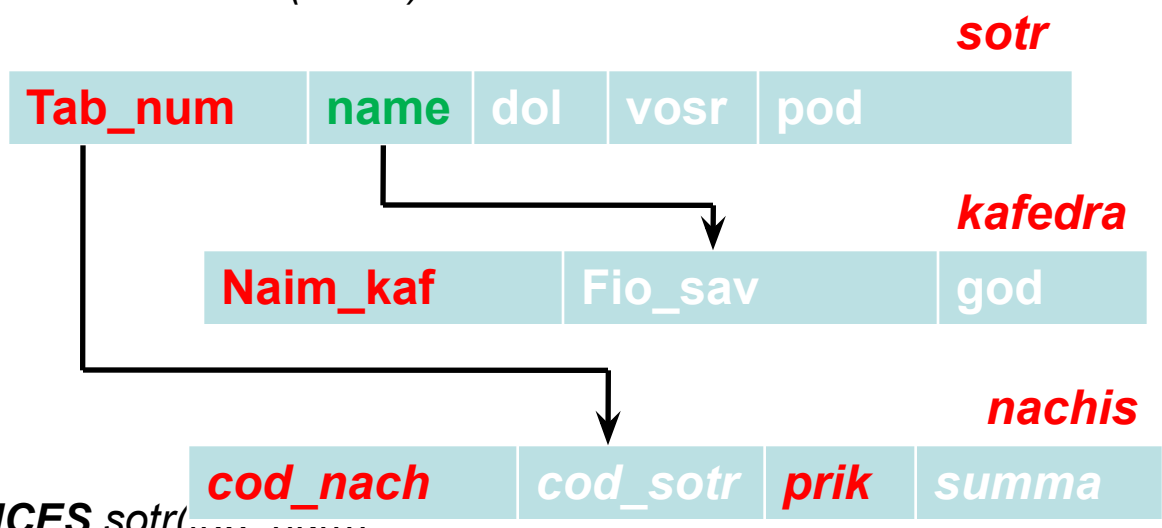
Описание таблиц на языке SQL

ПРИМЕР. Обеспечение целостности **объекта**, **приложения** и **ссылочной** целостности данных

- ***CREATE TABLE sotr*** *(*
  cod NUMBER (4) ***PRIMARY KEY NOT NULL****,*
  fio VARCHAR2(35) ***UNIQUE NOT NULL****,*
  dol VARCHAR2(25) ***CONSTRAINT*** *sotr_dol* ***CHECK*** *(dol* ***IN*** *('АССИСТЕНТ', 'СТАРШИЙ ПРЕПОДАВАТЕЛЬ', 'ДОЦЕНТ', 'ПРОФЕССОР')),*
  st VARCHAR2(3) ***CHECK*** *(st* ***IN*** *('КТН', 'ДТН', 'НЕТ')),*
  sv VARCHAR2(9) ***CHECK*** *(sv* ***IN*** *('ДОЦЕНТ', 'ПРОФЕССОР', 'НЕТ')),*
  unaq NUMBER (4),
  oklad NUMBER (5,2) ***CONSTRAINT*** *sotr_okl* ***CHECK*** *(oklad* ***BETWEEN*** *14000* ***AND*** *25000),*
  vosr NUMBER (2),
  pod VARCHAR2(25),
  adress VARCHAR2(20) DEFAULT 'НОВОЧЕРКАССК'
  );

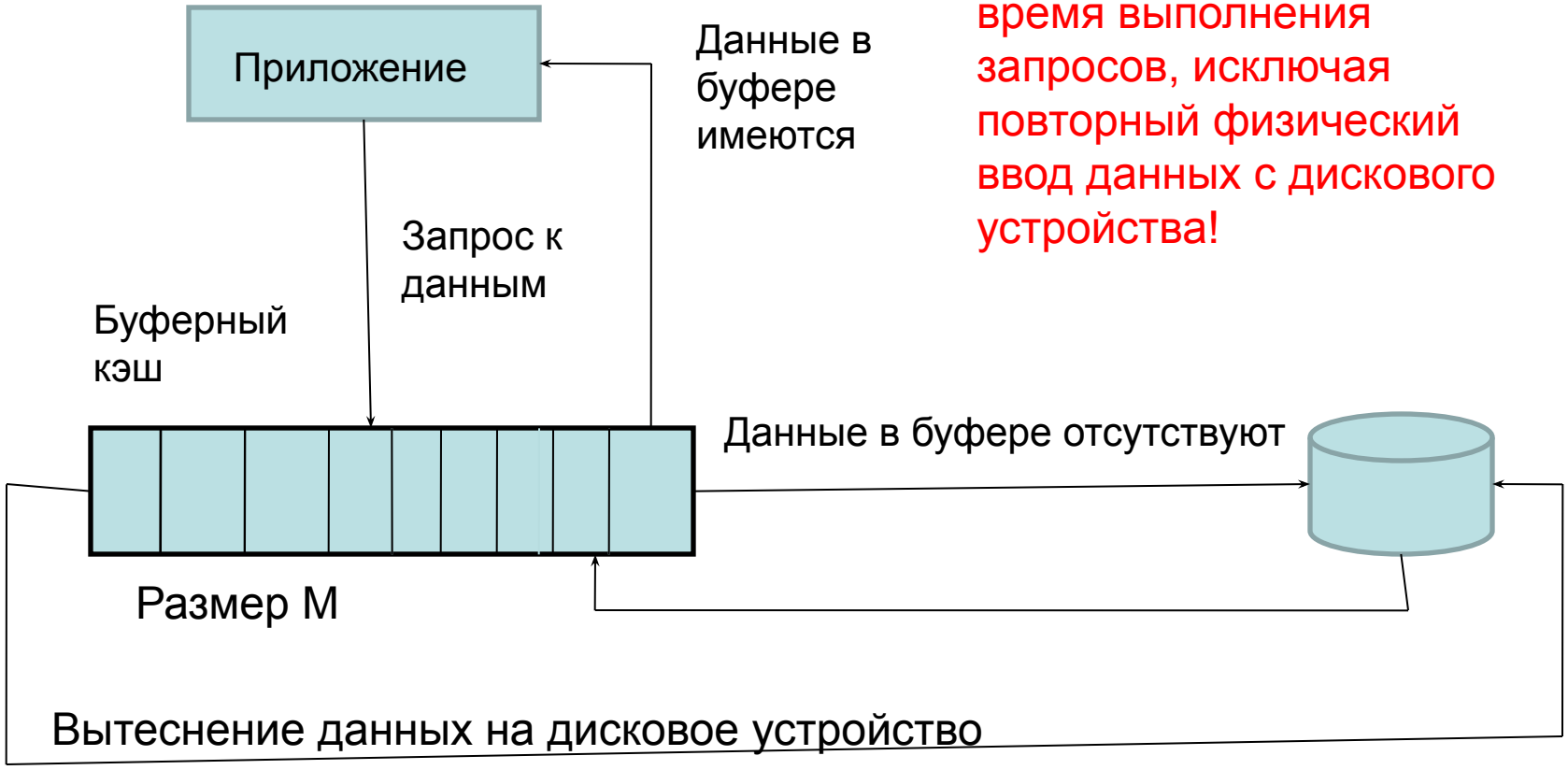
ПРИМЕР. Обеспечение целостности **объекта**, **приложения** и **ссылочной** целостности данных

```
CREATE TABLE sotr
(tab_num  INT PRIMARY KEY NOT NULL,
name  VARCHAR 2(200) UNIQUE,
dol  VARCHAR2 (200),
vosr  INT,
pod  VARCHAR 2(200));
        CREATE TABLE kafedra
         (naim_kaf  VARCHAR 2(200) PRIMARY KEY NOT NULL,
         fio_sav  VARCHAR 2(200) DEFAULT 'ИВАНОВ',
         god  DATE,
         FOREIGN KEY (fio_sav) REFERENCES sotr(name)
         ON UPDATE CASCADE
         ON DELETE DEFAULT);
CREATE TABLE nachis
(cod_nach  INT NOT NULL,
cod_sotr  INT NOT NULL,
prik  CHAR (20) NOT NULL,
summa  DEC (7,2),
PRIMARY KEY (cod_nach, prik),
FOREIGN KEY (cod_sotr) REFERENCES sotr(tab_num)
ON DELETE CASCADE
ON UPDATE CASCADE);
```



4 Управление буферами

Позволяет уменьшить время выполнения запросов, исключая повторный физический ввод данных с дискового устройства!



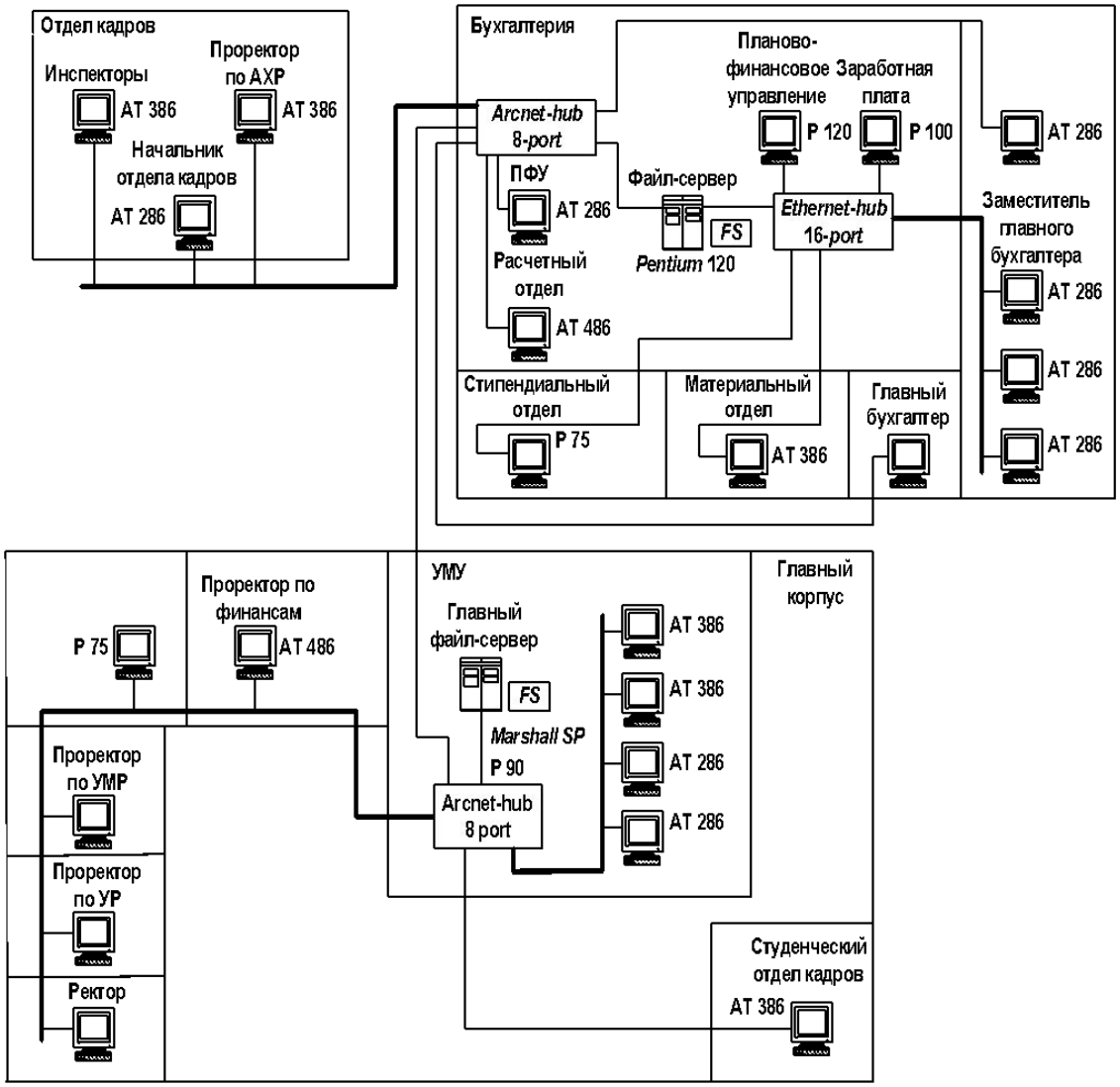
Ориентировано на повторное использование элементов базы данных (результатов выполнения запросов).

5 Обеспечение транзакционной целостности – поддержка свойства изолированности транзакций

Решение проблем:

1 потерянного обновления,
2 чернового чтения,
3 неповторяемого чтения,
4 строк – фантомов

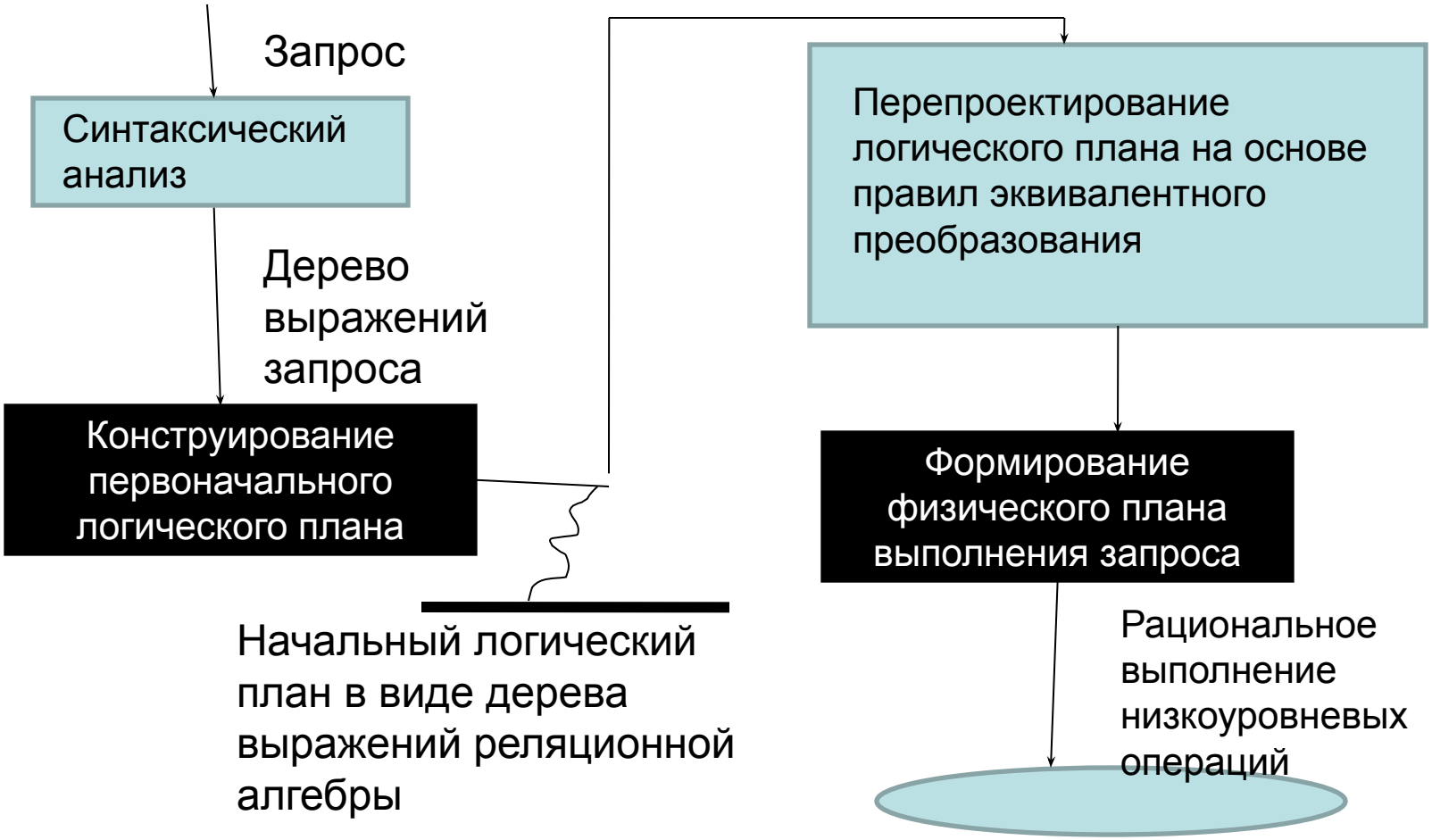
Для исключения перечисленных проблем промышленные СУБД поддерживают ***механизм блокирования и многовариантности данных***



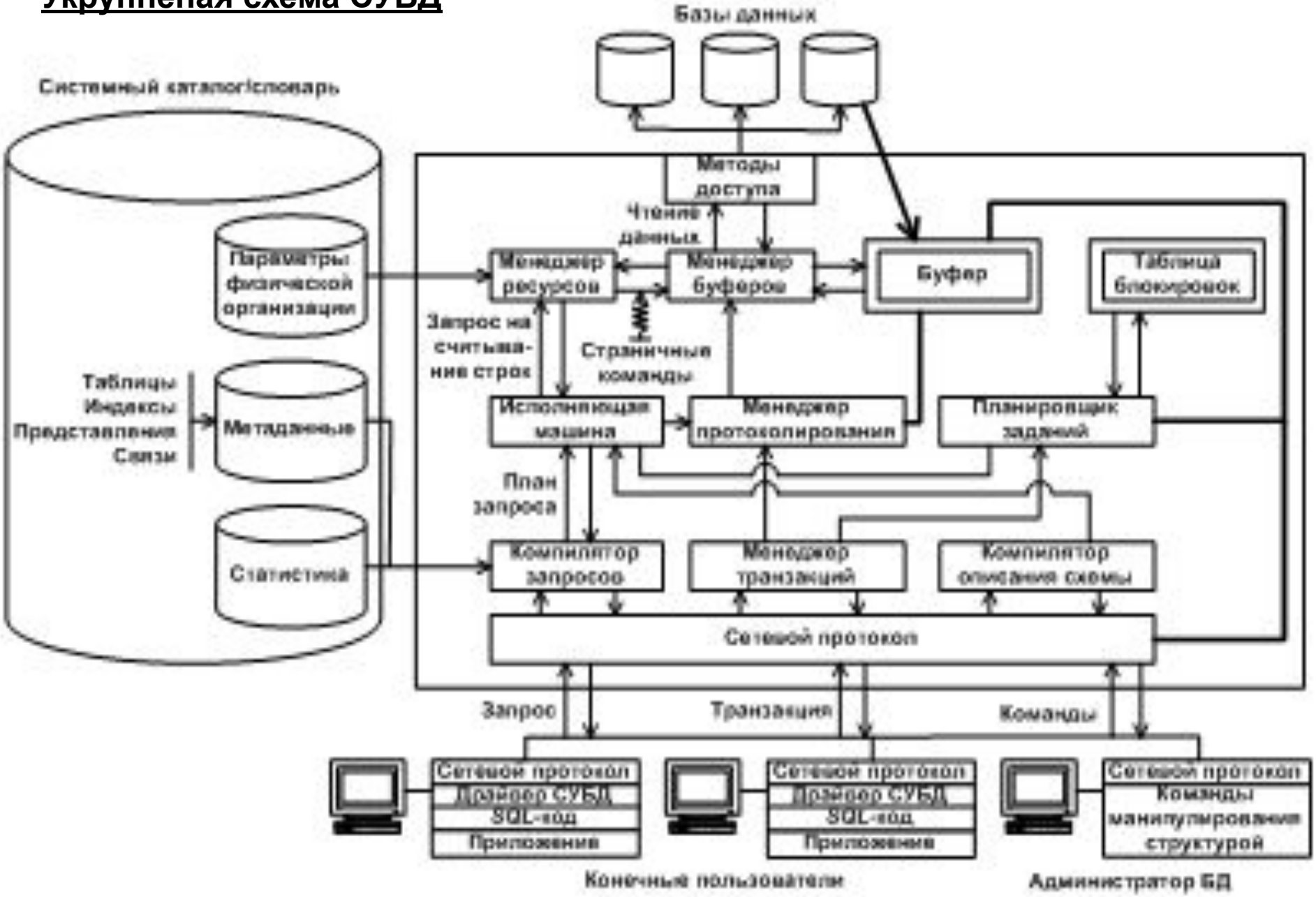
6 Организация обмена данными

- *Организация обмена данными* предполагает дополнение СУБД компонентами, которые позволяют организовать работу СУБД в локальной либо глобальной вычислительных сетях.
- Компоненты выполняют функции:
- - установка **соединений** с удаленными серверами,
- - организация **передачи запросов и данных** между серверами системы,
- - поддержка **расширенного системного каталога**, в котором содержатся сведения о распределении данных в вычислительной сети и т.д.

7 Оптимизация запросов



Укрупненая схема СУБД



Основными компонентами СУБД являются:

- *компилятор запросов* – предназначен для формирования физического плана реализации запроса, включает выполнение стадий синтаксического анализа, создания первоначального логического плана, перепроектирования логического плана на основе правил, конструирования физического плана с учетом статистической информации и описания метаданных;
- *исполняющая машина* – является ядром СУБД, координирующим выполнение каждой из операций, входящих в физический план запроса. Результат представляет собой команды на считывание отдельных строк таблицы.
- *менеджер буферов* – организует в оперативной памяти специальную область, предназначенную для временного хранения страниц, считанных с дискового устройства. К функциям этой компоненты относятся проверка наличия затребованных страниц в буфере и, в случае отсутствия, передача команд методам доступа на считывание необходимых страниц, а также вытеснение из буфера устаревших страниц;

метод доступа – предназначен для организации страничного обмена информацией между дисковым накопителем и оперативной памятью;

менеджер транзакций – осуществляет прием от приложения команд управления транзакциями, включая команды задания начала транзакции, фиксации ее результатов, отката, а также реализует координацию компонент, поддерживающих выполнение транзакции;

менеджер протоколирования – производит фиксацию всех обновлений элементов базы данных в журнале транзакций в соответствии с различными режимами протоколирования, а менеджер восстановления при возникновении сбоя переводит базу данных в новое корректное состояние;

планировщик заданий – предназначен для организации одновременного доступа к элементам данных параллельно выполняемых транзакций. Для обеспечения целостности данных эта компонента СУБД использует механизм блокирования данных;

компилятор описания схемы – применяется для трансляции описания таблиц в последовательность команд, которая передается исполняющей машине, а затем менеджеру ресурсов для изменения метаданных, хранящихся в системном словаре СУБД.

ТЕМА. Традиционная концептуальная модель данных Чена. Классификация атрибутов, сущностей и связей. Степень и кардинальность связи. Сильные, слабые и идентификационно – зависимые сущности

Эта модель впервые была предложена Пин-Шэн Ченом в 1976 году и включала следующие элементы: сущности, атрибуты и связи.

Сущность – абстрактный объект, предназначенный для информационного представления предметов реального мира различной природы либо связанных с ними процессов и событий.

Сущности, используемые для описания однотипных предметов, образуют *класс сущности*. Однотипные объекты имеют одинаковый набор свойств. *Экземпляром класса сущности* является сущность, в которой представлена информация о конкретном объекте описываемой предметной области. Каждому классу сущностей присваивается уникальное имя.

Сущности делятся на сильные и слабые. ***Сильные* сущности** соответствуют объектам, которые представлены в предметной области вне зависимости от наличия других объектов. *Слабые* сущности соответствуют объектам, которые существуют в предметной области только при наличии связанных с ними других объектов. Классу сущностей соответствует набор *атрибутов*, предназначенных для представления его свойств и однозначной идентификации. Атрибуты, используемые для представления свойств сущностей, называются *информационными*. Атрибуты, применяемые для идентификации объектов, позиционируются в качестве *идентификационных* или иначе ключевых атрибутов. Набор значений, которые могут быть присвоены атрибуту, называется *доменом*.

- Атрибуты делятся на ***простые и составные, однозначные и многозначные, исходные и производные***.
- **Простой** атрибут описывает элементарное свойство сущности и не может быть декомпозирован на составные компоненты. В частности, простым атрибутом является атрибут «*Расчетный счет*».
- В качестве **составных** атрибутов могут быть позиционированы такие, как «*Адрес*», «*ФИО*». На схемах простой атрибут изображается в виде овала, а составной атрибут – в виде иерархии связанных овалов.
- **Однозначный** атрибут принимает единственное значение для сущности. В частности, атрибут «*Дата рождения*» имеет единственное значение для сущности «*Персона*». **Многозначный** атрибут содержит произвольное количество значений для одной сущности. Например, для сущности «*Персона*» многозначным атрибутом является атрибут «*Научные публикации*». **Производные** атрибуты характеризуются тем, что они вычисляются на основе значений других атрибутов, которые могут принадлежать различным классам сущностей.

Классы сущностей соединяются между собой поименованными типами связей

- . Каждый уникально идентифицируемый экземпляр типа связи позиционируется как связь, которая представляет собой ассоциацию между сущностями, включающую по одной сущности из каждого участвующего в связи класса сущности. **Степенью** связи называется количество типов сущностей, которые участвуют в этой связи. Для *бинарных* связей степень равна двум, а *тернарная* связь имеет степень равную трем. На *ER*-диаграммах связь изображается в виде ромба с указанием имени связи. Эти связи называются связями типа «*ИМЕЕТ*» или связями обладания, *HAS-A relationships*. Использование такого термина обусловлено тем, что одна сущность имеет связь с другой сущностью. **Например, *Кафедра* имеет *Лаборатории*, *Профессор* имеет *Научные работы* и т.д.**
- **Показатель кардинальности** задает количество возможных связей для каждой сущности, участвующей в связи. В зависимости от значения показателя кардинальности выделяют следующие виды бинарных связей:
- связь типа **«*один-к-одному*»**, обозначаемая как «1:1», представляет бинарную связь, в которой каждой сущности первого из связанных классов сущностей соответствует единственная сущность второго класса и наоборот;
- связь типа **«*один-ко-многим*»,** обозначаемая как «1:*N*», в которой каждой сущности первого из связанных классов сущностей соответствует подмножество сущностей второго класса. В общем случае это подмножество может быть пустым;
- связь типа **«*многие-к-одному*»,** обозначаемая как «*M*:1», в которой каждой сущности первого из связанных классов сущностей может соответствовать единственная сущность второго класса, а каждой сущности второго класса соответствует подмножество сущностей первого класса. В общем случае подмножество может быть пустым;
- связь типа **«*многие-ко-многим*»,** обозначаемая как *M*:*N*, представляет бинарную связь, в которой каждой сущности первого из связанных классов сущностей соответствует подмножество сущностей второго класса и наоборот.

ТЕМА. Проектирование логической модели баз данных

- Разработанная концептуальная модель базы данных является исходной информацией для проектирования логической модели данных.
- Определение 1. *Логическая модель данных* – это совокупность типов записей, связей между ними, операций доступа к данным, правил обеспечения их целостности, соответствующих парадигме одной из общепринятых схем организации данных и независимых от характеристик конкретной СУБД, а также от физических параметров выбранной программно-технической платформы информационной системы.
- Определение 2. *Тип логической записи* – это структурированное описание свойств объекта, информация о котором представлена в базе данных. К общепринятым логическим моделям относятся следующие: **иерархическая; сетевая; реляционная; объектно-реляционная; объектно-ориентированная.**

***Иерархическая модель данных* является исторически первой логической структурой, которая впервые была предложена компанией *North American Rockwell* (США) и в 1967 году была использована корпорацией *IBM в СУБД IMS*.**

- Основу этой модели данных составляют иерархические древовидные структуры, вершины которой соответствуют записям определенного типа, называемых сегментами. Каждый экземпляр родительского сегмента может быть связан с произвольным количеством экземпляров сегментов-потомков. Сегмент, находящийся на верхнем уровне иерархии, является корневым. Тогда иерархическая база данных – это совокупность деревьев, вершиной которых являются различные экземпляры корневых сегментов. Для иерархической структуры определены такие понятия, как текущее состояние, навигационные операции, команды манипулирования данными.
- Необходимо отметить, что логические связи между сегментами при переходе к физическим структурам хранения отражаются в виде указателей, с помощью которых производится адресация физических экземпляров сегментов.

Основными недостатками иерархической модели данных являются следующие:

- вход в базу данных производится, в основном, через экземпляры корневого сегмента;
- навигация по базе данных осуществляется в соответствии с принципом иерархического следования «сверху-вниз» и «слева-направо».
- Эти недостатки предопределяют низкую производительность информационной системы при выполнении непредусмотренных запросов, связанных с доступом к экземплярам сегментов, находящихся на низших уровнях иерархии.

Сетевая модель данных была специфицирована в 1969 году рабочей группой конференции по языкам систем данных _Conference on Data Systems Languages Database Task Group, CODASYL DTBG_ и впоследствии, по существу, стала промышленным стандартом для сетевых СУБД.

- В общем случае сетевая структура данных представляется в виде графа общего вида, вершинами которого являются данные различных типов – от атомарных элементов данных до записей сложной структуры, содержащих логически взаимосвязанные элементы и агрегаты данных. Дуги графа соответствуют связям между записями сетевой структуры. Модель *CODASYL* является частным случаем сетевой модели и состоит из произвольного числа наборов различных типов. Наборы данных представляют собой двухуровневые иерархические структуры, содержащие записи двух типов: запись – владелец набора и запись – член набора. Между этими типами записей поддерживаются связи типа «один-ко-многим». Эта модель данных предоставляет возможность для разработчика системы создания произвольного числа входов в проектируемую структуру данных, что обеспечивает существенное уменьшение времени доступа к данным. В модели данных предусмотрены специальные операции автоматической навигации и манипулирования данными. Одной из первых СУБД, в которой была реализована сетевая модель данных, является СУБД *Integrated Data Store*, *IDS* компании *Ceneral Electric*.

Основной недостаток сетевых систем -

- **сложность и значительная трудоемкость программирования** приложений на встроенных языках, требующее высокой квалификации от разработчиков прикладных программ. Стремительное уменьшение рынка сетевых СУБД в начале 90-х годов связано с изменением парадигмы использования информационных систем. В 70-е и 80-е годы доминировала парадигма, в соответствии с которой разработчик системных и прикладных компонент должен предлагать технические решения, оптимизирующие использование ресурсов вычислительной системы.

- ***Реляционная модель данных*** была предложена в 1969 году преподавателем математики Э.Коддом, который по совместительству работал в Исследовательском центре компании *IBM*, в статье «*A Relational Model of Data for Large Shared Data Banks*» – «Реляционная модель данных для больших распределенных банков данных». В 1981 году за создание реляционной теории баз данных и вклад в ее развитие Э.Кодд был удостоен Тьюринговской премии. Эта модель данных основана на математическом понятии отношения и представлении отношений в виде таблиц. Для формализации операций доступа к данным используются теоретические положения реляционной алгебры и реляционного исчисления.

- ***Объектно-реляционная модель данных*** представляет собой сочетание преимуществ реляционной и объектной модели данных. Основные элементы этой модели представлены в стандарте *SQL*:1999 и базируются на Манифесте баз данных третьего поколения, направленном на защиту реляционной модели данных за счет расширения ее функциональности и поддерживаемых типов данных. К таким новым объектным типам данных относятся: столбцовые объекты, массивы переменной длины, вложенные таблицы и строковые объекты. Объектно-реляционная модель данных поддерживается такими СУБД, как *Oracle* 8*i* и выше, *PostgreSQL*, *Informix* и др.

- ***Объектная модель*** базируется на положениях Манифеста объектно-ориентированных СУБД, включающих тринадцать обязательных характеристик, которым должна удовлетворять любая объектная система. К таким характеристикам относятся: поддержка составных объектов; поддержка идентичности объектов; поддержка инкапсуляции; поддержка типов и классов; поддержка наследования типов или классов от их предков; поддержка динамического связывания; расширяемость наборов данных; вычислительная полнота языка манипулирования данными; поддержка больших баз данных; поддержка параллельной работы многих пользователей; обеспечение восстановления данных после сбоя; предоставление простых средств конструирования запросов к данным. Объектный подход реализован в СУБД *Jasmine* компании *CA*, *Cache* компании *Intersystem* и др.