Основные понятия и определения БД

- **Информационная система (ИС)** – взаимосвязанные данные, принадлежащие одной предметной области.
- **Ядро системы** – хранимые в ИС данные.

Требования к данным:

- доступность;
- изменяемость;
- наполняемость;
- быстрая обработка;
- формирование нужной отчетности.

- **Предметная область** - часть реальной системы, представляющая интерес для профессиональной деятельности (например: учеба, банк, законодательство, медицина).
- **Объект предметной области** - элемент информационной системы, информацию о котором нужно хранить.

Основные понятия и определения БД

- **Класс объектов** - совокупность объектов, обладающих одинаковыми свойствами.

Каждый объект описывается конечным набором свойств – характеристик.

Характеристика = **атрибут** (элемент данных) = **реквизит.**

Пользователь определяет характеристики исходя из решения конкретных задач.

Основные понятия и определения БД

***Атрибут характеризуют*:**

- ✔ уникальное имя – идентификатор;
- ✔ форма представления - тип данных;
- ✔ значение.

Пример.

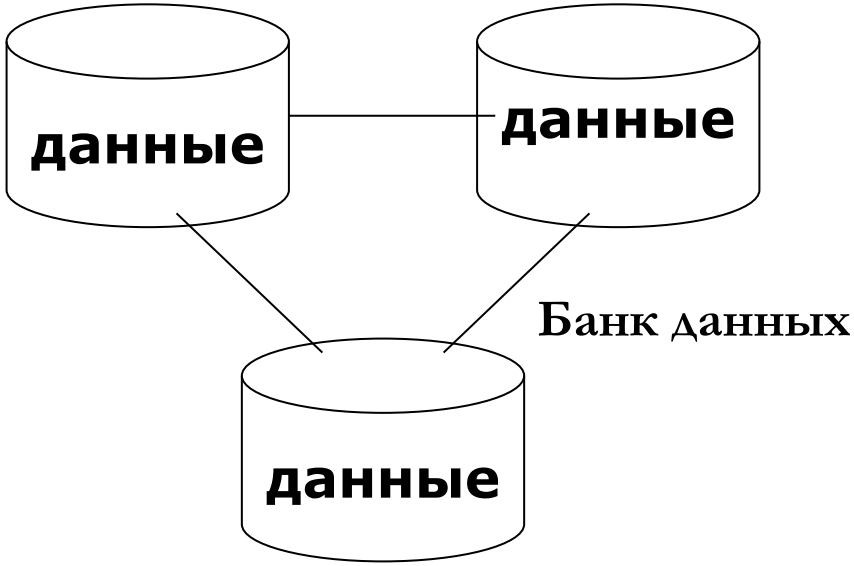
Для атрибута «ФИО»:

- *имя* – ФИО;
- *форма представления* – символьная;
- *значение* – Иванов И. И.

Основные понятия и определения БД

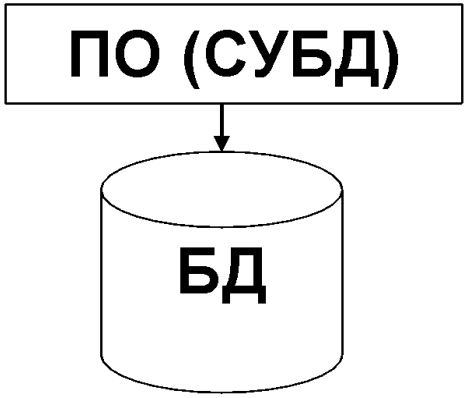
База данных

- ***База данных*** - совокупность связанных данных различного назначения, принадлежащих к конкретной предметной области.



Банк данных

- ***Банк данных*** – более широкое понятие – это несколько взаимосвязанных БД.



СУБД

- ***СУБД*** – это программное обеспечение, аппаратные средства, программируемая логика и процедуры, осуществляющие управление базами данных.

Основные понятия и определения БД

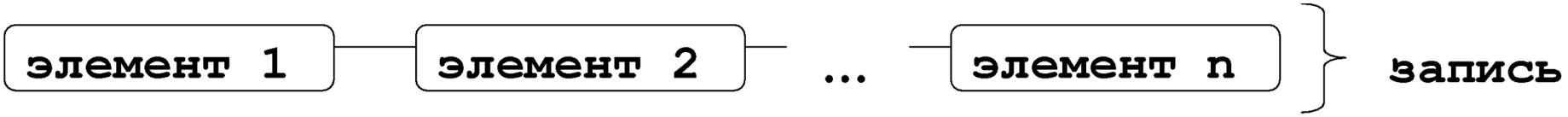
Критерии оценки БД:

1) *количественные* (объем памяти, время обработки запроса, стоимость);
2) *качественные* (доступный интерфейс, адаптируемость к любой предметной области, достоверность данных, защищенность, совместимость с операционными системами).

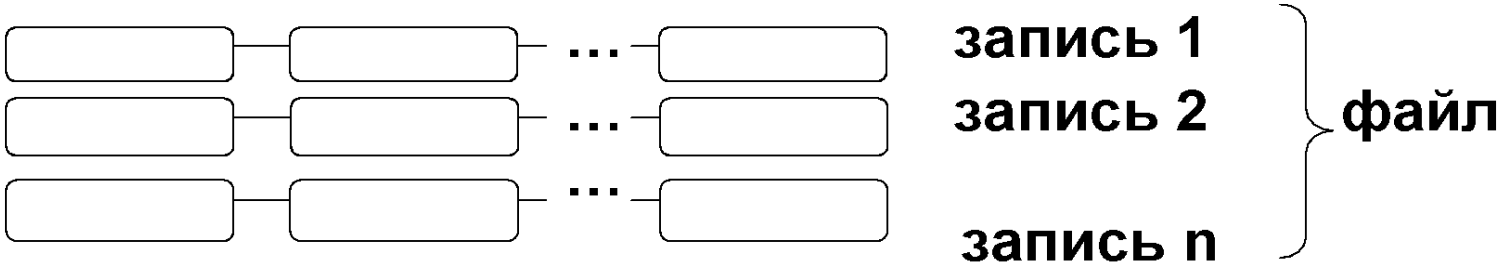
Ведение базы данных – реализация *основных функций пользователя*: ввод, корректировка, поиск, сортировка, вывод данных, удаление и т. д. (независимо от использования прикладных программ). Реализация функций – **система управления базами данных СУБД**.

Основные понятия и определения БД

- **Запись данных** - совокупность значений связанных элементов данных.



- **Файл данных -** совокупность однородных записей.



- **Ключевой элемент данных (ключ) –** такой элемент, по значению которого можно взаимнооднозначно определить значения остальных элементов данных, принадлежащих конкретной записи.

Основные понятия и определения БД

Пример:

Ф. И. О.	группа	предмет	дата экзамена	оценка
Петров А.С.	М-47	высш. матем.	20.05.98	хорошо
Петров А.С.	М-47	Физика	27.05.98	неудовл
Иванов С.А.	М-47	высш. матем.	20.05.98	отл
Сидоров А.А.	М-57	высш. матем.	20.05.98	удовл

Ключи:

- Ф.И.О.+ предмет;
- Ф.И.О.+ дата экзамена (если в один день сдается только один экзамен).

- **Первичный ключ** – это атрибут или группа атрибутов, которые единственным образом идентифицируют строку в таблице.

- **Альтернативный ключ** – это атрибут или группа атрибутов, несовпадающие с первичным ключом и также единственным образом идентифицируют строку в таблице.

Основные понятия и определения БД

- ***Словарь данных*** - централизованное "хранилище" сведений об объектах: составляющие их элементы данных, взаимосвязи между объектами, значения, формы представления.

  Словарь обеспечивает унификацию и единообразие представления данных.

- ***Администратор базы данных*** – лицо (группа лиц), на которое возложено управление средствами БД.

  В его функции входит взаимодействие с руководством предприятия, организации и с пользователями, обрабатывающими данные.

- ***Жизненный цикл БД*** - концепция, в рамках которой рассматривается *развитие БД во времени*.

  Жизненный цикл БД имеет 2 фазы:

1) *фаза анализа и проектирования*: сбор требований пользователей и проектирование самой базы данных ("бумажный" этап);
2) *фаза эксплуатации:* машинная реализация и доработка базы данных.

Основные этапы проектирования БД

Основные этапы проектирования БД

- **Общие информационные требования** – это требования, которые выдвигаются различными пользователями к содержанию проектируемой базы данных.
- **Требования обработки** – это требования, выдвигаемые пользователями к процессу обработки данных в базе.
- **Характеристики СУБД** – это свойства, которые должны бать присущи проектируемой СУБД.
- **Характеристики ОС и технических средств** – это возможности, которые предоставляют для проектирования выбранные технические и программные средства.

Основные этапы проектирования БД

I этап. Формулировка и анализ требований

Цель: сбор требований, предъявляемых к содержанию и процессу обработки данных различными пользователями

Путь: тестирование или анкетирование пользователей

Результат: спецификация требований. Определяются требования к БД, которые документируются в форме, доступной пользователям и разработчикам БД

Основные этапы проектирования БД

II этап. Концептуальное проектирование

Цель: построение независимой от СУБД информационной структуры (объединение требований от пользователей)

→

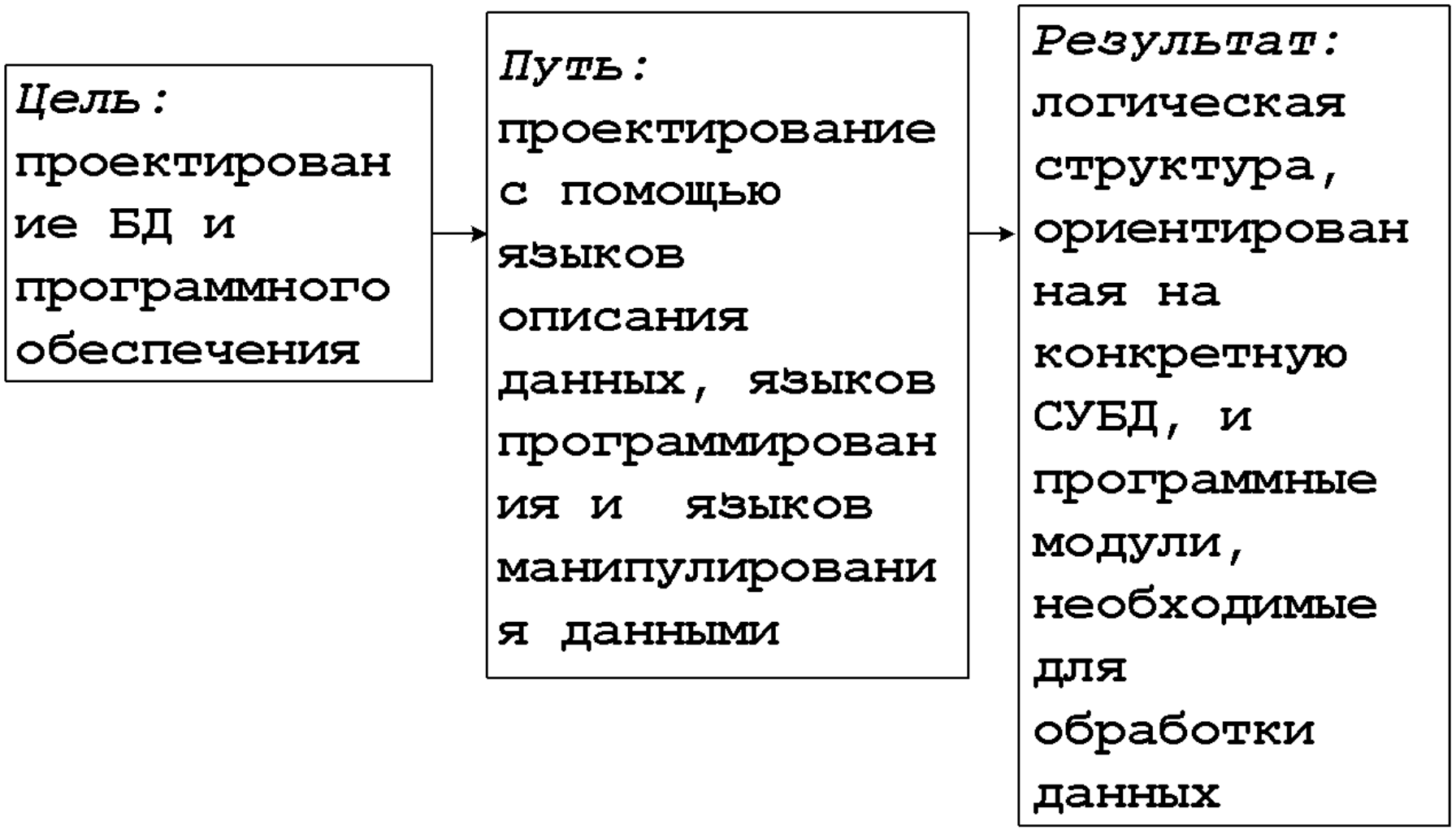
Путь: построение диаграмм «сущность-связь»

→

Результат: информационная структура. Представление информационных требований пользователей в виде диаграмм "сущность-связь".

Основные этапы проектирования БД

III этап. Проектирование реализации.



Основные этапы проектирования БД

IV этап. Физическое проектирование.

Цель: перенос данных на машину, создание физической структуры БД, проектирование оптимальных путей доступа к данным, выбор форматов для хранения записей

→

Путь: используются различные технические и программные средства.

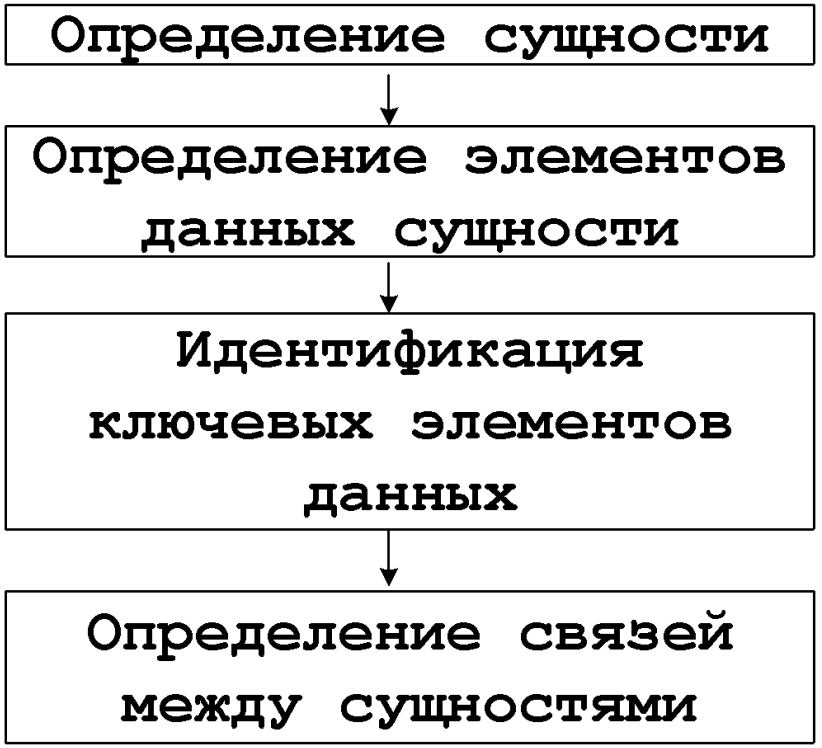
→

Результат: физическая структура (сама БД, представленная на материальном носителе).

Основные этапы проектирования БД

- **Сущность** – это основное содержание того явления или процесса, о котором необходимо собрать информацию (человек, место, явление, процесс и т. д).
- **Тип сущности** – набор однородных вещей, предметов, явлений, выступающих как единое целое (компьютер, служащий).
- **Экземпляр сущности** - конечный набор данных, которые относятся к конкретной сущности.

Общий подход к построению диаграмм “сущность-связь”:

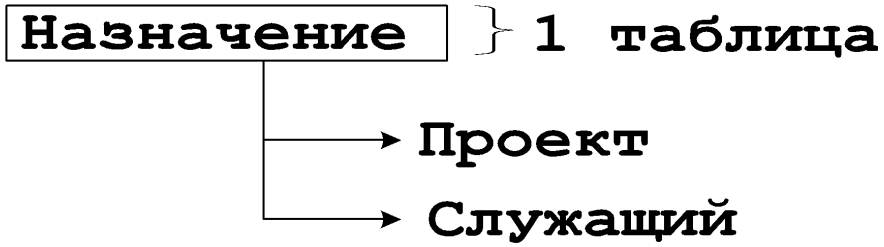


Основные этапы проектирования БД

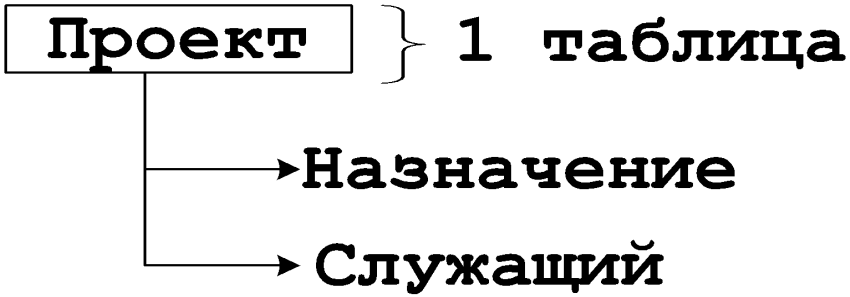
Сущности описываются атрибутами. Можно описать одни и те же факты различными способами. Например, факт: **назначение служащего для работы над проектом**, можно представить в виде трех диаграмм:

1) Сущность – *Служащий, Проект*; взаимосвязь - *Назначение.*

2) Сущность - *Назначение;* атрибуты сущности - *Служащий, Проект.*



3) Сущность - *Проект;* атрибуты сущности - *Служащий, Назначение.*



Основные типы данных

Основные типы данных

Простые типы данных.

Простые типы данных характеризуются тем, что не обладают внутренней структурой и представляют собой конечный набор типов, называемых также *элементарными, примитивными или базисными.*

- **Целый** и **вещественный типы (integer** и **real)** предназначены для числовых данных. Целые формируются из целочисленных значений, вещественные - с участием дробной части. Минимальные и максимальные значения определяются возможностями СУБД. Иногда эти два типа объединяются в тип **number**.

Основные типы данных

Простые типы данных.

- **Указатели (pointer)** применяются, когда данные расположены в различных участках памяти, и необходимо обеспечить некоторую цепочку обработки этих данных.

  Указатель = адрес следующего в цепочке данного.

  Обычно указатели представляют целым числом.

- **Логический тип (boolean)** представляет тип данных, состоящий из наборов переменных для описания данных, которые принимают два значения: истина, ложь.

- **Перечисляемый тип данных** обладает свойством перечисления принадлежащих ему данных.

  Например: дни недели, месяцы.

- **Интервальный тип (interval)** можно рассматривать как обозначение интервала любого заранее определенного типа с возможностью перечисления входящих в него значений.

  T = 00.00 … 24.00 (min … max)

Основные типы данных

Структурированные типы данных.

Структурированные типы данных предназначены для построения данных, обладающих внутренней структурой, из простых (базисных) типов.

- **Массив (array) -** конечное множество переменных фиксированного типа, объединенных одним общим именем.

  Одномерный массив - вектор, двумерный – матрица.

- **Прямой доступ** - *доступ* к любому элементу массива, осуществляемый с помощью указания имени массива и задания одного или нескольких индексов.

- **Файл** – упорядоченная совокупность записей.

Основные типы данных

Структурированные типы данных.

- **Фрейм (frame)-** структура данных, предназначенная для представления знаний в конкретной предметной области.

Фрейм состоит из отдельных полей - **слотов**, которые заполнены содержательными понятиями предметной области. Поля фрейма связаны между собой отношениями, реализованными в виде отдельных процедур или формул.

Фрейм является открытой структурой, т.е. в него могут добавляться (исключаться) некоторые поля; благодаря этому можно строить сеть, состоящую из отдельных фреймов.

Пример. Фрейм *«Сопротивление»*

- *Слот 1*: Сила тока I, А
- *Слот 2*: Напряжение U, В

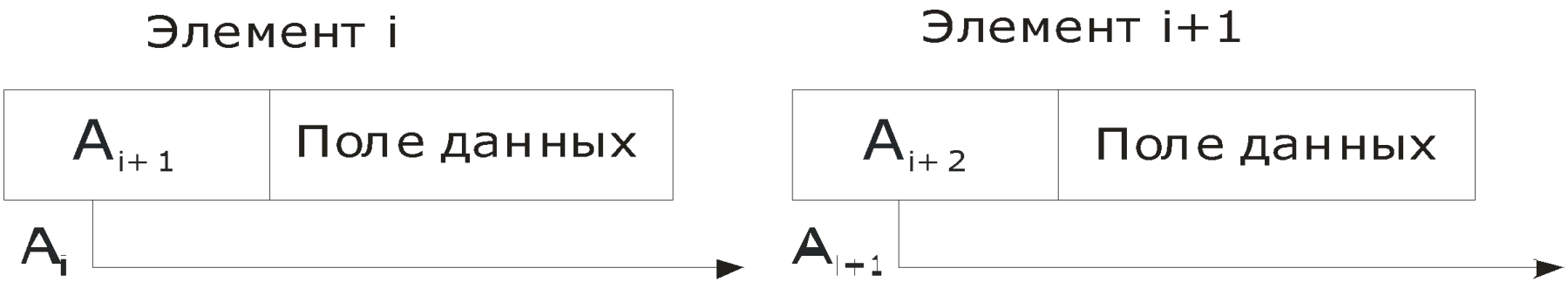
$$R = \frac{U}{I}$$

Основные типы данных

Ссылочные типы данных.

- **Список (list) -** конечный набор элементов, расположенных в произвольных участках памяти и связанных между собой с помощью указателей.

Последовательность обработки элементов списков задается с помощью указателей (адресов связи).



A – адрес связи

- **Адрес связи** i-го элемента - это начальный адрес (i+1)-го элемента списка.

Списки могут быть:

- простые;
- сложные.

Основные типы данных

Ссылочные типы данных.

- **Простым** называется список, все элементы которого являются неструктурированными, простыми.

К простыми спискам относятся стек, магазин, очередь.

- *Стек:* последний пришел - первый ушел.
- *Магазин:* отличается от стека тем, что в нем отсутствуют указатели (элементы расположены по порядку в памяти).
- *Очередь:* первый пришел - первый ушел.

- **Сложным** является список, в котором элемент тоже может быть списком.

  К сложным может быть отнесен *двухсвязный список* - тип списка, в котором доступ к данным осуществляется в двух направлениях: каждый элемент имеет два указателя, на следующий и предыдущий элементы.

- **Древовидная структура** данных представляет собой некоторое дерево (граф без циклов), вершинами которого являются элементы списка.

Модели представления данных

Модель данных отражает взаимосвязи между описанными в БД объектами.

Различие – в представлении взаимосвязей:

- между объектами;
- между атрибутами одного объекта;
- между атрибутами разных объектов.

Взаимосвязи бывают:

1) один к одному **1:1** (⟷);
2) один к многим **1:M** ( );
3) многие к одному **M:1** ( );
4) многие к многим **M:M** ( ).

Область отправления ⟶ **Область значения**

Модели представления данных

Иерархическая модель данных.

- **Узлы** – объекты или атрибуты.
- **Ветви** – взаимосвязи между атрибутами или объектами.
- **Корневой узел** – вершина модели.
- Узлы, расположенные на i-том уровне, называют **порожденными** относительно узлов i-1-ого уровня.
- Узлы, расположенные на i-1-ом уровне, называют **исходными** по отношению к узлам i-того уровня.

Свойства иерархических моделей:

- иерархия начинается с корневого узла;
- каждый узел состоит из одного или более элементов данных, характеризующих объект;
- каждый узел, расположенный на i-ом уровне, соединен только с одним узлом на (i-1)-ом уровне;
- исходный узел может иметь в качестве зависимых один или несколько порожденных узлов;
- пути доступа к каждому из узлов являются уникальными.

Модели представления данных

Иерархическая модель данных.

Достоинства модели:

+ простота понимания и использования;
+ обеспечение независимости данных.

Недостатки модели:

— трудность добавления и удаления объектов (необходимость изменения схемы целиком);

— увеличение времени доступа к данным (за счет прохода промежуточных узлов).

- **Схема -** описание логической структуры БД **.**

Схема содержит:

- имена объектов;
- атрибуты объектов;
- взаимосвязи между ними.

Модели представления данных

Иерархическая модель данных.

- **Экземпляр схемы** - схема с конкретными значениями в соответствующих элементах данных.
- **Иерархическая модель** – реализация только простых взаимосвязей (1:1, 1:M).

Пример.

Построить иерархическую схему БД *«Предприятие»,* устанавливающую взаимосвязи между 5 объектами:

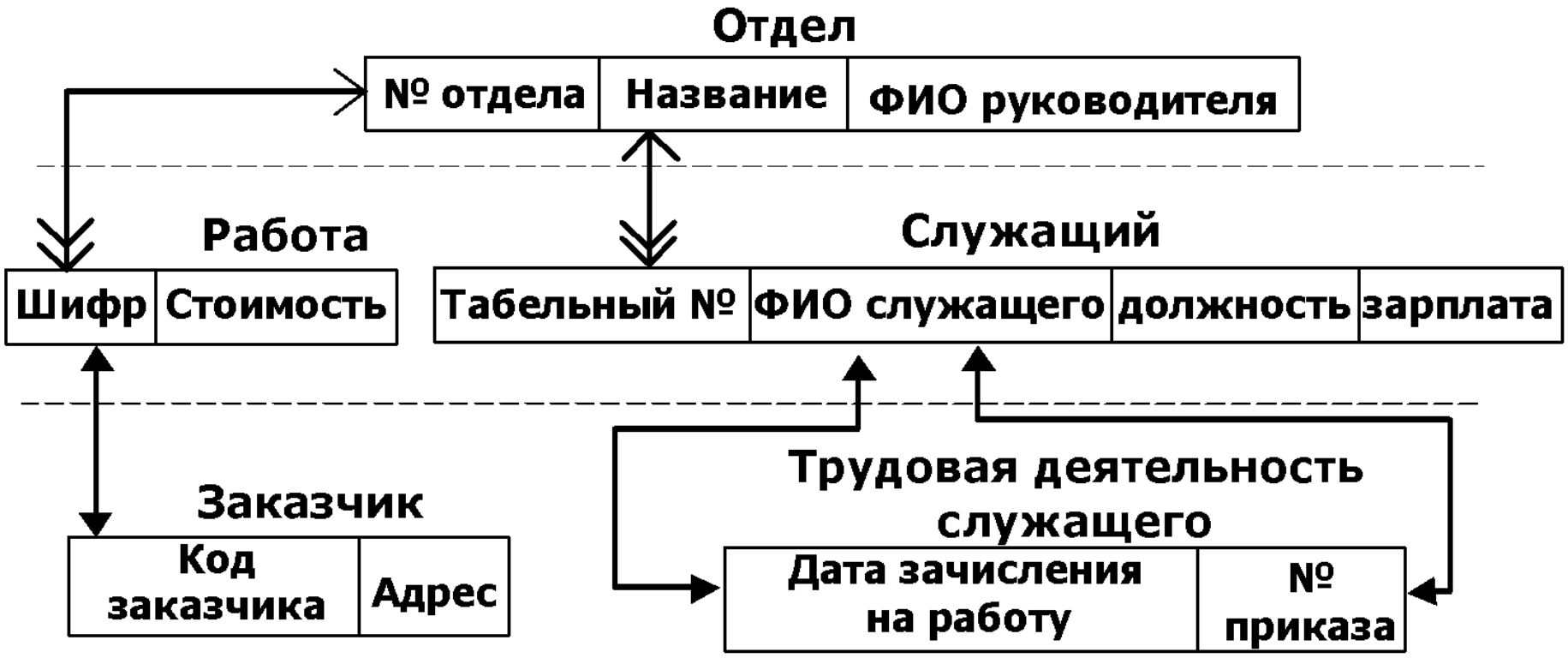
1) *Отдел* (№ отдела, название, ФИО руководителя);
2) *Работа* (шифр, стоимость);
3) *Служащий* (табельный номер, ФИО служащего, должность, зарплата);
4) *Заказчик* (код, адрес);
5) *Трудовая деятельность служащего* (дата зачисления на работу, № приказа).

Модели представления данных

Иерархическая модель данных.

Пример.

Схема БД «Предприятие»:



Модели представления данных

Сетевая модель данных.

- **Сетевая модель** – реализация взаимосвязей *всех 4-ех типов*.

! Любой элемент сетевой структуры может быть связан с любым другим ее элементом.

Количество взаимосвязей $\geq$ **Количеству объектов (атрибутов)**

В простой модели связь между элементами данных в отношении исходный - порожденный является однозначной.

! Взаимосвязи *M : M* могут существовать *только для объектов*, но не для атрибутов данных.

Модели представления данных

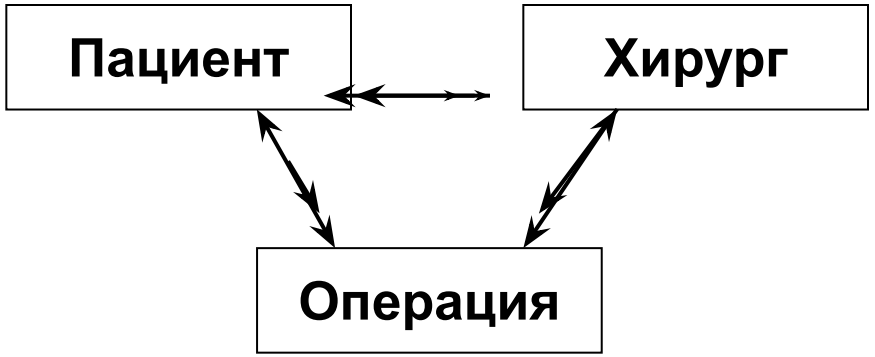
Сетевая модель данных.

Пример.

БД содержит 3 объекта:

пациент, *операция*, *хирург*.

Сетевая модель:



- Пациент может перенести более 1-ой операции – 1:М.
- Хирург произвел более 1-ой операции – 1:М.
- Хирург оперирует более 1-ого пациента – 1:М.

Достоинства модели:

+ простая реализация добавления и удаления (вносится корректировка во взаимосвязи);
+ возможность реализации взаимосвязи М : М.

Недостатки модели:

— сложность физической реализации;
— возможная потеря независимости данных при реорганизации БД.

Модели представления данных

Реляционная модель данных.

- **Реляционная модель** – совокупность данных в виде *таблиц, или отношений*.

+ Обширная теоретическая база;
+ Форма доступна любому пользователю;
+ Освобождена от недостатков, присущих иерархической и сетевой моделям.

- **Отношением R** на множествах **D1, D2, … Dn** называется подмножество их декартового произведения **R ⊆ D1×D2×…×Dn**.
- Множества **D1, D2, … Dn - домены**.
- n - **степень отношения R,** количество доменов.
- **Кортеж** - упорядоченная совокупность элементов множества **ki = < d1, d2, … dn >.**
- **Мощность отношения** - количество кортежей.

Модели представления данных

Реляционная модель данных.

Пример:

Имеются два множества:

1) *ФИО* = {Иванов С.И., Петров Р.А., Зайцев В.Н.};
2) *Учебная группа* = {А-56, Р-81, М-17}.

Зададим *отношение «**Студент**»:*

Студент = ФИО ^{x} Учебная группа = {<Иванов С.И., А-56>, <Иванов С.И., Р-81>, <Иванов С. И., М-17>, <Петров Р.А., А-56>, …, <Зайцев В.Н., М-17>}

Мощность данного отношения ||*Студент*||=9 (3×3).

Степень отношения = 2.

ФИО	**Учебная группа**
Иванов С.И.	А-56
Иванов С.И.	Р-81
Иванов С.И.	М-17
Петров Р.А.	А-56
. . .	. . .
Зайцев В.Н.	М-17

Модели представления данных

Реляционная модель данных.

- Запись вида **R(A, B)** называется **схемой отношения** на множествах A и B.
- **Экземпляром схемы** называется заполненная данными таблица.
- Совокупность схем отношений составляет **схему реляционной БД.**

Термин	**Альтернативный термин**	**Приблизительный эквивалент**
Отношение	Таблица	Файл
Домен	Столбец, атрибут	Поле
Кортеж	Строка	Запись, экземпляр данных
Ключ	---	Идентификатор записи

Модели представления данных

Реляционная модель данных.

Свойства реляционной модели:

- отсутствуют одинаковые кортежи;
- порядок столбцов не существенен и задается пользователем;
- порядок кортежей не существенен и задается пользователем;
- все домены отношения R имеют атомарный характер (т.е. являются неделимыми и не могут быть отношениями).

Недостаток модели

— сложность при оценке производительности.

Достоинства модели:

+ простое понимание и наглядное представление;
+ возможность обработки непроцедурных запросов;
+ наличие обширной теоретической базы;
+ независимость данных, т.е. интерфейс пользователя не связан с деталями организации физической структуры.

Операции в реляционной БД

Любой алгебраический подход предполагает наличие операндов и совокупности операций над ними. В реляционной алгебре в качестве **операндов** выступают отношения, или таблицы, а в качестве **операций**:

- объединение;
- пересечение;
- разность;
- декартово произведение;
- проекция;
- соединение;
- деление.

Операции *объединения, пересечения и разности* выполняются только над парами совместимых отношений.

- Два отношения R и S называются **совместимыми**, если у них:

1) равны степени;
2) соответствующие атрибуты принадлежат одинаковым доменам.

Операции в реляционной БД

- **Объединением** отношений **R** и **S** называется отношение **Q = R ∪ S,** которое состоит из кортежей, принадлежащих либо отношению **R**, либо отношению **S**.

  {k: k∈R ∨ k∈S}, где **k** – кортеж отношения

- **Пересечением** отношений **R** и **S** называется отношение **Q = R ∩ S,** которое состоит из кортежей, принадлежащих одновременно и отношению **R**, и отношению **S**.

  {k: k∈R ∧ k∈S}

- **Разностью** отношений **R** и **S** называется отношение **Q = R \ S,** которое состоит из кортежей, принадлежащих отношению **R** и не принадлежащих отношению **S**.

  {k: k∈R ∧ k∉S}

Q = R \ S ≠ S \ R - операция разности *некоммутативная.*

Операции в реляционной БД

Пример.

Студенты группы А-50 **(R):**

ФИО	№ билета
Кузнецов П.И.	030097
Исаев В.Р.	030098
Щербаков В.В.	030099
Андреев А.А.	030101
Чернов С.П.	030100

Студенты спецгруппы ФАВТ по английскому языку **(S):**

ФИО	№ билета
Орлов П.А.	030214
Щербаков В.В.	030099
Дмитриев С.Н.	030201
Исаев В.Р.	030098
Смирнов А.С.	030185

Операции в реляционной БД

Пример.

R∩S – студенты из А-50, обучающиеся в спецгруппе:

ФИО	№ билета
Щербаков В.В.	030099
Исаев В.Р.	030098

R∪S – все студенты из двух групп:

ФИО	№ билета
Кузнецов П.И.	030097
Исаев В.Р.	030098
Щербаков В.В.	030099
Чернов С.П.	030100
Андреев А.А.	030101
Орлов П.А.	030214
Дмитриев С.Н.	030201
Смирнов А.С.	030185

Операции в реляционной БД

Пример.

R\S – студенты А-50, не обучающиеся в спецгруппе:

ФИО	№ билета
Кузнецов П.И.	030097
Андреев А.А.	030101
Чернов С.П.	030100

S\R – студенты спецгруппы не из А-50:

ФИО	№ билета
Орлов П.А.	030214
Дмитриев С.Н.	030201
Смирнов А.С.	030185

Операции в реляционной БД

- **Декартовым произведением** отношений **R** и **S** называется отношение **Q = R⊗S,** которое представляет собой множество кортежей, полученных *конкатенацией* (приписыванием) каждого кортежа отношения **R** с каждым кортежем отношения **S**.

$$\{ (r \,||\, S) : r \in R \;\&\; s \in S \}$$

- ✔ *Степень* результирующего отношения Q равна сумме степеней исходных отношений **R** и **S**:

степень **Q** = степень **R** + степень **S**

- ✔ *Мощность* результирующего отношения **Q** равна произведению мощностей **R** и **S.**

$$|Q| = |R| * |S|$$

Операции в реляционной БД

Пример операции «декартово произведение».

R =

A	B
a1	b1
a2	b2

S =

C	D	E
c1	d1	e
c2	d1	e
c3	d2	e

Q = R⊗S =

A	B	C	D	E
a1	b1	c1	d1	e
a1	b1	c2	d1	e
a1	b1	c3	d2	e
a2	b2	c1	d1	e
a2	b2	c2	d1	e
a2	b2	c3	d2	e

Операции в реляционной БД

- **Проекция** представляет собой выборку из каждого кортежа отношения **R** значений атрибутов, входящих в домены, по которым производится проекция, и удаление из полученного отношения повторяющихся строк.

  Операция проекции является *унарной*.

- **Селекция (выборка, ограничение)** - операция, осуществляющая выбор из отношения **R** кортежей, удовлетворяющих условиям выборки.

$$\mathbf{R[A\Theta U]: \{k: k \in R \ \& \ (k[A]\Theta U)\},}$$

где:

- **A** – домен;
- **Θ={=, ≠, >, ≥, <, ≤ }** – множество условий выборки;
- **U** – константа, значение которой принадлежит или не принадлежит множеству значений домена **A**, т.е. $\mathbf{U \in \{A\} \vee U \notin \{A\}.}$

Операции в реляционной БД

Соединение – одна из важнейших операций реляционной алгебры, позволяет создавать таблицы на основе имеющихся. Является *бинарной* операцией (выполняется над парами отношений).

Имеет несколько **разновидностей:**

1) тета-соединение;
2) экви-соединение;
3) естественное соединение.

Соединение может выполняться над парами *несовместимых* отношений.

- **Соединением** двух отношений **R** и **S** называется отношение **Q**, в которое могут входить домены отношений **R** и **S** и/или их скалярное выражение. Соединение есть результат последовательного применения декартового произведения и селекции.

Если в отношениях R и S имеются *атрибуты с одинаковыми именами*, то:

степень Q = степень R + степень S – число повторяющихся атрибутов.

Операции в реляционной БД

Θ–соединение над отношениями R и S по атрибутам A и B:

$$Q = R[A\Theta B]S; \quad A \in R, \; B \in S,$$

где **Θ** – одна из операций сравнения (аналогично селекции).

Пример.

R =

A	C
a1	10
a2	20
a3	30

S =

B	D
5	d1
20	d2

Q = R[C>B]S =

A	C	B	D
a1	10	5	d1
a2	20	5	d1
a3	30	5	d1
a3	30	20	d2

Экви-соединение – частный случай Θ – соединения; когда используется операция сравнения «равно» (=).

Q = R[C=B]S =

A	B	C	D
a1	20	20	d2

Операции в реляционной БД

Имеются отношения R (A, B, X, Y, Z) и S (C, D, E, X, Y, Z); X, Y, $Z \subseteq R \,\&\, S$

- **Естественным соединением** отношений **R** и **S** называется отношение **Q = R** ⋈ **S**, которое состоит из множества кортежей, таких, которые принадлежат одновременно как отношению **R**, так и отношению **S.**

Операцию естественного соединения можно разбить на следующие **этапы:**

1) выполнить декартово произведение отношений;
2) выполнить выборку по совпадающим значениям атрибутов, имевших одинаковые имена;
3) выполнить проекцию по совпадающим атрибутам.

Операции в реляционной БД

Операция **деления** выполняется над отношениями, которые имеют хотя бы один общий атрибут.

Рассмотрим два отношения **R** и **S**, которые имеют общие атрибуты: **A∈R** и **B∈S**.

Деление $Q = R[A \div B]S$ выполняется в **два этапа**:

I. Выбираются допустимые кортежи **[Â]∈R[A]** и обозначаются **T = qR(K[Â]).**

II. В результирующее отношение Q входят только те кортежи, для которых выполняется условие **Q[S] ⊆ qR(K[Â]).**

Операции в реляционной БД

R =

№ группы	ФИО	Ин. язык
М-15	Иванов	англ.
М-15	Иванов	нем.
М-25	Петров	англ.
М-25	Петров	кит.
М-25	Петров	нем.
М-25	Сидоров	нем.
М-35	Васин	нем.

Пример операции «деление»:

S =

Ин. язык
англ.
нем.

Студенты, изучающие одновременно английский и немецкий: $Q = R[A \div B]S =$

№ группы	ФИО
М-15	Иванов
М-25	Петров
~~М-25~~	~~Сидоров~~
~~М-35~~	~~Васин~~

Реляционное исчисление

Реляционное исчисление (РИ) используется для теоретической формулировки запросов и является альтернативой реляционной алгебре.

По сравнению с реляционной алгеброй РИ декларативно (формулируется в виде высказываний)и реализуется в виде предикатов.

Реляционная алгебра – процедуральна, т.е. связана с выполнением последовательности процедур – операций над отношениями.

Реляционное исчисление – декларативно.

- **РИ** – это некоторая формула, которая записывается только с помощью кортежных переменных и двухместных предикатов.

РИ является алгебраической интерпретацией запроса пользователя, переведенного с естественного языка в ППФ.

Реляционное исчисление

Основные компоненты РИ:

1. Кортежные переменные, в качестве которых выступают домены отношений, определенных над множествами кортежей, составляющих данные отношения.

2. Двухместные предикаты вида XΘY, где:

   X – кортежная переменная (атрибут);

   Θ – множество операций сравнения {=, ≠, >, ≥, <, ≤ } ;

   У - либо константа, принадлежащая множеству значений кортежной переменной X (предикат принимает истинное значение);

   - либо константа, не принадлежащая множеству значений кортежной переменной X (предикат принимает ложное значение);

   - либо кортежная переменная – домен.

3. Правильно построенная формула (ППФ).

   ППФ должна состоять из: предикатов, логических операций («и», «или», «не»), кванторов общности и существования.

Реляционное исчисление

Любая ППФ должна удовлетворять следующим **условиям**:

- каждый предикат является ППФ;
- если любой предикат – ППФ, то и его отрицание – также ППФ;
- если R и S - ППФ, то $R \vee S$ – ППФ и $R \wedge S$ – ППФ.

Запросы делятся на **два класса**:

1) простые.
2) сложные.

- **Простыми** называются запросы, сформулированные только к одной таблице (количество условий не ограничено).
- **Сложными** называются такие запросы, реализация которых требует обработки условий для двух и более таблиц. Результатом обработки сложного запроса также является таблица.

Реляционное исчисление

Пример.

Пусть имеются три отношения:

1) *Больница* (№, название, адрес, число мест);
2) *Врач* (№ страховки, ФИО врача, специальность);
3) *Пациент* (№, ФИО пациента, адрес, дата рождения, ФИО врача).

Запросы к БД.

1. *Найти название больницы, число мест в которой равно 35:*

   Больница [название больницы, число мест = 35].

2. *Определить ФИО врачей со специальностью «кардиолог»:*

   Врач [ФИО врача, специальность = кардиолог].

3. *Определить ФИО пациентов, которых лечит врач с номером страховки 13:*

   Пациент [ФИО пациента, ФИО врача = Врач [ФИО врача, № страховки = 13]]

   или

   Врач [ФИО врача, № страховки = 13]$\triangleright\triangleleft$ Пациент[ФИО, ФИО врача]

Функциональные зависимости

- **Согласованное состояние БД -** состояние БД, в котором все существующие значения элементов данных являются достоверными и непротиворечивыми.

- **Функциональная зависимость (ФЗ)** существует, когда один или более доменов отношения R *уникально определяет* один или более доменов этого же отношения.

A □ B

Область отправления □ Область значений.

- ФЗ **A □ B** является **полной**, если **B** зависит *от всех значений* **A**.

Все атрибуты отношения должны быть задействованы во множестве ФЗ данного отношения.

- Достаточно одного появления с одной стороны ФЗ.
- Расположение относительно стрелки неважно.

Функциональные зависимости

***Пример**:*
отношение R(ФИО, год рожд.)
Функциональная зависимость вида:
F_R = { *ФИО* □ *год рожд.* }.
Ограничение: значения поля ФИО не должны повторяться.

ФИО	Год рождения
Иванов А.В.	1960
Зуев М.Н.	1963
Смирнова Л.В.	1960
Яшина А.Н.	1961

- Если **A□B & B□A,** то **A□B – взаимнооднозначное соответствие.**

***Пример**:* отношение «П*редприятие»*(название, расч. счет).

Ограничение: каждое предприятие имеет только один расчетный счет. Тогда ФЗ:

назв. предприятия □ *расч. счет предприятия*
расч. счет предприятия □ *назв. предприятия*.

Следовательно – взаимнооднозначная ФЗ.

Функциональные зависимости

Распространенный случай - **отсутствие ФЗ**.

Пример: Отношение «У*чеба»* (ФИО студента, дисциплина).
Ограничения:
1) студент изучает более одной дисциплины;
2) одна и та же дисциплина изучается более чем одним студентом.
Следовательно - отсутствует ФЗ.

При определении ФЗ возможны следующие ситуации:

1. Наличие ФЗ **A□B**, но отсутствие зависимости **B□A** (**A□B&B□A**).
2. Существование обеих ФЗ: **A□B&B□A**.
3. Отсутствие ФЗ (**A□B&B□A**).

Понятие ФЗ распространяется на случаи с двумя и более атрибутами.

Пусть дано отношение R(A, B, C, D, E). Множество атрибутов **A, B, C имеет ФЗ на атрибут D**, если сочетание значений <a, b, c> соответствует единственному значению d.

$$ABC \square D;\ \langle a, b, c\rangle : d \in D;\ a \in A, b \in B, c \in C.$$

Функциональные зависимости

Пример.

Отношение «*Сессия*» (ФИО студента, дисциплина, дата сдачи, ФИО преподавателя, оценка).

Ограничение: студент не может сдавать более одного экзамена в один день.

ФЗ: Fсессия = { ФИО студента, дисциплина □ дата сдачи;

ФИО студента, дисциплина □ оценка;

ФИО студента, дисциплина □ ФИО преподавателя }

Ключ отношения «*Сессия*» - *ФИО студента* и *дисциплина*.

- **Вероятный ключ** отношения R – это один или несколько атрибутов, которые вместе однозначно определяют значения остальных атрибутов в данном отношении.

Если вероятных ключей в отношении несколько, то один из них назначают первичным ключом.

- **Первичный ключ** отношения R - такое сочетание атрибутов, по значениям которого можно однозначно определить значения остальных атрибутов среди множества вероятных ключей.

Функциональные зависимости

Определение ключевых атрибутов:

Для отношения задаются ограничения (в словесной форме)

↓

Выписывается множество функциональных зависимостей. Достаточно, чтобы каждый из атрибутов отношения появился один раз справа или слева от стрелки.

↓

Составляется ключ отношения (объединяются атрибуты, стоящие с левой стороны от стрелки)

Функциональные зависимости

Аксиомы функциональных зависимостей.

X, Y, Z, W - атрибуты отношения R.

1) **Рефлексивность:**

$$\forall x \in R: X \square X,$$

т.е. атрибут однозначно определяет сам себя.

2) **Расширение:**

если **X □ Y**, то **∀ W ∈ R : X, W □ Y**

3) **Аддитивность**:

если **X □ Y & X □ Z**, то **X □ Y, Z**

4) **Проективность (декомпозиция):**

если **X □ Y, Z**, то **X □ Y & X □ Z**

5) **Транзитивность:**

если **X □ Y & Y □ Z**, то **X □ Z**

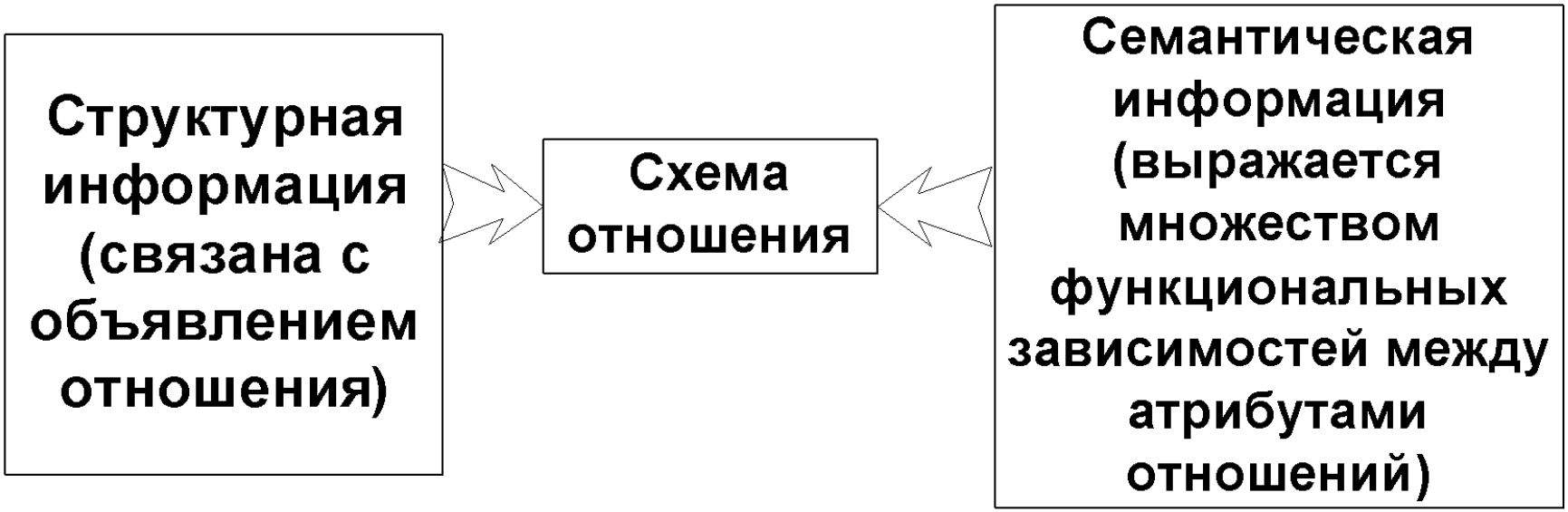
6) **Псевдотранзитивность.**

если **X □ Y & Y □ Z**, то **X, W □ Z**, где **X, Y, Z, W ∈ R**.

Функциональные зависимости

- Два множества **F** и **G** функциональных зависимостей являются **эквивалентными**, если:
  - любая ФЗ **g** ∈ **G** может быть выведена из множества ФЗ **F**;
  - любая ФЗ f ∈ F может быть выведена из множества ФЗ G.
- Когда множества F и G эквивалентны, то говорят, что каждое из них является **покрытием** для другого.

Нормализация отношений



- **Корректной** называется схема базы данных, в которой отсутствуют нежелательные функциональные зависимости.
- **Декомпозиция**(разбиение) – замена исходного отношения отношениями, которые являются проекциями исходного отношения.

  Это и составляет основу процесса нормализации.

Нормализация отношений

Декомпозиция должна сохранять:

- ✔ *обратимость* - возможность получения исходного множества отношений естественным соединением их проекций;
- ✔ *эквивалентность* - при переходе к исходной схеме не должны появляться новые кортежи и не должны отсутствовать кортежи имевшие место в исходном отношении.

- **Нормализация** - замена схемы другими, в которых отношения имеют более простую и регулярную структуру.

Первая нормальная форма (1 НФ).

- Отношение R находится в **первой нормальной форме**, если значения всех доменов атомарны для любого атрибута **Ai∈R** - отсутствует многозначность атрибутов. Значение домена не атомарно, если в приложении оно используется по частям.

Нормализация отношений

Вторая нормальная форма (2 НФ).

- Отношение находится во **второй нормальной форме**, если оно находится в первой нормальной форме и каждый непервичный (неключевой) атрибут полностью зависит от ключа.

Пример: отношение *Поставка* (№ поставщика, товар, цена).

Ограничение: поставщик может поставлять различные товары, и один и тот же товар поставляется различными поставщиками; но каждый товар поставляется по одной и той же цене.

Fпоставка = { № поставщика ⊅ Товар;

Товар ⊅ № поставщика;

№ поставщика, товар → цена;

Товар → цена}

Ключ: № поставщика и товар.

Существует ФЗ, которая включает в себя атрибут, принадлежащий ключу (зависимость от части ключа), - отношение не во 2 НФ.

Нормализация отношений

Вторая нормальная форма (2 НФ).

Последствия зависимости от части ключа (неполная ФЗ атрибута *Цена* от ключа):

1. *Включение*: у поставщика появляется новый товар появляется информация о его цене (если нет - отношение не в 1 НФ).
2. *Удаление*: поставки некоторого товара прекращаются из отношения удаляется информация об этом товаре и его цене.
3. *Обновление (аномалия):* при изменении цены товара необходим полный пересмотр отношения. Иначе - появятся кортежи, которые будут содержать недостоверную информацию о цене.

Разложение отношения *Поставка* на два отношения: R1(№ поставщика, товар), R2(товар, цена) устраняет неполную функциональную зависимость от ключа.

Разложение избавляет от аномалии обновления и избыточности данных и позволяет восстановить исходное отношение.

Нормализация отношений

Третья нормальная форма (3 НФ).

- Схема отношения R находится в **третьей нормальной форме**, если она находится во второй нормальной форме и отсутствует транзитивная зависимость непервичного атрибута от ключа.

Пример: отношение *График* (№ рейса, дата, код пилота, ФИО пилота).

F = { код пилота □ ФИО пилота;

ФИО пилота □ код пилота;

№ рейса, дата □ код пилота}

Первичный ключ: *№ рейса* и *дата*.

Нормализация отношений

Третья нормальная форма (3 НФ).

Операции включения, удаления, и обновления.

1. *Включение.* При появлении нового *рейса* – появляется информация о: *дате* рейса, *коде пилота*, *ФИО пилота*.
2. *Удаление.* При удалении рейса - удаляется информация о: *№ рейса*, *дате*, *коде* и *ФИО пилота*.
3. *Обновление (аномалия).* При изменении *кода пилота* необходимо просмотреть все отношение. В противном случае, отношение - несогласованно.

Проблема - транзитивная зависимость *ФИО пилота* через *код пилота* от ключа. Эта зависимость – нежелательная (аномалия обновления).

Для приведения отношения R к третьей нормальной форме нужно разбить его на: R1(№ рейса, дата, код пилота) и R2(код пилота, ФИО пилота).

Нормализация отношений

Алгоритм декомпозиции в 3 НФ.

1. Получить исходное множество ФЗ для атрибутов БД. Возможны сочетания атрибутов по два, три и т.д. Доказывается или опровергается каждая ФЗ.
2. С помощью шести аксиом о ФЗ сокращается исходное множество ФЗ, полученное на шаге 1. Прекращается, когда сочетания атрибутов содержат первичный ключ.
3. Получить минимальное покрытие множества ФЗ. В нем должны отсутствовать ФЗ, которые являются следствием оставшихся после шага 2 объединяются в одну ФЗ с одинаковой левой частью: **Fmin = { f1, f2, …, fk, …, fn }**
4. Для каждой ФЗ шага 3 создать проекцию исходного отношения: **Rk[fk],** где **k = 1…n, fk -** объединение атрибутов, принадлежащих левой и правой частям **fk**.
5. Если первичный ключ не вошел ни в одну проекцию, то создается отношение **R*,** которое содержит только первичный ключ.

Нормализация отношений

Алгоритм декомпозиции в 3 НФ.

Свойства алгоритма декомпозиции:

1) сохраняет все функциональные зависимости исходного отношения;
2) обеспечивает соединение без потерь;
3) обеспечивает уменьшение избыточности.

Рекомендации по применению:

- *Учитывать взаимооднозначные соответствия:* определить "старший" атрибут, который потом представлял бы все атрибуты взаимооднозначных соответствий.
- *Сокращать объем переборов различных вариантов на шаге 1:* не рассматривать ФЗ, которые являются следствием аксиом 1-6 и отсутствующие ФЗ.

Соединение без потерь с сохранением функциональных зависимостей

Признаки потери:

- потеря некоторых функциональных зависимостей после декомпозиции;
- невозможно восстановить исходное множество функциональных зависимостей;
- можно получать корректную схему с сохранением всех функциональных зависимостей.

В схеме отношения R имеется множество функциональных зависимостей.

Схема R **разложима без потерь** на подсхемы R1, R2, … Rn с сохранением функциональных зависимостей, если для любого r[R], исходная схема может быть восстановлена соединением ее проекций.

$$R = r[R1] \bowtie r[R2] \bowtie \dots r[Rn]$$

Разложение R на R1 и R2 - разложение без потерь, когда:

1. R1 ∩ R2 □ R1\R2

2. R1 ∩ R2 □ R2\R1

Соединение без потерь с сохранением функциональных зависимостей

Пример.

Отношение: «*Служащий*» (№ служащего, Отдел, Город). *ФЗ*:

Fслужащий = {№ служащего → Отдел; № служащего → Город}.

Варианты декомпозиции:

I. R1 (№ служащего, Отдел); R2 (№ служащего, Город).
R1 ∩ R2 = (№ служащего); R1 \ R2 = (Отдел); R2 \ R1 = (Город).

№ служащего → Отдел, № служащего → Город.

Это декомпозиция без потерь с сохранением функциональных зависимостей.

II. R'1 (№ служащего, Отдел); R'2 (Отдел, Город).
R'1 ∩ R'2 = (Отдел); R'1 \ R'2 = (№ служащего); R2 \ R1 = (Город).

Отдел → № служащего, Отдел → Город.

Это – декомпозиция с потерями, поскольку появилась новая функциональная зависимость (Отдел → Город).

Соединение без потерь с сохранением ФЗ

Метод табло.

Применяется, если исходная схема разлагается более чем на две подсхемы.

Дано отношение R и множество функциональных зависимостей.

Строится таблица, в которой:

- строки таблицы - подсхемы отношения R;
- столбцы - список атрибутов исходной схемы без повторений.
- содержание - символы aj, если элемент строки i в столбце j соответствует атрибуту aj подсхемы Ri. В противном случае - bij.

Заполнение таблицы:

1) просматриваются все функциональные зависимости (X → Y);
2) если для атрибутов из X найдутся строки, где в соответствующих местах стоят aj, то bij этих строк, соответствующие столбцам атрибутов из Y, заменяются на aj.

Результат:

- если хотя бы одна строка полностью заполнена aij, то полученное разложение - **разложение без потерь** с сохранением ФЗ;
- в противном случае **разложение с потерями**.

Соединение без потерь с сохранением ФЗ

Метод табло.

Пример.

R (A, B, C, D);

F_R = {A → C, B → C, CD → B, C → D};

R_1 (A, B), R_2 (B, D), R_3 (A, B, C), R_4 (B, C, D) - подсхемы R.

	A	B	C	D
R_1	a_1	a_2	b_{13} → a_3	b_{14} →a_4
R_2	b_{21}	a_2	b_{23} → a_3	a_4
R_3	a_1	a_2	a_3	b_{34} → a_4
R_4	b_{41}	a_2	a_3	a_4

- **AC:** b_{13} → a_3;
- **BC:** b_{23} → a_3;
- **DB:**
- **CD:** b_{34} → a_4;
  b_{14} → a_4;

Разложение - без потерь (R1 и R3 – заполнены а).

Методы хранения и доступа к данным

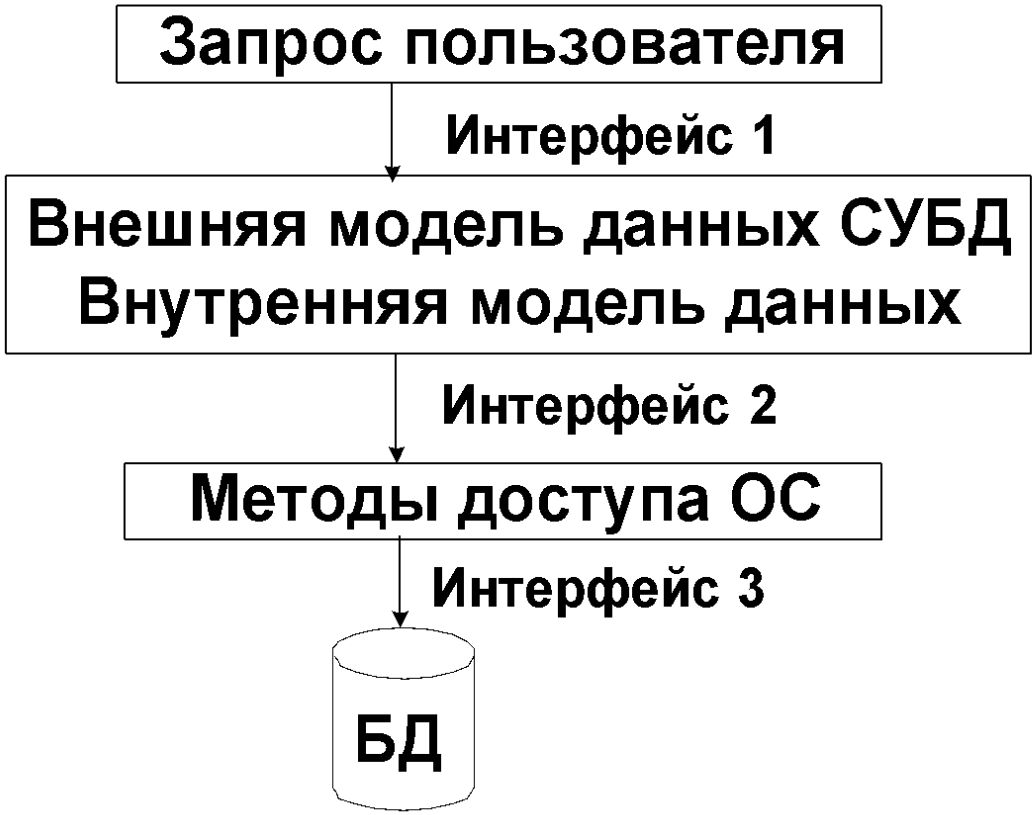
1. Последовательный доступ.
2. Прямой.
3. Индексно-последовательный.
4. Индексно-произвольный.
5. Хеширование (методы вычисления адреса).
6. Инвертирование.

Параметры сравнения методов доступа:

- **Эффективность доступа** - величина, обратная количеству обращений к внешней памяти, требуемых для доступа к конкретной записи.
- **Эффективность хранения** - величина, обратная количеству байт вторичной памяти, требуемых для размещения одного байта исходных данных.

Методы хранения и доступа к данным

Три уровня интерфейса.



- *На первом уровне* - согласование представления запроса пользователя с внешней моделью описания данных в СУБД (происходит внутри БД).
- *На втором уровне* - согласование внутренней физической модели данных СУБД с методами доступа операционной системы ЭВМ (запрос становится понятен машине).
- *На третьем уровне* - обращение операционной системы к физическим данным.

Методы хранения и доступа к данным

Последовательный метод доступа.

Поиск осуществляется в основном файле - просмотр всех записей и выполнение сравнения поискового признака и соответствующего значения атрибута.

Особенности:

- не требует создания дополнительных файлов;
- самая высокая трудоемкость.

Прямой метод доступа.

Строится таблица в которой записывается:

значение действительного ключа + физический адрес записи.

Существует взаимно-однозначное соответствие между ключом записи и ее физическим адресом.

Достоинства:

+ высокое быстродействие;
+ возможность добавлять и удалять записи;
+ эффективность хранения зависит от распределения памяти.

Методы хранения и доступа к данным

Индексно-последовательный метод доступа.

Последовательный поиск данных осуществляется:
1) в поле (теле) индекса (специально организованный файл);
2) среди физических блоков записей.
Записи упорядочены по возрастанию первичного ключа.

Пример.

Существует исходный файл:

ФИО	доп. информация
Волков С.Н.	…
Зайцев А.А.	…
Иванов М.В.	…

Записи объединяются в блоки. Индекс - наибольшее значение ключа в блоке, адрес - физический адрес.

Методы хранения и доступа к данным

Индексно-последовательный метод доступа.

Индексный файл:

Индекс	Адрес (ссылка)
Зайцев А.А.	**блок 1**
Петров Д.С.	**блок 2**
Сидоров В.А.	**блок 3**

Данный метод не совсем хорош с точки зрения добавления и удаления записей.

блок 1
Волков С.Н.
Зайцев А.А.

блок 2
Иванов М.В.
Петров Д.С.

блок 3
Сидоров В.А.

Методы хранения и доступа к данным

Индексно-произвольный метод доступа.

Размещение записи в индексе носит произвольный характер. Существует программа рандомизации - по значению исходного ключа определяет номер блока, в котором размещена запись.

В адресе может находиться:

- адрес блока;
- место записи в блоке.

Данный метод позволяет:

- ✔ осуществлять более быстрый поиск;
- ✔ удобнее удалять и добавлять записи.

Для него требуется больше памяти.

Методы хранения и доступа к данным

Индексно-произвольный метод доступа.

Пример.

Индексный файл:

Индекс	Адрес (ссылка)
Петров Д.С.	**блок 1**
Зайцев А.А.	**блок 2**
Иванов М.В.	**блок 3**
Сидоров В.А.	**блок 1**
Волков С.Н.	**блок 2**

блок 1
Петров Д.С.
Сидоров В.А.

блок 2
Зайцев А.А.
Волков С.Н.

блок 3
Иванов М.В.

Методы хранения и доступа к данным

Инвертирование данных.

- **Инвертирование** - поиск по ключевым полям.

По каждому ключевому полю строится инвертированный файл.

Ключ - то поле, по которому он строится (количество записей в файле = количеству неповторяющихся значений данного поля записи - мощность домена).

Пример.

Нужно определить, кто из студентов конкретной группы проживает в конкретном общежитии.

ФИО	группа	адрес (общежитие)
Волков С.Н.	М-18	03
Зайцев А.А.	М-38	04
Иванов М.В.	М-18	04
Петров Д.С.	М-28	03
Сидоров В.А.	М-28	03

Методы хранения и доступа к данным

Инвертирование данных.

Пример.

Инвертированный файл по полю *Группа*:

Индекс	***Адрес***	***Значения ключей:***
М-18		**Волков С.Н., Иванов М.В.**
М-28		**Петров Д.С., Сидоров В.А.**
М-38		**Зайцев А.А.**

Файл может быть инвертирован полностью и частично в зависимости от того, по всем ли не ключевым полям строятся инвертированные файлы.

Методы хранения и доступа к данным

Инвертирование данных.

Пример.

Инвертированный файл по полю *Адрес*:

Индекс	**Адрес**		**Значения ключей:**
03			Волков С.Н., Петров Д.С., Сидоров В.А.
04			Иванов М.В., Зайцев А.А.

Методы хранения и доступа к данным

Хеширование.

Множеству возможных значений исходного ключа ставится в соответствие один конкретный адрес в памяти машины, по которому должны размещаться записи с этим значением ключа.

Пример.

Возможные значения ключа: от 01 до 99.

№ студ. билета	ФИО
18	Иванов М.В.
44	Петров Д.С.
15	Зайцев А.А.
82	Сидоров В.А.
38	Волков С.Н.

Методы хранения и доступа к данным

Линейное хеширование.

1. Весь диапазон значений исходного ключа разбивается на ряд непересекающихся подмножеств.
2. Каждому подмножеству соответствует одно преобразованное значение ключа и адрес в памяти.

Пример:

Основная область			Область переполнения		
Адрес	Запись	Ссылка	Адрес	Запись	Ссылка
101	18	1001	1001	15	
102	44	1002	1002	38	
103			1003		
104	82		1004		

01 – 30 □ 1
31 - 50 □ 2
51 - 75 □ 3
76 - 99 □ 4

Соответствия:
1 □ 101,
2 □ 102,
3 □ 103,
4 □ 104.

Специальные вопросы проектирования БД

- **Реорганизация БД** - изменение концептуальной, логической или физической структуры БД.
- **Реструктурирование -** изменение концептуальной или логической структуры.
- **Реформатирование** - изменение физической структуры.

Уровни абстракции при проектировании БД.

1. *Концептуальный.* Реорганизация при изменении информационных требований. Итог: изменение сущностей атрибутов или взаимосвязей (элементов данных и отношений 1:1; 1:M; M:M; M:1).
2. *Реализации.* Реорганизация - изменение записей, типов элементов и связей между записями.
3. *Физический.* Реорганизация - изменения в форматах или содержании хранимых записей, в методах доступа, блокировки, сжатия и т.д.

Специальные вопросы проектирования БД

***Примеры реорганизации*:**

- *добавление* нового типа элемента в структуру записи;
- *изменение* связей между двумя или более типами записей;
- *создание* новой базы данных на основе уже имеющихся БД.

Стратегии реорганизации БД.

1. Реорганизация на месте.
2. Реорганизация путем загрузки и перезагрузки.
3. Реорганизация приращениями.
4. Реорганизация параллельно с эксплуатацией.

При использовании первых двух стратегий БД выводится в автономный режим и становится недоступной для пользователя на некоторое время.

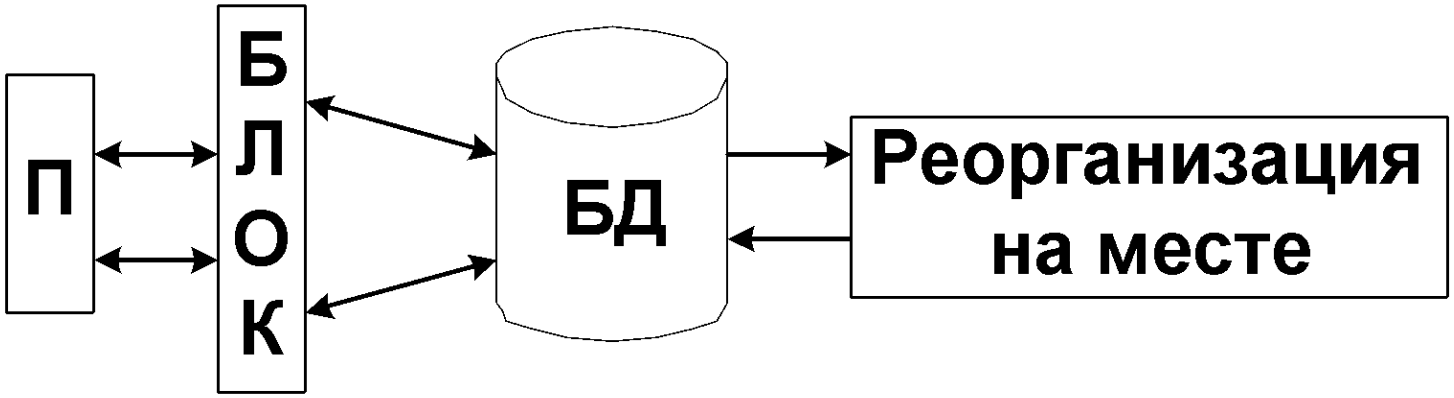
Специальные вопросы проектирования БД

Стратегии реорганизации БД.

1. Реорганизация на месте.

Этапы:

I. Блокировка доступа к БД.
II. Реорганизация БД.
III. Разблокировка доступа и продолжение работы.



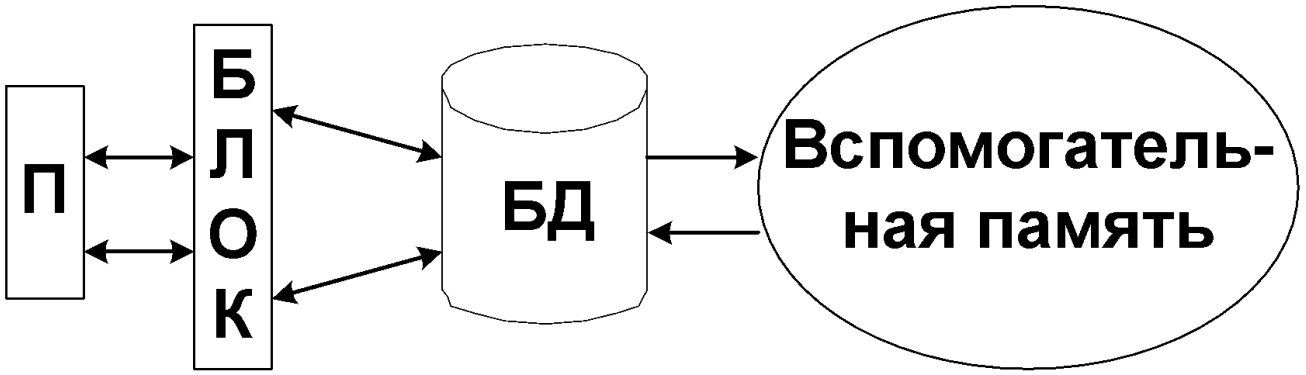
Специальные вопросы проектирования БД

Стратегии реорганизации БД.

2. Реорганизация путем разгрузки и перезагрузки.

Этапы:

I. Блокировка доступа к БД.
II. Вся БД и программное обеспечение переписывается во вспомогательную память.
III. Реорганизация во вспомогательной памяти.
IV. Все из вспомогательной памяти переносится в основную.
V. Разблокировка доступа.



Специальные вопросы проектирования БД

Стратегии реорганизации БД.

3. Реорганизация приращениями.

- ✔ БД не переводится в автономное состояние;
- ✔ реорганизация по мере ссылок на элементы данных.

4. Реорганизация параллельно с эксплуатацией.

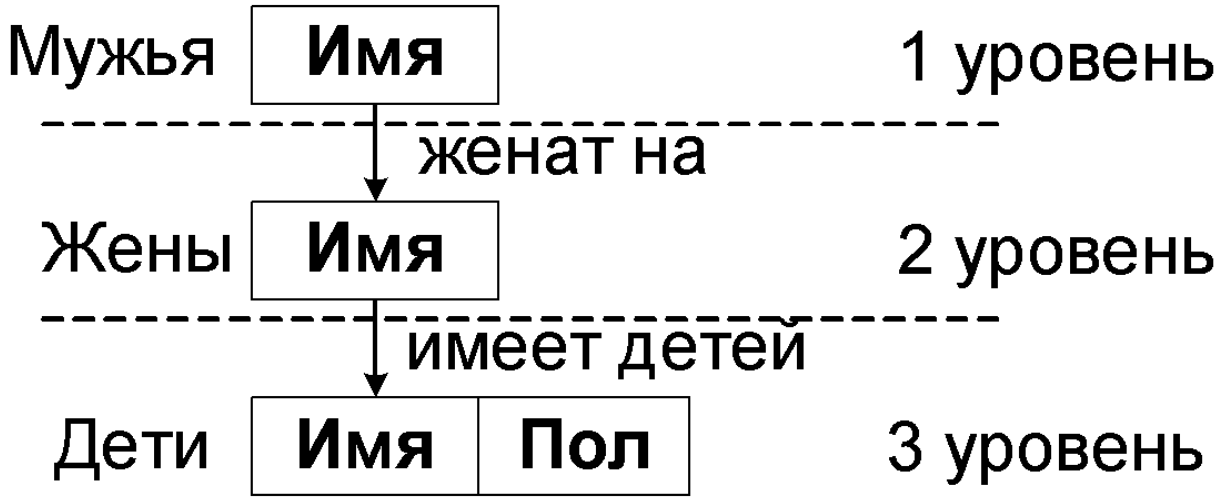
- ✔ БД не переводится в автономное состояние;
- ✔ ограничение прав доступа к реорганизуемым в данный момент данным;
- ✔ реорганизация параллельно с эксплуатацией.

Специальные вопросы проектирования БД

- **Реструктурирование** - преобразование логической структуры БД, связанное с новыми информационными требованиями или требованиями обработки данных.

Пример (для иерархических структур).

Линейная структура:



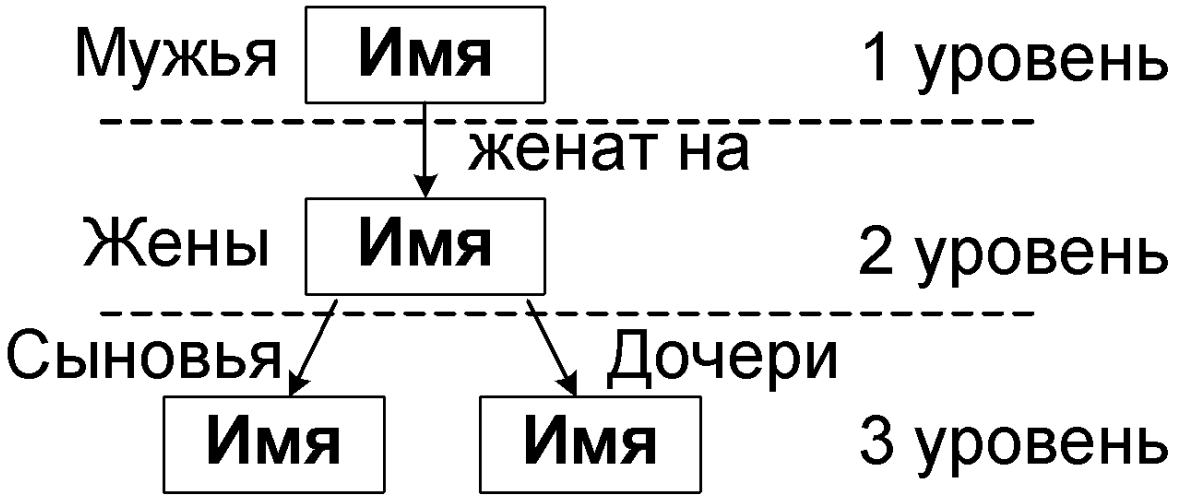
Специальные вопросы проектирования БД

Реструктурирование.

- **Расчленение** - это операция, при которой один тип записей разделяется на два или более типов на основании значения одного или более элементов данных.

Пример (для иерархических структур).

Расчленение типа записи ДЕТИ на СЫНОВЬЯ и ДОЧЕРИ, на основании элемента данных *Пол*.



Специальные вопросы проектирования БД

- **Соединение** - операция обратная расчленению. Она состоит в объединении экземпляров записей в единственный тип записей. В тип записи добавляется элемент данных.(сыновья и дочери - ДЕТИ).
- **Сжатие** двух или более уровней иерархии в один. Некоторые атрибуты записей опускаются.

Пример.

- **Расширение** - операция обратная сжатию. Объекты более подробно описываются с помощью дополнительных объектов на последующих уровнях.

Защита данных

- **Защита данных** - предупреждение несанкционированного или случайного доступа к данным, их изменения или разрушения.

СУБД поддерживает некоторые специальные виды защиты.

Основные группы людей, которые работают с БД:

- пользователи (90% вреда БД);
- программисты (2% вреда БД);
- эксплуатационники (8% вреда БД).

Для защиты данных от пользователя, должны быть выполнены:

- ✔ *идентификация* (сообщение пользователем системе, кто он такой);
- ✔ *проверка* (проверка введенного пароля);
- ✔ *санкционирование* (разрешение выполнять только установленные действия).

Защита данных

Для упорядочения работы пользователя администратор БД составляет **таблицу пользователей**, в которой:

- столбцы – файлы;
- строки – пользователи;
- пересечение строк и столбцов – режимы работы, которые могут осуществляться пользователями с этими данными.

Пол-ли\Данные	F_1	F_2	. . .	F_n
Пользователь 1	чтение	запрет	. . .	запрет
Пользователь 2	обновление	чтение		чтение

Если пользователь превышает предоставленные при идентификации полномочия, то СУБД его отключает.

Защита данных

Программисты увеличивают % нарушений и ошибок.

Программисты делятся на:

- *прикладные программисты* - разрабатывают прикладные программы, имеют вход в программы на уровне пользователя;
- *системные программисты* - осуществляют работу только с системными программами и не имеют доступа к ППП;
- *операторы* - могут выполнять, но не разрабатывать программы.

Целостность данных

Управление целостностью является важной функцией СУБД и обеспечивает поддержку БД в согласованном состоянии в любой момент времени.

Вопросы целостности актуальны:

- в многопользовательском режиме с параллельным манипулированием данными;
- в однопользовательском режиме - каждое обновление БД должно удовлетворять ее структурным и семантическим ограничениям.

Целостность данных

Неявные определяются выбранной моделью данных.

Примеры:

- в иерархической модели – отсутствие связей между узлами одного уровня;
- в реляционной модели – отсутствие одинаковых кортежей или атрибутов.

Явные объявляются пользователем и подобны *переключающим процедурам*, которые связывают изменения в одном экземпляре записи с другими экземплярами связанных записей.

Пример: удаление записи в объекте «Кадры» влечет за собой удаление соответствующей записи по уволившемуся в объекте «Зарплата» Управление целостностью является важной функцией СУБД и обеспечивает поддержку БД в согласованном состоянии в любой момент времени.

Целостность объектов и связей

Характеризуют объекты и их свойства

Характеризуют связи между объектами

Пример.

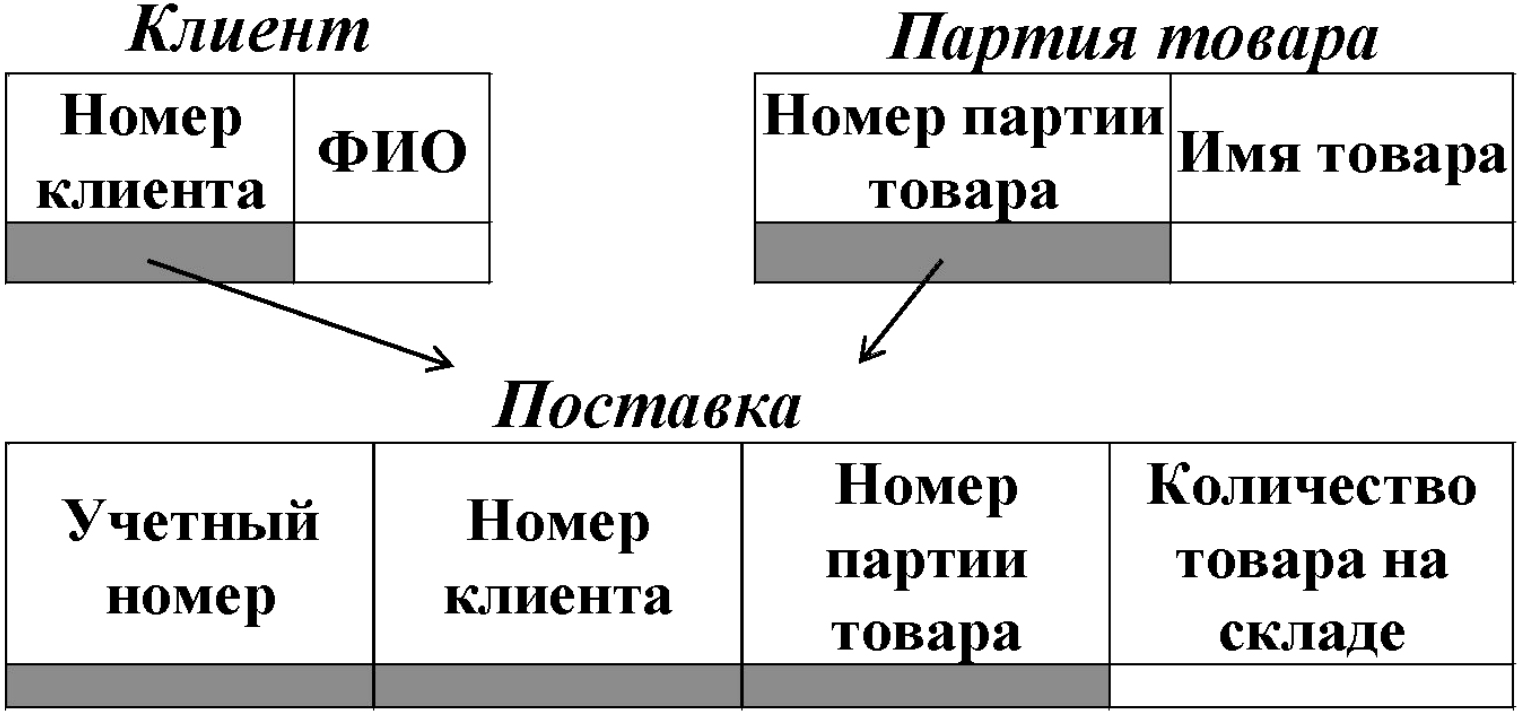
Рассмотрим 3 отношения:

1) *«Клиент»* (№ клиента, ФИО);
2) *«Партия товара»* (№ партии товара, наименование товара);
3) *«Поставка»* (учетный №, № клиента, № партии товара, объем поставки).

Целостность объектов и связей

Пример.

Отношения «*Клиент*» и «*Партия товара*» выражают объекты, а отношение «*Поставка*» выражает связь между этими двумя объектами.



Целостность объектов и связей

Пример. Ограничения целостности:

1. По атрибутам «№ клиента» и «№ партии товара» в отношениях «*Клиент*» и «*Партия товара*» можно однозначно определить ФИО клиента и наименование товара соответственно. Значения этих атрибутов не должны повторяться или принимать пустое значение. Данные атрибуты являются ключами соответствующих отношений.

«№ клиента» $\longrightarrow$ *«ФИО клиента»*

«№ партии товара» $\longrightarrow$ *«Наименование товара»*

2. В отношении «*Поставка*» ключом является объединение трех атрибутов: «Учетный №», «№ клиента» и «№ партии товара». Задав значения этих атрибутов, можно однозначно определить значение атрибута «Объем поставки».

«Учетный №», «№ клиента» $\longrightarrow$ *«Объем поставки»*
и «№ партии товара»

Целостность объектов и связей

Пример. **Ограничения целостности**:

3. Значения атрибутов «№ клиента» и «№ партии товара», указанные в отношении «*Поставка*», обязательно должны содержаться в отношениях «*Клиент*» и «*Партия товара*». В противном случае невозможно будет получить сведения о клиенте и наименовании товара.

4. В отношении «*Поставка*» может быть указанна только часть из номеров клиентов и партий товара, содержащихся в отношениях «*Клиент*» и «*Партия товара*» соответственно. Эти сведения фиксируются в отношении «*Поставка*» только в случае выполнения поставки определенного товара определенному клиенту.

Целостность объектов и связей

Пример. **Ограничения целостности**:

5. Добавление в отношение «*Клиент*» информации о новом клиенте осуществляется независимо от отношения «*Поставка*». Аналогично добавление кортежа в отношение «*Партия товара*» не влечет изменений в отношении «*Поставка*».

6. При добавлении в отношение «*Поставка*» нового кортежа необходимо проверить наличие указанных номеров клиента и партии товара в отношениях «*Клиент*» и «*Партия товара*» соответственно.

7. Удаление кортежа из отношения «Поставка» осуществляется без проверок в отношениях «*Клиент*» и «*Партия товара*».

Целостность объектов и связей

Можно установить **два типа ограничений целостности:**

1) ограничение целостности объектов;
2) ограничение целостности связей.

- **Ограничения целостности объектов** – это ограничения, записанные словесно и реализованные с помощью программ, осуществляющих проверку ограничений для конкретного объекта, не проверяя значений атрибутов у других.

- **Ограничение целостности связей** – это ограничения, которые требуют выполнения условий, связанных с проверкой значений атрибутов других объектов *(переключающие процедуры)*.

Для последнего ***примера*** из приведенных семи ограничений:

- целостность объектов обеспечивают 1, 2, 5, 7.
- целостность связей обеспечивают 3, 4, 6.

Целостность объектов и связей

Пример.

В отношении *«Клиент»* ключом является атрибут «№ клиента», а в отношении *«Партия товара»* - атрибут «№ партии товара».

В отношении *«Поставка»* ключ состоит из трех атрибутов:

- один атрибут из отношения *«Клиент»* - «№ клиента»;
- один атрибут из отношения *«Партия товара»* - «№ партии товара»;
- один собственный атрибут – «Учетный номер».

- ***Внешний ключ*** - ключевой атрибут некоторого отношения, значения которого являются значениями ключа другого отношения.

Целостность приложений

Ограничения целостности объектов и связей - свойства реляционной модели.

Ограничения целостности приложений связаны с конкретными приложениями и реальными данными.

Данные ограничения либо задаются пользователем, либо заложены в СУБД.

Наличие ограничений целостности приложений позволяет утверждать, что состояние БД в любой момент времени является целостным и удовлетворяет потребностям пользователя.

Определение ограничений целостности осуществляется посредством **утверждений**. Утверждения позволяют определить такие совокупности данных, которые нарушают целостность БД.

Более важное место занимает ограничения целостности самих данных, чем способов их расчета.

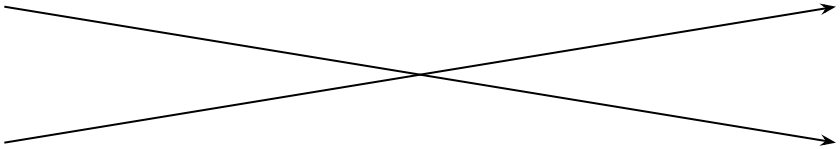
Целостность приложений

Пример.

Представлены два отношения:

«Служащий»

№ служащего	ФИО служащего	№ принадлежности	Пол	Оклад



«Отдел»

№ отдела	Наименование отдела	№ руководителя	Кол-во служащих	Годовая смета

Целостность приложений

Перечислим **ограничения целостности:**

Целостность объектов:

1.1 № служащего является ключом отношения *Служащий.*

1.2 № отдела является ключом отношения *Отдел.*

Целостность связей:

2.1 Значение атрибута «№ отдела» отношения *Отдел* должно обязательно встречаться среди значений атрибута «№ принадлежности» в отношении *Служащий.*

2.2 Значение атрибута «№ руководителя» отношения *Отдел* должно встречаться среди значений атрибута «№ служащего» в отношении *Служащий.*

Целостность приложений

Целостность приложений:

3.1 Атрибут «Пол» может принимать только значения "мужчина" или "женщина".

3.2 Размер оклада служащих не должен превышать 30 000 руб.

3.3 При обновлении значения атрибута «Оклад» новое значение не должно быть меньше прежнего.

3.4 При обновлении значения атрибута «Оклад» новое значение суммы окладов для каждого отдела не должно превышать прежнее более чем на 30%.

3.5 Средний оклад женщин должен составлять 80-120% среднего оклада мужчин.

3.6 Значение атрибута «Количество служащих» отношения *Отдел* должно быть равно количеству кортежей отношения *Служащий*, принадлежащих рассматриваемому отделу.

3.7 Оклад руководителя отдела не должен быть меньше оклада служащего отдела.

Классификация введенных ограничений целостности:

1. Статические ограничения и ограничения перехода.

- *Статические* ограничения должны выполняться для любого состояния БД и при выполнении любых процедур преобразования (ограничения **3.1 и 3.2**).
- *Ограничения перехода* связаны с процедурой сравнения предыдущего значения атрибута с последующим после редактирования и требуют проверки выполнения заданных условий (ограничения **3.3 и 3.4**).

2. Отложенные и безотлагательные ограничения.

- *Безотлагательными* называют ограничения, допускающие возможность проверки одновременно с изменением значений данных (ограничение **3.4**).
- *Отложенными* называют ограничения, для которых проверка выполнения имеет смысл по завершении необходимых операций (между операциями может наблюдаться нарушение целостности данных) (ограничения **3.6 и 3.7**).

3. Ограничения для кортежей и множеств.

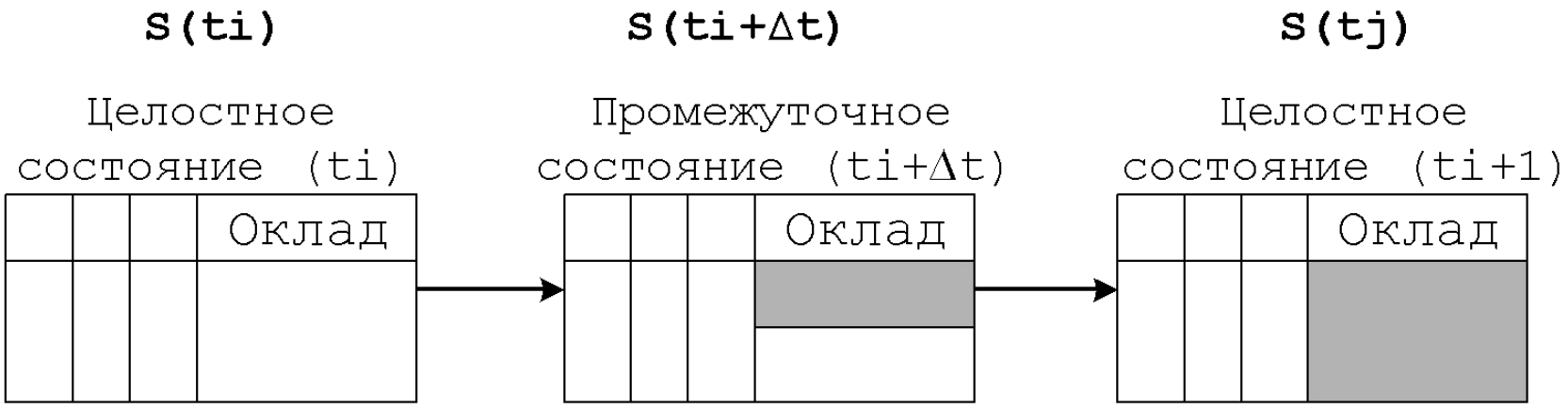
- *Ограничениями для кортежей* называют такие ограничения, проверку выполнения которых можно осуществить, используя отдельные кортежи отношений (ограничения **3.1, 3.2, 3.3**).
- *Ограничения множеств* представляют собой ограничения целостности, относящиеся к итоговым и средним значениям (ограничения **3.4 и 3.5**).

Логический элемент работы

- ***Логический элемент работы*** – непрерывное управление данными, при котором БД из одного целостного состояния переводят в другое целостное состояние.

Пример: операция повышения размера оклада всем служащим на 15%.

Перед началом операции размер оклада соответствует ограничениям целостности. В ходе операции ЭВМ последовательно изменяет размер оклада, поэтому существует состояние, когда оклад будет повышен только части служащих, а у других служащих останется старое значение оклада. Такое состояние - не целостное. После завершения операции, база данных возвращается в целостное состояние. Логический элемент работы – непрерывное управление данными, при котором БД из одного целостного состояния переводят в другое целостное состояние.



Логический элемент работы на примере операций реляционной алгебры

Каждое действие можно рассматривать с трех позиций:

- атрибута;
- кортежа;
- отношения.

1. Операция объединения

Отношения: R1(A, B, C) и R2(A, B, C).
Объединение: R* = R1 U R2.
Новое отношение - R*(A, B, C).
При удалении (добавлении) кортежа из R*(A, B, C) - кортеж должен быть удален (добавлен) только из одного отношения (R1 или R2).

Логический элемент работы на примере операций реляционной алгебры

2. Операция соединения

Отношения: R1(A, B) и R2(B, C).

Соединение по атрибуту B: $R^* = R1 \bowtie R2$.

При удалении (добавлении) кортежа из R* -соответствующие части этого кортежа удаляются (добавляются) из R1 и R2.

3. Операция разности

R* состоит из кортежей принадлежащих R1 и не принадлежащих отношению R2: $R^* = R1 \setminus R2$.

При удалении (добавлении) кортежа из R* - этот кортеж удаляется (добавляется) только из R1.

4. Операция пересечения

Отношение, состоящее из общих кортежей R1 и R2: $R^* = R1 \cap R2$.

При удалении (добавлении) кортежа из R* - кортеж удаляется или добавляется и из R1 и из R2.

Управление данными

Важные факторы, влияющие на работу БД:

- ✔ целостность;
- ✔ стабильность;
- ✔ надежность данных.

Наличие в базе неверных данных усложняет их обработку и может привести к нежелательным последствиям.

- ***Управление данными*** - способы повышения надежности при выполнении операции над данными в БД.

Способы повышения надежности данных.

Аспекты управления данными для повышения надежности:

- ***Управление доступом*** **-** понимается защита данных в БД от несанкционированного доступа.
- ***Управление целостностью*** **-** защита данных в БД от неверных (в отличие от незаконных) изменений и разрушений.

Управление данными

Поддержание целостности - обеспечение правильности базы данных в любой момент времени.

Рассматривается в трех **аспектах:**

- ***Обеспечение достоверности*** - предотвращение возможности появления недопустимых значений данных из-за ошибки в управлении данными.
- ***Управление параллелизмом*** – предотвращение нарушения целостности *БД* при одновременном выполнении нескольких операций.
- ***Восстановление*** - возможность за предельно короткое время восстанавливать состояние БД после появления неисправности программного или аппаратного обеспечения.

Управление параллелизмом

- **Логический элемент работы** - операции обработки, в результате которых БД из некоторого целостного состояния переходит в другое целостное состояние.
- **Управлением параллелизмом** - способы обеспечения целостности БД при одновременном выполнении логических элементов работы.

Одновременное выполнение операций может привести к нежелательным последствиям.

Одновременное (параллельное) выполнение нескольких логических элементов работы - одновременное использование ими базы данных (в ЭВМ внутренние механизмы параллельной работы входных и выходных устройств и центрального процессора).

Логические элементы работы, выполняемые одновременно - *одновременно выполняемые элементы*.

Обнаружение параллельности выполнения логических элементов работы

Анализируется:

- множество входных (считываемых) данных X
- множество выходных данных Y
- логический элемент работы W.

Два логических элемента работы Wi и Wj выполняются параллельно при условиях:

1) **Xi □ Yj = 0**

2) **Yi □ Xj = 0**

3) **Yi □ Yj = 0**

- выходные данные одного логического элемента работы не должны быть затронуты другим логическим элементом работы;
- никакие два логических элемента работы не должны модифицировать общие переменные.

Одновременное выполнение и логический элемент работы

В реальных системах:

- БД размещаются на внешних запоминающих устройствах;
- выполнение операции - передача части данных в рабочую область оперативной памяти ЭВМ;
- обмен данными производится блоками (страницами);
- изменение данных осуществляется в рабочей области.

Если содержимое оперативной памяти вновь переписать на магнитные диски, то - изменение содержимого БД (отображение изменений в БД).

Нахождение рабочей области и БД в различных местах приводит к ошибкам при пересылке к снижению надежности.

Основная проблема - в ходе выполнения одного логического элемента работы, "открывается" процесс в другом, одновременно выполняемом элементе.

Одновременное выполнение и логический элемент работы

Пример: Отношение *Оплата*

№ службы	Имя службы	Принадлежность	Размер оплаты
001	Иванов	Производство	100000
002	Петров	Делопроизводство	90000
003	Колесников	Делопроизводство	130000
004	Бондаренко	Производство	180000
005	Сидорова	Здравоохранение	80000

1 : средняя оплата производственных служащих.

2 : повышение на 20% оплаты всем служащим.

3 : переход служащих с именами Петров и Колесников из делопроизводителей в производственных служащих

Одновременное выполнение и логический элемент работы

При одновременном выполнении над отношением *Оплата* каких-либо логических элементов работы (1, 2, 3), получается не целостный результат.

1 осуществляет просмотр БД, но результат его выполнения изменяется в зависимости от наличия одновременного выполнения элементов 2, 3.

Временная несогласованность БД свойственна обработке данных на ЭВМ, и избежать ее невозможно.

Цель управления параллелизмом - с учетом архитектуры ЭВМ наиболее широко реализовать возможность одновременного выполнения логических элементов работы, обеспечивая при этом целостность БД.

Основные понятия блокировки

- Для одновременного выполнения логических элементов работы без нарушения целостного состояния БД выполняется **блокировка**.

Простая модель

- **Логический элемент работы W0** - последовательность элементарных операций aij, каждая из которых может быть одной из трех:

  1) Изменить объект данных X (aij*xi).

  2) Блокировать Lx.

  3) Разблокировать Ux.

Объектами блокировки могут быть:

- значения атрибутов;
- кортежи;
- отношения.

- Любая операция aij кроме операций блокировки осуществляет изменение объекта.
- Блокировку объекта может снять только один логический элемент работы, который установил эту блокировку.

Основные понятия блокировки

Логический элемент работы Wi - **правильно оформленный,** если:

1) Перед обработкой объекта должна быть выполнена его блокировка.

2) После обработки объект должен быть разблокирован (освобожден).

3) Перед освобождением блокированного объекта не должна выполняться его повторная блокировка.

4) Не разблокированный объект не должен освобождаться.

1) P xai2xUxLyai5yUy; правильно оформленные
2) P xLyai3xaiuyUxUxUy;
3) P xai2xLyaiuy; неправильно оформленные
4) P xai2xLyaiuyUxUyUz.

Реляционная алгебра

Два отношения могут использоваться параллельно, если они не имеют общих атрибутов:

- входные данные одного отношения не модифицируются другим отношением;
- никакие два отношения не модифицируют общие переменные.

Если пара отношений имеет хотя бы один общий атрибут, то для обеспечения целостности при одновременной работе используют блокировку.

Пример. Правильно оформленный логический элемент работы:

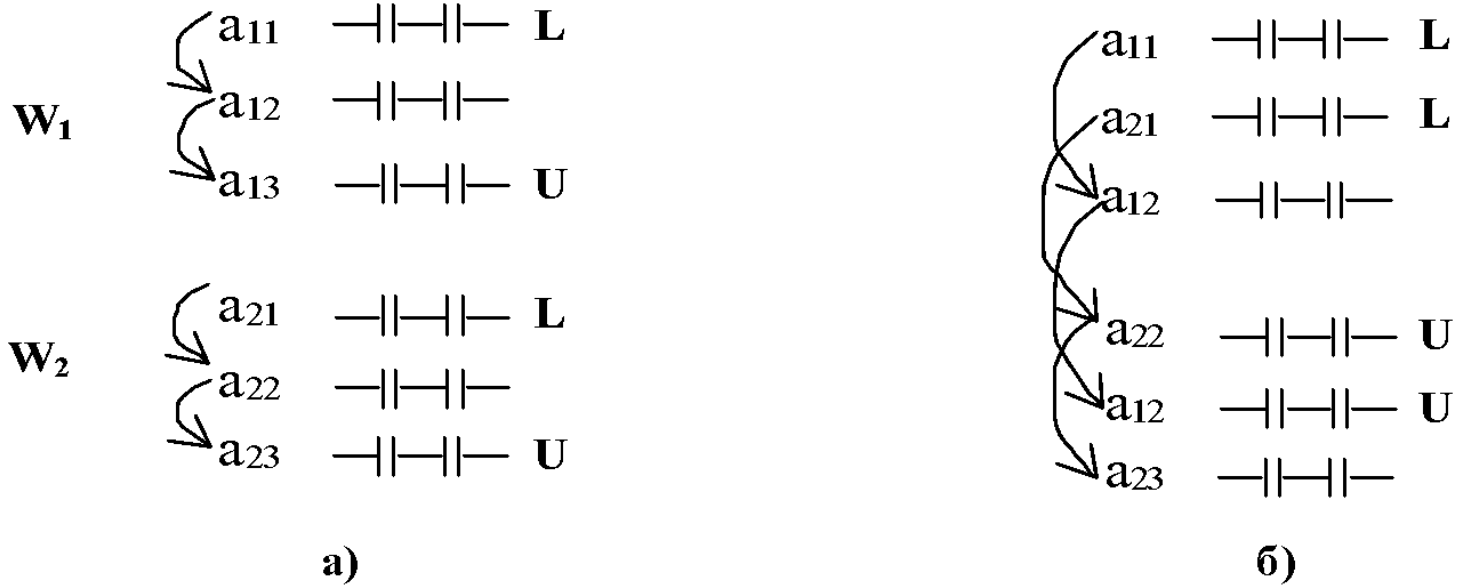
```
W1
   a11 книгу регистрации продажи в
       кредит
   a12 запись продажи и остатка в
       книгу регистрации
   a13 книгу регистрации продажи в
       кредит
```

```
W2
   a21 книгу регистрации продажи в
       кредит
   a22 запись взноса и остатка в
       книгу регистрации
   a23 книгу регистрации продажи в
       кредит
```

Реляционная алгебра

- Порядок выполнения элементарных операций aij, образующих логические элементы работы W1, W2, Wn - **расписание совокупности элементов.**

Расписание для элементов W1 и W2:



а - последовательное расписание.

Для обеспечения целостности БД при выполнении логических элементов работы необходимо соблюдать определенные правила выполнения блокировки (последовательное расписание).

Реляционная алгебра

- ***Последовательное расписание*** - расписание, в котором элементарные операции aij каждого логического элемента работы обязательно являются смежными.
- Расписание, выполнение которого дает результаты, эквивалентные результатам последовательного расписания – ***сериализуемое*** (гарантирует целостность БД).
- Расписание правильно оформленных логических элементов работы, в которых повторная блокировка ранее блокированного объекта происходит после его разблокирования - ***легальное расписание***.
- Совокупность логических элементов работы, для которой все легальные расписания являются сериализуемыми - ***надежная***.

  Если интервалы блокирования всех объектов в логическом элементе работы накладываются друг на друга, то - ограничение двухфазовой блокировки.

  Если логические элементы работы W1, W2, ..., Wn являются правильно оформленными и для них выполняется двухфазная блокировка, то совокупность элементов (W1, W2, ..., Wn) надежная.

Уровни целостности и типы блокировок

Не все операции, выполняемые логическими элементами работы, изменяют значения объектов. Это учитывается при организации блокировки объектов.

1) При одновременном выполнении двух логических элементов работы W1 и W2 могут возникнуть ситуации:

   а) Нарушение целостности - **потеря обновления**. Вызвано наложением изменяемых значений одинаковых объектов.

   б) Возникновение ошибки в назначении объекта, изменяемого элементом W1. Может привести к ошибке выполнения элемента W2(невозможно исключить только результат выполнения W1)- **взаимозависимость восстановления**.

Уровни целостности и типы блокировок

2) Изменение значения объекта логическим элементом работы W1 и считывание значения элементом W2. Ситуации:

а) Изменение логическим элементом работы W1 значения объекта, считанного логическим элементом работы W2. Для W2 отсутствует **воспроизводимость считывания**.

б) Возникновение отказа перед окончанием логического элемента работы W1 - исключает изменения выполнения W2 (возможен возврат в состояние до выполнения W1). Есть возможность считывания W2 неправильных значений объекта - W1 **загрязняет** W2.

3) Чтение объекта логическим элементом работы W1, изменение значений объекта логическим элементом работы W2.

Уровни целостности и типы блокировок

4) Чтение значений объекта логическими элементами работы W1 и W2. Существует несколько уровней целостности. Можно определить вид и период блокировки:

а) При изменении значений объекта - монопольная блокировка - ни какие другие одновременно выполняемые логические элементы работы не могут считывать либо изменять значение объекта.

б) При использовании монопольной блокировки не проводить освобождение объекта до окончания данного логического элемента работы (может привести к потере обновления, взаимозависимости восстановления и др.)

в) При одновременном выполнении логических элементов работы (только выборка значений объектов), используется совместная блокировка - одновременно выполняется считывание значений объекта, но запрещено выполнение этого значения.

г) Проводить освобождение объекта до окончания логического элемента работы не следует.

Уровни целостности и типы блокировок

Использование возможных вариантов блокирования одинаковых объектов при одновременном выполнении логических элементов работы.

Вид первоначальной блокировки	Вид последующей блокировки		
	нет	Совместная блокировка	Монопольная блокировка
нет	0	0	0
Совместная блокировка	0	0	X
Монопольная блокировка	0	X	X

0 – допустима следующая попытка блокировки.

X – не допустима следующая попытка блокировки.