Реляционная модель

Определение
Неформально: РМ - это модель, которая описывает как организовать данные в таблицах и как определить связи между этими таблицами

Аспекты реляционной модели:

- **Структурный аспект** — данные в базе данных представляют собой набор отношений.
- **Аспект целостности** — отношения (таблицы) отвечают определенным условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.
- **Аспект обработки** (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление)

Определение целостности
Виды целостностей (Мб из дока с вопросами, НО!)
Как названия и смысл целостностей из картинки 1 соотносится с названиями и смыслом целостностей из картинки 2 и 3??

Целостность

- Целостность сущностей
  - Ключи
- Ссылочная целостность
  - Только сообщение о нарушении целостности
  - Метод значений NULL
  - Каскадный метод
  - Метод значений по умолчанию

Пример (возможно ли такое состояние таблиц в БД?)
Сотрудник (ТН, ФИО, НомПасп, НомОтдела)
Отдел (НомОтдела, НазОтдела)
Сотрудник.НомОтдела → Отдел.НомОтдела

Аспекты модели

- Структуры данных
- Манипулирование данными
  - Реляционная алгебра
  - Реляционное исчисление
    - на кортежах
    - на доменах
- Целостность данных

7. Целостность, нормальные формы

Целостность сущности – Каждая запись сущности должна обладать уникальным идентификатором и содержать данные. В таблице нет повторяющихся строк

Ссылочная целостность в реляционной базе данных – это согласованность между связанными таблицами. Ссылочная целостность обычно поддерживается путем комбинирования первичного ключа и внешнего ключа. Для соблюдения ссылочной целостности требуется, чтобы любое поле в таблице, объявленное внешним ключом, могло содержать только значения из поля первичного ключа родительской таблицы

Ссылочная целостность реализуется **внешним ключом**

Целостность домена указывает, что все столбцы в реляционной базе данных должны быть объявлены в определенном домене. Все значения некоторого атрибута принадлежат множеству допустимых значений

Структуры данных модели:
Отношение - конечное подмножество декартова произведения доменов D1x D2x … xDn

Атрибут - Неформально: именованный столбец отношения.
Более формально: свойство, которое описывает конкретное отношение

Домен - диапазон значений атрибута. Домен задается базовым типом и логическим выражением, определяющим элементы домена *(Домен ИМЕНА может базироваться на строковом типе, но содержать только те строки, которые НЕ начинаются на Ъ или Ь знак)*

Схема отношения - именованное множество пар формата {имя атрибута, имя домена} или же запись вида R (A1:D1, A2:D2, …,Ak:Dn) или R (A1, A2, …,Ak), где k ≥ n, R – имя отношения, Ai – имена атрибутов, Dj - имена доменов

Степень отношения - число атрибутов в отношении

Кортеж - элемент отношения. Неформально: запись в таблице (Строка в таблице)

Кардинальное число - число кортежей в отношении
8

Тип данных. Тип данных БД полностью адекватен типу данных в языках программирования чееее

Первичный ключ - атрибут или множество атрибутов, однозначно определяющих кортеж в отношении

Внешний ключ - подмножество атрибутов некоторого отношения R2, значения которых должны совпадать со значениями некоторого потенциального ключа некоторого отношения R1

Потенциальный ключ — подмножество атрибутов, минимальное по включению

Вторичный индекс по полю F есть связь между множеством значений F (по сути доменом) и множеством записей рассматриваемого файла

1. Реляционная алгебра

Определение

Реляционная алгебра — замкнутая система операций над отношениями в реляционной модели данных.

Проектирование реляционной базы данных.
!!Что тут писать то!! (Про ER-модель наверное)
Операции РА

Теоретико-множественные операции

- **Объединение, пересечение, разность**. Условия выполнения этих операций: схемы отношений – одинаковы (кол. Атрибутов одинаково, соответствующие атрибуты определены на одних и тех же доменах)
- **Декартово произведение**. Ограничений для выполнения нет

Специальные операции

Селекция
Пусть R(A1,A2, … Ak) – отношение.

Операция $б_f$ (R(A1,A2, … Ak)) порождает отношение R1(A1,A2, … Ak) кортежи которого

- Принадлежат исходному отношению R (A1,A2, … Ak)
- Удовлетворяют логической функции f, которая строится из операндов, атрибутов отношения R, констант, операторов сравнения: >,<,=,≠,≤,≥, логических операторов: and, or, not, скобок: (,)
- Проще говоря, селекция - фильтр по некоторым параметрам

Например R(A, B, C, D) = (1 2 3 4), (1 2 3 5), (1 3 4 8), (1 6 7 5), тогда

$б_{(B<C)and(D<C)}$(R) = R2(A, B, C, D) = (1 6 7 5)

Проекция
Пусть R(A1,A2, … Ak) – отношение.
Операция πAi1, Ai2, … Ais(R) порождает отношение R1 с атрибутами Ai1,Ai2 … Ais из отношения R, где is ≤ k, следующим образом:

- Каждый кортеж в R1 формируется из кортежа отношения R путем удаления не отмеченных в операции πAi1, Ai2, … Ais(R) значений атрибутов
- В получившемся отношении одинаковые кортежи удаляются.

Например, если у нас есть отношение R(A, B, C, D), то A A = A1 , A2 , ..., Ak проекция πAB (R) = R1 (A, B)

Соединения

- **Эквисоединение** $R \bowtie_f S = б_f (R \times S)$

  Пример (Типо суть в том, что мы делаем декартово, потом селекцию по функции которая под знаком бабочки находится. В данном случае оставляем кортежи где значение B из отношения R меньше D из отношения S)

A	B	C
1	2	3
4	5	6
7	8	9

а) Отношение R

D	E
3	1
6	2

б) Отношение S

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

в) Отношение $R \triangleright\triangleleft S$ $B<D$

Рис. 4.4. Пример $<$ — соединения

- **Натуральное соединение R |><| S**, выполняется только для тех отношений, у которых есть общие атрибутые

  $R \bowtie_f S = \pi_{атр\ R \cup атр\ S}(\sigma_f(R \times S))$, где f функция вида $A_{r1}=A_{s1}\&A_{r2}=A_{s2}\& \ldots \&A_{rl}=A_{sl}$,

  $A_{ri}=A_{si}$ – одинаковые атрибуты из R и S

Например, R(A, B, C) = (1 2 3), (3 2 1), (4 1 5); S(C, D) = (3 2), (7 8), (1 6). Общий атрибут, видно, это C, поэтому останется R |><| S = R3(A, B, C, D) = (1 2 3 2), (3 2 1 6)

Еще один пример (Если под знаком бабочки ничего нету, то имеется в виду натуральное соединение)

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

а) Отношение R

B	C	D
b	c	d
b	c	e
a	d	b

б) Отношение S

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

в) Отношение $R \triangleright\triangleleft S$

Еще один пример

Вопрос **18**

Выполнен

Баллов: 1,00 из 1,00

Отметить вопрос

Пусть даны отношения

R (A, B, C) S (B, D)

1 3 4 2 1

2 5 7 3 7

4 2 2

Результатом какой операции является отношение T, если имеет вид

T (A, B, C, D)

1 3 4 7

4 2 2 1

Выберите один ответ:

- LEFT JOIN соединение
- Натуральное соединение
- Деление
- Декартово произведение
- RIGHT JOIN соединение

- **LEFT OUTER JOIN**  , **RIGHT OUTER JOIN**  , **FULL OUTER JOIN** 

Деление

Пусть R(A:цел, B:цел, C:цел), S(A:цел, B:цел), тогда R(A,B,C) ÷ S (A, B) = T(C)
Отношение T есть множество кортежей t длины |R|-|S|, таких что для всех кортежей u длины |S|, принадлежащих S, кортеж tu принадлежит R.

Например, R(A,B,C) = (1,2,3), (2,4,3) (1,2,1),(2,3,4); S(A,B) = (1,2),(2,4)
R(A,B,C) ÷ S (A, B) = T(C) = (3)

Функциональная зависимость:

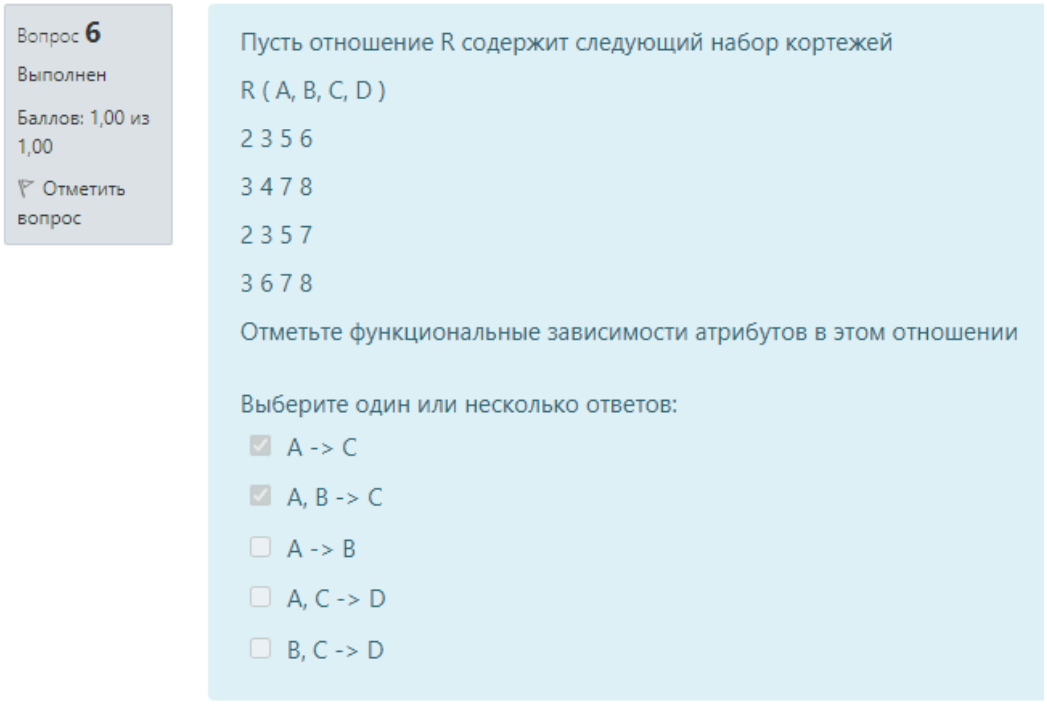
Множество атрибутов Y функционально зависит от X на R (X → Y), если в отношении R не могут содержаться 2 кортежа, компоненты которых:

- Совпадают по всем атрибутам из X,
- но не совпадают по одному или более атрибутам из Y

Функциональные зависимости отражают
Зависимость неключевых атрибутов от КЛЮЧА,
Отношение N:1 разных групп объектов

Полная и частичная функциональная зависимость
Полной функциональной зависимостью называется зависимость неключевого атрибута от всего составного ключа. Частичной функциональной зависимостью будем называть зависимость неключевого атрибута от части составного ключа

Примеры функциональной зависимости



Вопрос **7**

Выполнен

Баллов: 1,0 из 1,0

Отметить вопрос

Пусть отношение R содержит следующий набор кортежей

R (A, B, C, D)

2 3 5 6

3 4 7 8

2 3 5 7

3 6 7 8

Отметьте функциональные зависимости атрибутов в этом отношении

Выберите один или несколько ответов:

- [ ] B, C -> D
- [x] B -> C
- [x] A -> C
- [ ] A, C -> D
- [ ] A -> D
- [x] A, B -> C

2. Реляционное исчисление где t – переменная кортеж некоторой фиксированной длины,

а φ – формула, построенная из атомов и совокупности операторов.

Семантика выражения – множество кортежей t, удовлетворяющих формулу φ

Атомы формулы φ могут быть 3 типов:

1. R (s), где R – имя отношения, а s – переменная-кортеж.
   Этот атом означает, что s есть кортеж в отношении R.

   В программировании – это позиция объявления переменной.
   Отношения: Студенты(н_ст, имя, н_гр), Преподаватели(н_пр, имя, тел),
   Курсы(н_кр, назв, фак, сем), Расписание(дн, пара, н_кр, н_пр, н_гр)
   Объявления переменных: Студенты(с), Преподаватели(п), Курсы(к), Расписание(р),
2. s[*i*] θ u[*j*], где s и u являются переменными-кортежами,
   а θ – арифметический оператор сравнения (<, >, = и т.д.) Этот атом означает,
   что *i –ый* компонент s находится в отношении θ с *j –ым* компонентом u.
   Примеры: с[3] = р[5], к[1] = р[3]
3. s[*i*] θ α, α θ s[*i*] , где s является переменной-кортежем, а α – константа
   Примеры: к[2] = ‘Базы данных’

- В выражениях РИК переменные-кортежи могут быть свободными или связанными.
- Переменная связана, если в формуле ей предшествует квантор ∃ или ∀.
- Понятие свободной переменной аналогично понятию глобальной переменной в программировании, понятие связанной переменной – понятию локальной переменной

Формулы

1. Каждый атом есть формула. Все вхождения переменных-кортежей в атоме являются свободными в этой формуле.
2. Если φ_1 и φ_2 – формулы, то $\varphi_1 \& \varphi_2$, $\varphi_1 \vee \varphi_2$, $\neg \varphi_1$ – также формулы. Переменные-кортежи являются свободными или связанными в построенных формулах точно так же какими он были в исходных формулах.
3. Если φ – формула, то $\exists s (\varphi)$ также формула. Она истинна, если в области определения переменной s найдется хотя бы один кортеж, при котором формула φ примет значение 'истина'
4. Если φ – формула, то $\forall s (\varphi)$ также формула. Она истинна, если для всех кортежей из области определения переменной s формула φ принимает значение 'истина'
5. В формулах могут быть скобки
6. Ничто иное не является формулой

Примеры выражений РИК для отношений РА:

Если { t | R (t) } – множество кортежей отношения R,
а { t | S (t) } – множество кортежей отношения S, то

Объединение R ∪ S выражается как
{ t | R (t) ∨ S (t) }

Разность R\ S выражается как
{ t | R (t) & ¬ S (t) }

Реляционное исчисление на доменах
Выражения реляционного исчисления на доменах имеют вид:

{ x1, x2, … , xk | φ (x1, x2, … , xk) }

а φ – формула, построенная из атомов и совокупности операторов, а x1,

x2, … , xk – переменные, определенные на доменах отношений, входящих в формулу

3. Связь реляционной алгебры и реляционного исчисления

Теорема

Если E – выражение реляционной алгебры, то существует эквивалентное ему безопасное выражение в реляционном исчислении с переменными-кортежами.

Идея доказательства

Индукция по числу вхождений операторов в E.

Безопасные формулы РИК (Помогите, что это вообще и зачем??????) (это чтобы за конечное время можно было определить истинность)

Выражение РИК $\{ t \mid \varphi(t) \}$ называется безопасным если удовлетворяются следующие условия:

1. Всякий раз, когда t удовлетворяет φ, каждый компонент t есть элемент $DOM(\varphi)$
2. Для любого подвыражения φ вида $\exists u(\omega(u))$ каждый компонент u принадлежит $DOM(\omega)$, если u удовлетворяет ω
3. Для любого подвыражения φ вида $\forall u(\omega(u))$ каждый компонент u принадлежит $DOM(\omega)$, если u удовлетворяет $\neg\omega$

Условия 2,3 позволяют устанавливать истинность формул вида $\exists u(\omega(u))$ или $\forall u(\omega(u))$, рассматривая только u, составленные из принадлежащих $DOM(\omega)$ символов

Примеры

1. Любая формула $\exists\, u(R(u) \vee \ldots)$ удовлетворяет условию 2
2. Любая формула $\forall u\, (\neg R(u) \vee \ldots)$ удовлетворяет условию 3

4. Транзакции

Будем предполагать, что действия над базой данных выполняются с помощью операторов SQL. **(А каким образом еще можно действия над бд выполнять кроме SQL???)**

Определение
Транзакция – это последовательность операторов SQL, которая принимается или отменяется как единое целое. Единая логическая единица или работа

Транзакция - **выполнение некоторой совокупности операций, которая принимается или отменяется как единое целое. Единая логическая единица или работа**

Все операторы SQL начинают транзакцию, **кроме следующих 18**:
Connect, Set Connection, Disconnect,
Set Session Authorization, Set Catalog, Set Schema,
Set Names, Set Time Zone,
Get Diagnostics, Set Transaction, Set Constraints,
Commit, Rollback,
Declare Cursor, Declare Local Temporary Table,
Begin Declare Section, End Declare Section, Whenever

Любая транзакция – это совокупность трех непременных составляющих:

- Запрос
- Выполнение
- Отчет

Транзакция базы данных, по определению, должна быть *атомарной* (она должна быть либо завершена полностью, либо не иметь никакого эффекта вообще), *согласованной* (она должна соответствовать существующим ограничениям в базе данных), *изолированной* (она не должна влиять на другие транзакции) и *долговечной* (она должна записываться в постоянное хранилище)

Атомарность (Atomicity). Все или ничего!
Либо фиксируются все операции транзакции (commit), либо откат (rollback)

Согласованность (Consistency). Шире понятия целостности.
Транзакция должна соответствовать существующим ограничениям в базе данных. Согласованность может быть реализована на уровне бизнес логики. Например, операция «списание» должна соответствовать некоторой операции «зачисление»

Изолированность (Isolation).
При выполнении транзакций параллельно одни транзакции не должны оказывать влияние на результат выполнения других транзакций.

Долговечность (Durability).

Если пользователь получил подтверждение от системы, что транзакция выполнена, он может быть уверен, что сделанные им изменения не будут отменены из-за какого-либо сбоя.

Транзакция называется **двухфазной**, если все операции блокировки предшествуют всем операциям снятия блокировки.

5. Параллельные транзакции, проблемы, решения

Потерянное обновление
Потерянные обновления возникают, когда две или более транзакций выбирают одну и ту же строку и изменяют ее на основании ее исходного значения. Каждая транзакция не знает о других транзакциях. Последнее обновление изменяет данные других транзакций, что приводит к потере данных. При обновлении поля двумя транзакциями одно из изменений теряется.

Преждевременное чтение
Рассмотрим следующий сценарий совместного выполнения транзакций 1 и 2. Транзакция 1 изменяет объект базы данных A. Параллельно с этим транзакция 2 читает объект A. Поскольку операция изменения еще не завершена, транзакция 2 видит несогласованные "грязные" данные.

Неповторяющееся чтение

- Возникает, когда в рамках одной транзакции при повторном чтении ранее прочитанные данные оказываются изменёнными. Это может произойти, если другая транзакция внесла изменения в прочитанные ранее данные.

Фантомные вставки
Возникают, когда транзакция выполняет запрос на выборку данных с использованием некоторого условия, а затем повторно выполняет этот же запрос, но видит другие данные, которые соответствуют условию. Таким образом, фантомные вставки возникают, когда другая транзакция вставляет новые строки данных между двумя повторными выполнениями запроса в первой транзакции.

Отличие между фантомными вставками и неповторяющимися чтениями заключается в типе изменений данных, которые наблюдаются при повторном выполнении операций. Фантомные вставки связаны с появлением новых строк данных, которые соответствуют условиям выборки, в то время как неповторяющееся чтение связано с изменением уже существующих данных. Обе этих проблемы возникают из-за параллельного выполнения транзакций, где одна транзакция влияет на данные, видимые другой транзакцией.

1. **Решение проблем параллельного исполнения транзакций**

1. Блокировки

- Элемент блокируется до начала операции.
- Примитив синхронизации. Ожидание снятия блокировки.
- Все установленные блокировки в конце концов должны быть сняты.

2. Пессимистические блокировки. Предотвращают доступ к данным для одновременных транзакций.

3. Оптимистические блокировки. Отслеживают возникновение конфликтов и при необходимости выполняют откат транзакций.

Что может быть элементом блокировки? (исходя из презентации)

- Отношение
- Кортеж
- Компоненты кортежа
- Возможно некоторая совокупность кортежей (блок)

Что может быть элементом блокировки? (исходя из ответов)

- Строка таблицы базы данных
- Физическая страница базы данных
- Таблица базы данных
- Элемент строки таблицы базы данных

Грануляции блокировок (Locking Level)

В логических единицах БД:

- Базы данных (database-level locking).
- Таблицы (table-level locking). Блокировки быстро устанавливаются и снимаются, но максимально ограничивают доступность данных
- Строки (row-level locking). Применяется к отдельной строке таблицы. Наиболее распространены.
- Элементы (item-level locking). Блокируется элемент (конкретное значение) строки или столбца. Вариант идеален с точки зрения параллелизма, но работает довольно медленно. Не имеет широкого распространения.

В физических единицах БД:

- Пространства БД или таблицы (dbspace-level locking, tablespace-level locking,)
- Страницы (dbspace-level locking). Единица хранения, настраиваемый размер, 1К.

Проблемы параллельного исполнения

1. **Бесконечные ожидания**
   Неформально: транзакция хочет заполучить ресурс, но этот ресурс всё время “перехватывают” другие транзакции (далее на 290 стр. Ульмана).
   Решение:
- Очередь транзакций

2. **Тупики (Чем они от бесконечного ожидания отличаются?)(стр 290 Ульмана)**
   Неформально: известные всем с пелёнок Deadlock-и (далее на 291 стр. Ульмана).
   Решение:
- Одновременная блокировка всех необходимых для транзакции элементов БД;
- Ввести ЛИНЕЙНОЕ упорядочение элементов БД, требовать запрос блокировок делать в соответствии с ЭТИМ порядком;
- Строим граф ожиданий. Вершины графа – транзакции. Каждый цикл в графе указывает на тупик. Анализируется построенный граф.В случае обнаружения цикла делается рестарт (Rollback) одной из транзакций попавших в цикл.

Сериализуемость (Параллельное исполнение транзакций КОРРЕКТНО т. и т. т., когда их совместный результат будет тем же самым, что и при исполнении этих транзакций в некотором последовательном порядке.)

Расписание совокупности транзакций – порядок в котором выполняются ЭЛЕМЕНТАРНЫЕ шаги этих транзакций (блокировка, чтение, и т.д.).

Расписание называется **последовательным**, если все шаги каждой транзакции выполняются вслед или перед всеми шагами других транзакций.

Расписание называется сериализуемым, если его результат ЭКВИВАЛЕНТЕН результату некоторого последовательного расписания.

6. Нормальные формы (1,2,3,БК)

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц)

1 Нормальная форма
[Неформально]:
Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице

Таблицы в 1НФ должны соответствовать определённым принципам:

- В таблице не должно быть дублирующих строк
- В каждой ячейке таблицы хранится атомарное значение (одно не составное значение)
- В столбце хранятся данные одного типа
- Отсутствуют массивы и списки в любом виде

[Формально]:
Отношение R находится в 1НФ, если все атрибуты определены на простых доменах, т.е. все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из более элементарных составляющих

2 Нормальная форма
[Неформально]:
Отношение R находится в 2 НФ, если оно находится в 1 НФ и все неключевые атрибуты R функционально полно зависят от ключа.

Таблицы в 2НФ должны соответствовать определённым принципам:

- Таблица должна находиться в 1НФ
- Таблица должна иметь ключ
- Все ключевые столбцы таблицы должны зависеть от полного ключа (если ключ составной)
- Если ключ состоит из нескольких столбцов, то все остальные неключевые столбцы должны зависеть от всего ключа, т.е. от всех столбцов в этом ключе
- В таблице не должно быть данных, которые можно получить, зная только половину ключа, т.е. только один столбец из составного ключа
- Таблица должна иметь правильный ключ, по которому можно идентифицировать каждую строку

[Формально]:
Отношение R находится в 2НФ если оно находится в 1НФ и все неключевые атрибуты функционально полно зависят от первичного ключа (не зависит от части ключа)

Множество атрибутов Y функционально полно зависит от X, если

- Имеется функциональная зависимость $X \rightarrow Y$
- Не существует такого $Z \subset X$ такого, что $Z \rightarrow Y$

3 Нормальная форма
[Неформально]:

- Таблица должна быть во 2НФ
- В таблицах должна отсутствовать транзитивная зависимость
  Транзитивная зависимость - это когда неключевые столбцы зависят от значений других неключевых столбцов

- Таблица должна содержать правильные неключевые столбцы (Неключевые столбцы не должны играть роль ключа в таблице)

[Формально]:
Отношение R находится в 3НФ, если оно находится в 2НФ и все неключевые атрибуты R нетранзитивно зависят от ключа.

Нормальная форма Бойса-Кодда
[Неформально]:
- Таблица должна быть в 3НФ
- Ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов
- Требования НФБК актуальны только для таблиц с составными ключами
- Часть составного первичного ключа не должна зависеть от неключевого столбца

[Формально]:
Отношение R удовлетворяет НФ Бойса-Кодда, т. и т. т., когда для любой нетривиальной зависимости R вида A1A2 … An → B множество {A1, A2, …, An } образует суперключ для R.

Тривиальные ФЗ. Говорят, что функциональная зависимость A1 A2 … An → B является тривиальной, если атрибут B совпадает с любым из атрибутов Ai , i = 1,…n

Нетривиальные ФЗ. A1 A2 … An → B1,B2, …, Bk называется нетривиальной, если по меньшей мере один из атрибутов Bi не является элементом множества {A1, A2, …, An }

Суперключ - подмножество атрибутов отношения, которое содержит в себе ключ

Суперключ - подмножество атрибутов отношения, удовлетворяющее требованию уникальности: не существует двух кортежей данного отношения, в которых значения этого подмножества атрибутов совпадают

Потенциальный ключ — подмножество атрибутов, минимальное по включению

Примеры заданий с НФ

Вопрос **4**

Выполнен

Баллов: 1,00 из 1,00

Отметить вопрос

В какой нормальной форме находится отношение R (**A, B**, C, D, E),

если **A, B** - ключ отношения

и имеется функциональная зависимость

B -> D

Выберите один ответ:

- в BCNF
- в третьей НФ
- в первой НФ
- во второй НФ

Вопрос **11**

Выполнен

Баллов: 1,00 из 1,00

Отметить вопрос

В какой **наибольшей нормальной форме** находится отношение R (**A, B**, C, D, E),

если **A, B** - ключ отношения и

имеются следующие функциональные зависимости:

C -> D

D -> E

Выберите один ответ:

- в третьей НФ
- в первой НФ
- во второй НФ
- в нормальной форме Бойса-Кодда

Вопрос **15**

Выполнен

Баллов: 2,00 из 2,00

Отметить вопрос

Выберите правильный вариант декомпозиции отношения R (**A, B**, C, D, E) при приведении его к 3-ей нормальной форме

A, B - ключ отношения

Имеются функциональные зависимости:

B -> C

D -> E

Выберите один ответ:

- R (A, B, C, D, E) разбивается на R1 (B, C), R2 (A, B, D) и R3 (D, E)
- R (A, B, C, D, E) разбивается на R1 (A, B), R2 (C, D) и R3 (D, E)
- R (A, B, C, D, E) разбивается на R1 (A, B, C), R2 (C, D, E)

Вопрос **28**

Выполнен

Баллов: 2,00 из 2,00

Отметить вопрос

В какой наибольшей нормальной форме находится отношение R (**A, B**, C, D, E), если **A, B** - ключ отношения и имеются следующие функциональные зависимости:

A, B -> C

A, B -> D

A, B -> E

Выберите один ответ:

- в первой НФ
- в нормальной форме Бойса-Кодда
- в третьей НФ
- во второй НФ

(Пример с экза) В какой максимальной нормальной форме находится отношение R(A, B, C, D, E), a,b - ключ но C ->A?

Максимум в 3 НФ, тк неключевой атрибут функционально определяет ключевой атрибут

Вопрос 23

Верно

Баллов: 1,00 из 1,00

Выберите вариант набора функциональных зависимостей, при котором отношение R (**A, B**, C, D, E) будет находиться в 3-ей нормальной форме если **A, B** - ключ отношения

Выберите один ответ:

- A, B -> C; B -> D; A, B -> E
- A, B -> C; A, B -> D; A -> E
- A, B -> C; A, B -> D; A, B -> E ✓
- A, B -> C; A, B -> D; D -> E
- A -> C; A, B -> D; A, B -> E

На экзе вопросы ниже все не спрашивали, но мало ли кому пригодится)
(По крайней мере челов, у которых посещаемость <50%)

---

7. Декомпозиция отношений

Определение

Декомпозиция - разбиение отношения на несколько других

Как правильно делать

$R(A_1, A_2, \dots A_n)$ разбивается на $S(B_1, B_2, \dots B_m)$ и $T(C_1, C_2, \dots C_k)$ если:

1. $\{A_1, A_2, \dots A_n\} = \{B_1, B_2, \dots B_m\} \cup \{C_1, C_2, \dots C_k\}$
2. Кортежи S являются проекцией кортежей R на атрибуты $B_1, B_2, \dots B_m$
3. Кортежи T являются проекцией кортежей R на атрибуты $C_1, C_2, \dots C_k$
4. При этом необходимо, чтобы

$$R(A_1, A_2, \dots A_n) = S(B_1, B_2, \dots B_m) \bowtie T(C_1, C_2, \dots C_k)$$

Декомпозиция отношения «Проекты»

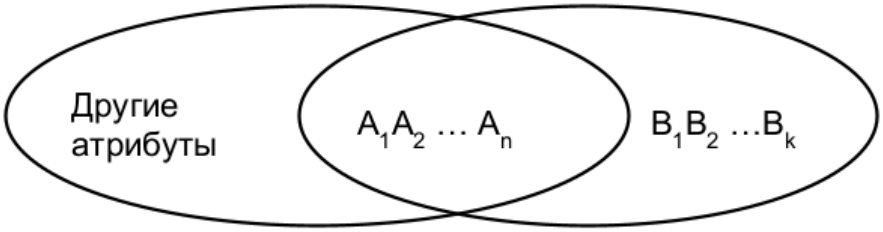
N_проекта	N_сотр	N_эт	Дата_нач	Дата_кон	Назв_проекта	N_отдела	Назв_отдела
12-08	1104	1	1.01.2012	31.03.2012	Лексический анализ	5	Разработки компиляторов
12-08	1105	1	1.01.2012	31.03.2012	Лексический анализ	5	Разработки компиляторов
12-09	1106	2	1.04.2012	30.06.2012	Синтаксический анализ	5	Разработки компиляторов
12-09	1107	2	1.04.2012	30.06.2012	Синтаксический анализ	5	Разработки компиляторов

N_проекта	N_сотр	N_эт	Дата_нач	Дата_кон	N_отдела	Назв_отдела
12-08	1104	1	1.01.2012	31.03.2012	5	Разработки компиляторов
12-08	1105	1	1.01.2012	31.03.2012	5	Разработки компиляторов
12-09	1106	2	1.04.2012	30.06.2012	5	Разработки компиляторов
12-09	1107	2	1.04.2012	30.06.2012	5	Разработки компиляторов

N_проекта	Назв_проекта
12-08	Лексический анализ
12-09	Синтаксический анализ

Приведение к нормальным формам

- Поиск в исходном отношении R нетривиальных функциональных зависимостей $A_1A_2 \dots A_n \rightarrow B_1B_2 \dots B_k$, нарушающих нормальную форму.
- Разбиение отношения R на 2 отношения, например, T и S, таких, что



- T будет состоять из атрибутов, образующих нетривиальную ФЗ,
- S – из атрибутов отношения R, входящих в ключ и не вошедших в отношение T.
- Исходное отношение должно восстанавливаться из 2-ух получившихся отношений с помощью **НАТУРАЛЬНОГО СОЕДИНЕНИЯ**

Отношение проекты

Проекты(N_проекта, N_сотр, N_эт, Дата_нач, Дата_кон, Назв_проекта, N_отдела, Назв_отдела)

ФЗ:

1. N_проекта, N_сотр, N_эт → Дата_нач, Дата_кон, Назв_проекта, N_отдела, Назв_отдела

2. N_проекта → Назв_проекта, Дата_нач, Дата_кон

3. N_отдела → Назв_отдела

Проекты(N_проекта, N_сотр, N_эт, Дата_нач, Дата_кон, Назв_проекта, N_отдела, Назв_отдела)

1. По ФЗ **2** разбиваем отношение **ПРОЕКТЫ** на 2 отношения (**приводим к II НФ**):

Проекты (N_проекта, Назв_проекта, Дата_нач, Дата_кон)
Участие (N_проекта, N_сотр, N_эт, N_отдела, Назв_отдела)

2. По ФЗ **3** разбиваем отношение **УЧАСТИЕ** на 2 отношения (**приводим к III НФ**):

Проекты (N_проекта, Назв_проекта , Дата_нач, Дата_кон)
Отделы (N_отдела, Назв_отдела)
Сотрудники-проекты (N_проекта,N_сотр, N_эт, N_отдела)

ВСЕ ?

8. Модель внешней памяти

Файл куча — последовательные данные или временной файл, это файл, в котором такая стратегия записи: выделяются блоки и они заполняются записями (блок всегда больше записи считаем).

Хэш — каждая хранимая запись базы данных размещается по адресу, который вычисляется с помощью специальной хеш функции на основе значения некоторого поля данной записи, т.е. хешполя, или хешключа. Вычисленный адрес называется хеш адресом.

Индексный файл – это хранимый файл особого типа, в котором каждая запись состоит из двух значений, а именно данных и указателя. Данные соответствуют некоторому полю (индексному полю) из индексированного файла, а указатель служит для связывания с соответствующей записью индексированного файла. Индексное поле также называется индексным ключом (index key).

9. СУБД

Определение
СУБД – это специальный программный комплекс для обеспечения доступа к данным и управления ими

В контексте СУБД, можем определить БД следующим образом:

База данных – совокупность данных находящаяся под контролем специального пакета программ - системы управления базами данных (СУБД)

Типовые функции СУБД:

1. **Определение конкретной базы данных:**
- типов данных, структур данных, ограничений

2. **Создание или Загрузка начального содержимого базы данных**

3. **Манипулирование базой данных**
- Извлечение данных: поиск по запросам, создание отчетов
- Модификация данных: вставка, удаление и обновление содержимого баз данных

4. **Управление многопользовательским доступом**

- Допускается одновременный доступ нескольких пользователей к извлечению, изменению и добавлению данных
- Управление совместным доступом гарантирует, что ТРАНЗАКЦИЯ либо успешно завершится, либо будет завершена без изменения содержимого БД

5. **Управление целостностью данных**

- Физическая целостность
    - Отсутствие нарушений спецификаций схемы хранения, а также физических разрушений данных на носителе
- Логическая целостность
    - Данные в БД остаются согласованными и корректными во всех отношениях

6. **Управление защитой от несанкционированного доступа к базе данных**

СУБД реализует **независимость данных** на 2 уровнях: логическом и физическом

Независимость данных в основном означает, что если данные изменяются на каком-то уровне, это не влияет на представление данных на другом уровне

Физический уровень: невосприимчивость приложений к изменениям в физическом представлении данных и в методах доступа к данным

Логический уровень: отсутствие влияния изменений в логической структуре базы данных на работу пользователей и пользовательских программ (Изменение состава полей в таблице БД, Изменение типов данных, Формирование представлений (виртуальных таблиц))

Компоненты СУБД:

- **Ядро**, которое отвечает за управление данными во внешней и оперативной памяти, и журнализацию,
- **Процессор языка базы данных**, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода
- **Подсистема поддержки времени исполнения**, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД
- **Сервисные программы** (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы

10. Запросы **(или мб про что то другое написать)**

Определение, структура, свойства, этапы оптимизации

11. Триггеры

Определение
Триггер - программный код (процедура), который хранится в СУБД и вызывается на исполнение только тогда, когда с указанной таблицей происходит определенное событие

Свойства триггера

- Триггер "срабатывает" только при наступлении определенного события (event), описание которого содержится в тексте триггера.
    - К числу допустимых событий (Event) обычно относят операции INSERT, DELETE и UPDATE указанного отношения. Во многих СУБД "событием", на которое способен отреагировать триггер, разрешено считать и факт завершения транзакции.
- Действие может выполняться либо до, либо после события триггера
    - Под действием (Action) понимается последовательность операций с базой данных, отвечающая логике определенного события. Оформляется как программный блок, хранящийся в базе данных
- Триггеры определяются ТОЛЬКО на таблицах, не на VIEW
- Перед выполнением операции триггер может проверить условие выполнения

В секции "действия" текста триггера разрешено использовать далеко **не все команды SQL**

Например, нельзя применять команды

- Управления транзакциями
- Организации соединения между клиентом и сервером базы данных (скажем, CONNECT и DISCONNECT),
- Создания (CREATE), изменения (ALTER) и удаления (DROP) элементов схемы,
- Определения параметров сеанса (SET) и пр

Функции триггера

- **Функция согласования данных.** Триггеры используются для обеспечения целостности данных в базе данных. Мы можем связать триггер с той или иной SQL командой, таким образом, чтобы триггер проверял связанные таблицы на согласованность данных
- **Функция очистки данных.** Данная функция является подмножеством функции из пункта 1. При наличии связей по внешнему ключу может выполниться каскадное удаление данных из таблиц, связанных ограничением внешнего ключа.
- **Функция журнализации.** Часто при помощи триггеров разработчики создают таблицы-журналы, в которых фиксируются различные изменения в базе данных.

Обычно журналы создаются для фиксации изменений, которые вносят различные пользователи базы данных, таким образом можно отследить какой пользователь внес то или иное изменение в ту или иную таблицу базы данных

12. Архитектура клиент-сервер

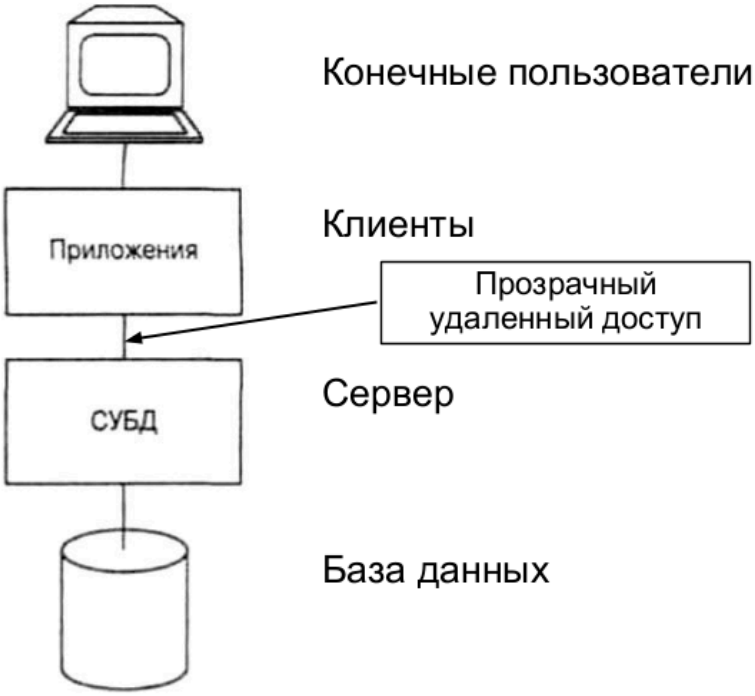
Определение
Термин "**клиент/сервер**" определяет архитектуру, или логическое разделение обязанностей

Клиенты — это различные приложения, которые выполняются с использованием СУБД. Это как приложения, написанные пользователями, так и встроенные приложения, предоставляемые поставщиками СУБД или некоторыми сторонними поставщиками программного обеспечения.

Сервер— это сама СУБД.
Он поддерживает все основные функции СУБД: определение данных, манипулирование данными, защиту данных, поддержание целостности данных и т.д. и предоставляет полную поддержку внешнего, концептуального и внутреннего уровней.

Клиент — это приложение, которое называют также интерфейсной частью (front end), а **Сервер** — прикладной частью (back end) или СУБД.

Типовая схема клиент-сервер архитектуры



13. SQL-92: Структура стандарта

Что тут писать? Напишите если знаете пжж

14. SQL-92. Операторы описания данных

Что тут писать? Напишите если знаете пжж

15. SQL-92. Операторы манипулирования данными

Что тут писать? Напишите если знаете пжж

16. SQL-92. Ограничения целостности

Операторы ограничения целостности

На уровне колонки в операторе CREATE TABLE:

PRIMARY KEY
REFERENCES
NOT NULL
UNIQUE
CHECK (…)

На уровне таблицы в операторе CREATE TABLE
PRIMARY KEY (…)
FOREIGN KEY (…)
UNIQUE (…)
CHECK (…)

На уровне баз данных
Assertion (ограничение общего вида, утверждения).Объявляются с помощью конструкции, содержащей служебное слово CREATE.
Состоит из следующих частей:

- Служебных слов CREATE ASSERTION
- Названия A ограничения
- Служебного слова CHECK
- Условия C, заключенного в круглые скобки
- CREATE ASSERTION A CHECK (C)

17. SQL-92. Операторы ограничения доступа.

Что тут писать? Напишите если знаете пжж

18. SQL-92. Представления.

До сих пор мы рассматривали данные из таблиц БД, или как говорят, постоянных базовых таблиц

SQL позволяет определять и использовать объекты, которые не содержат собственных данных

Эти объекты называются VIEW и свое содержимое они заимствуют из других таблиц

VIEW (Представление) – это именованная виртуальная таблица

Содержимое VIEW формируется в результате выполнения запроса **(какого запроса?)** **(select который задан при его определении)**
CREATE VIEW - создание представления

19. SQL-92. Курсоры.

Курсор (cursor) – объект, связанный с запросом

В контексте базы данных **курсор** представляет собой ресурс, который позволяет программам или пользовательскому коду взаимодействовать с результатами выполнения SQL-запроса. Курсор обеспечивает доступ к набору данных, возвращенному запросом, и позволяет последовательно перебирать и обрабатывать эти данные

Объявление курсора
EXEC SQL DECLARE CURSOR Томск_продавцы

Для начала работы с курсором нужно выполнить
оператор OPEN
EXEC SQL OPEN CURSOR Томск_продавцы;

Для построчного извлечения данных - FETCH
EXEC SQL FETCH Томск_продавцы
INTO :id_num, :salesperson, :loc, :comm;

В конце работы нужно выполнить оператор CLOSE
EXEC SQL CLOSE CURSOR Томск_продавцы;

20. Способы встраивания SQL в языки программирования

SQL/CLI (SQL Call Level Interface) входит в стандарт SQL 99 – это API на языке C

ODBC - (Open Database Connectivity) – на момент создания – промышленный стандарт.

JDBC (Java Database Connectivity) – скопирован с ODBC, но для вызовов из программ на Java, а не на C.

Блоки SQL-кода (modules). Аналогичен embedded SQL, но один и тот же блок кода могут вызывать разные приложения.

Dynamic SQL

Embedded SQL

21. Модели данных, кроме реляционной (Постреляционная, Сетевая, Иерархическая, ER)

Постреляционная модель представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных.

Важным аспектом традиционной реляционной модели данных является тот факт, что элементы данных, которые хранятся на пересечении строк и столбцов таблицы, должны быть неделимы и единственны

Модель допускает **многозначные поля** – поля, значения которых состоят из множества подзначений.

Набор значений **многозначных полей** считается самостоятельной таблицей, встроенной в основную таблицу.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей.

Это обеспечивает высокую наглядность представления информации и повышение эффективности её хранения и обработки.

Свойства постреляционной модели:

- О таблицах, содержащих многозначные поля, говорят, что они находятся в непервой нормальной форме (Non First Normal Form-NFNF) или NF2
- Так как многозначные поля подчиняются правилам, позволяющим обращаться с ними, как с таблицами, встроенными в другие таблицы, форма NF2 не нарушает принципы реляционной алгебры.

При условии, что вложенная таблица удовлетворяет общим критериям (например имеет уникальный ключ), естественным образом происходит расширение операторов реляционной алгебры.

В большинстве случаев гораздо более эффективно осуществлять доступ к многозначным полям одновременно с остальными данными, зная, что их всегда можно извлечь и рассматривать как отдельную таблицу в тех случаях, когда это может понадобиться.

(Chat GPT) Сетевая модель данных была разработана в 1960-х годах и использовалась в ранних информационных системах. В сетевой модели данные представляются в виде сети или графа, где каждый узел представляет сущность, а связи между узлами определяют отношения между этими сущностями

Базовыми объектами модели являются:

- Элемент данных
- Агрегат данных
- Запись
- Набор

Элемент данных — то же, что и в иерархической модели – минимальная информационная единица, доступная пользователю СУБД.

Агрегат данных соответствует следующему уровню обобщения в модели.

(Chat GPT) Элемент данных (Data Element) - это основной компонент данных в сетевой модели. Он представляет конкретные данные, которые нужно хранить или извлекать. Элемент данных может быть любым атомарным значением, таким как число, строка или логическое значение.

(Chat GPT) Агрегат данных (Data Aggregate) - это группа элементов данных, связанных друг с другом. Агрегат данных может содержать один или более элементов данных и представляет собой логическую структуру данных. Например, агрегат данных может представлять запись с несколькими полями.

(Chat GPT) Запись (Record) - это структура данных, которая представляет собой группу агрегатов данных. Запись может содержать несколько агрегатов данных, связанных между собой. Каждый агрегат данных в записи обычно имеет свой уникальный идентификатор, который используется для установления связей между записями.

(Chat GPT) Набор (Set) - это коллекция записей, которая представляет собой логическую группу связанных данных. Набор может содержать одну или более записей, которые могут быть связаны между собой посредством отношений. Наборы позволяют организовывать данные в структурированном виде и устанавливать связи между различными записями.

Утверждения о сетевой модели данных из теста:

- Тип записи может быть одновременно владельцем и членом нескольких типов наборов **(Какие владельцы? Какие члены типов? Что за тип записи?)**
- Между двумя типами записи может быть определено любое количество типов наборов **(Какие типы наборов? Почему любое количество?)**

В модели определены агрегаты двух типов:

- **Вектор**

- **Повторяющаяся группа**

Агрегат данных имеет имя, и в системе допустимо обращение к агрегату по имени.

- Агрегат **типа вектор** соответствует линейному набору элементов данных
    - Например, агрегат Адрес может быть представлен следующим образом:
    - Адрес = (город улица дом квартира)
- Агрегат **типа повторяющаяся группа** соответствует совокупности векторов данных.

Например, агрегат Зарплата соответствует **типу повторяющаяся группа** с числом повторений 12

Зарплата

Месяц	Сумма
янв	12500
фев	13000
мар	13500
апр	13000
май	14000
...	...
дек	14500

Записью называется совокупность *агрегатов* или *элементов данных*, моделирующая объект из некоторого класса объектов реального мира.

Понятие записи соответствует понятию "сегмент" в иерархической модели.

Для записи, так же как и для сегмента, вводятся понятия *типа записи* и *экземпляра записи*.

При моделировании реального мира могут быть варианты, что определить как тип записи

Компоненты СУБД сетевой модели:

- Язык описания данных схемы
- Язык описания данных подсхемы
- Язык манипулирования данными
- Резидентный модуль СУБД
- Язык управления размещением на внешних носителях

Иерархическая модель данных является наиболее простой среди всех даталогических моделей. Это модель данных, в которой данные организованы в древовидную структуру

Один объект выступает как родительский, а с ним может быть связано множество подчиненных объектов.

Иерархия проста и естественна в отображении взаимосвязи между классами объектов.

Основными информационными единицами в иерархической модели являются:

- **Поле** (элемент данных)
  Поле данных определяется как минимальная, неделимая единица (элемент) данных, доступная пользователю с помощью СУБД. (Например, адрес клиента)

- **Сегмент** (запись)
  Сегмент в терминологии Американской Ассоциации по базам данных DBTG (Data Base Task Group) называется записью, при этом в рамках иерархической модели определяются два понятия: тип сегмента и экземпляр сегмента.

  Тип сегмента (тип записи) — это поименованная совокупность типов полей (элементов данных), в него входящих.

  Экземпляр сегмента образуется из конкретных значений полей (элементов данных), в него входящих.

  Каждый тип сегмента в рамках иерархической модели образует некоторый набор однородных экземпляров сегмента.

  Ключом называется набор полей, однозначно идентифицирующих экземпляр сегмента.

- **Групповое отношение** (дерево – **физическая база данных**)

Ограничения иерархической модели (Ответ из теста):

- Каждый подчиненный сегмент может быть связан только с одним родительским сегментом
- Каждый сегмент может быть связан с произвольным числом подчиненных сегментов
- В каждой физической БД существует один корневой сегмент

Схема иерархической БД (совокупность деревьев)

—----------

ER-модель – это представление базы данных в виде наглядных графических диаграмм. ER-модель визуализирует процесс, который определяет некоторую предметную область. Диаграмма «сущность»-«связь» – это диаграмма, которая представляет в графическом виде сущности, атрибуты и связи

Основные компоненты ER модели:

- Множество сущностей
- Атрибуты сущностей
- Связи

Этапы *ПРОЕКТИРОВАНИЯ* ER модели:

1. **Выделение сущностей-объектов**
- Определение атрибутов сущностей
- Выделение идентифицирующих атрибутов- ключей
2. **Определение связей**
- Вид связи: 1:1, 1:m, n:m
- Обязательность связей
- Фиксация модели в виде ER-диаграммы

Принципы *ПРОЕКТИРОВАНИЯ* ER модели:

1. **Достоверность**
- Привлечение экспертов
- Детальное изучение предметной области
2. **Отсутствие избыточности**
3. **Простота**
- Включайте в проект только те структурные элементы, без которых нельзя обойтись
4. **Выбор подходящих связей**
- Множества сущностей можно соединить разными связями. Если брать любые – возможна избыточность
5. **Использование элементов адекватных типов**

Этапы *РАЗРАБОТКИ* ER модели:

1. **Разработка локальных представлений** (частей информационной системы
- Формулирование сущностей (имена, содержание)
- Выбор идентифицирующего атрибута
- Спецификация связей (имена, типы)
- Добавление описательных атрибутов

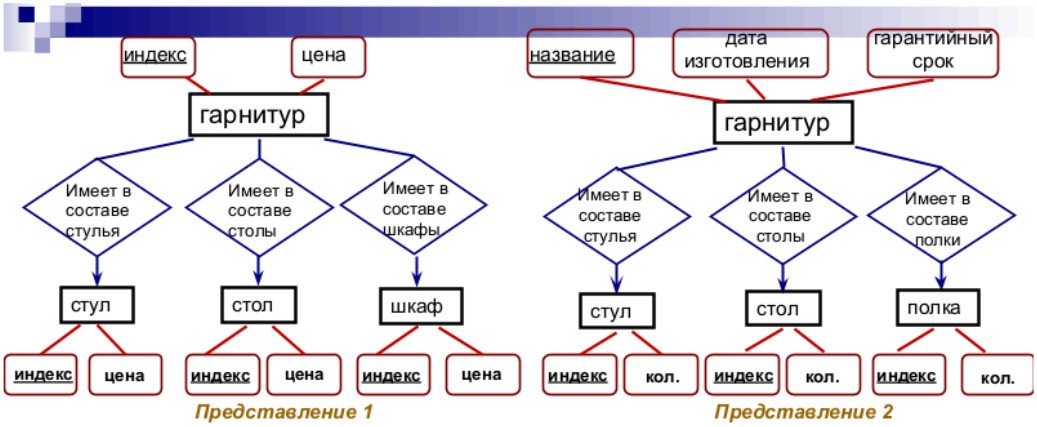
2. **Объединение представлений пользователей**

- Идентичность (Два или более элементов являются идентичными, если они имеют одинаковое семантическое значение)
- Агрегация (Декартово произведение сущностей)
- Обобщение (Абстракция данных, позволяющая трактовать класс объектов как ОДИН объект)

Различные операции (Видимо какие то обычные теоретико множественные) (Из презентации: Объединение представлений пользователей)

Идентичность

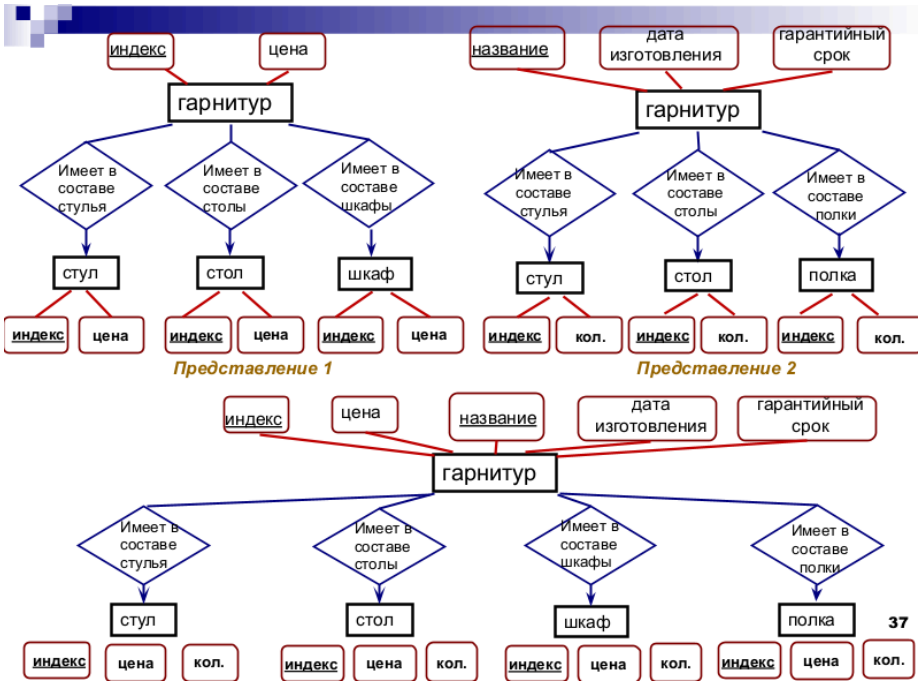
- Два или более элементов являются идентичными, если они имеют **одинаковое семантическое значение**
- В силу абстрагирования при выявлении сущностей идентичность элементов **устанавливается экспертом**
- Идентичность и подобие
- Персона, Служащий, Сотрудник, Работник, Персонал



Агрегация

- Агрегация – декартово произведение сущностей
- Новая сущность формируется на основе данных о частях объектах
  - Имя, Должность, Ном_отдела = Сотрудник
  - Имя, Паспорт, Дата приема = Сотрудник
  - Имя, Номер-страхового полиса, Адрес = Сотрудник

Имя, Номер-страхового полиса, Паспорт, Должность, Ном_отдела, Дата приема, Адрес = СОТРУДНИК



Обобщение

- Обобщение – это абстракция данных, позволяющая трактовать **класс объектов** как **ОДИН объект**
- При агрегации **части соединяются в целое**
- Обобщение фиксирует **РОДО-ВИДОВЫЕ отношения**
- **ВИД** есть **РОД** в совокупности с видовым отличием

- Ваз 2109, Ваз 2101, Ваз 2107 = Автомобили марки ВАЗ
- Автомобили марки ВАЗ, автомобили марки Хонда, … = Автомобили
- Автомобиль, велосипед, мотоцикл, трактор = Колесное средство передвижения

Итоговая схема