Оглавление

№1. Основные понятия и определения банка данных (БнД) Инфологический и датологический аспекты рассмотрения банка данных

Опр. ***Информация*** – это любые сведения о каком-либо объекте, событии или процессе, над которыми выполняются действия:

- восприятия;
- передачи;
- преобразования;
- хранения;
- использования.

Опр. Системы, служащие для регистрации, обработки, хранения и передачи информации, называют ***информационными системами***. Выделяют различные классы автоматизированных информационных систем: информационно-справочные, информационно-поисковые, информационно-логические и т.п.

Соотносительно двум понятиям – информация и данные, различают два аспекта при проектировании информационных систем: инфологический и даталогический.

1. ***Инфологический*** аспект употребляется при рассмотрении вопросов, связанных со смысловым содержанием данных, независимо от способов их представления в памяти системы. На этапе инфологического проектирования информационной системы должны быть решены вопросы:

- о каких объектах или явлениях реального мира требуется накапливать и обрабатывать информацию в системе;
- какие их основные характеристики и взаимосвязи между собой будут учитываться;
- уточнения вводимых в информационную систему понятий об объектах и явлениях, их характеристиках и взаимосвязях.

Таким образом, на этапе инфологического проектирования выделяется часть реального мира, определяющая информационные потребности системы, т.е. ее ***предметная область***.

2. ***Даталогический*** аспект употребляется при рассмотрении вопросов представления данных в памяти информационной системы. При даталогическом проектировании

- разрабатываются соответствующие формы представления информации в системе посредством данных,
- приводятся модели и методы представления и преобразования данных,
- формулируются правила смысловой интерпретации данных, т.е. формируется ***семантика*** данных.

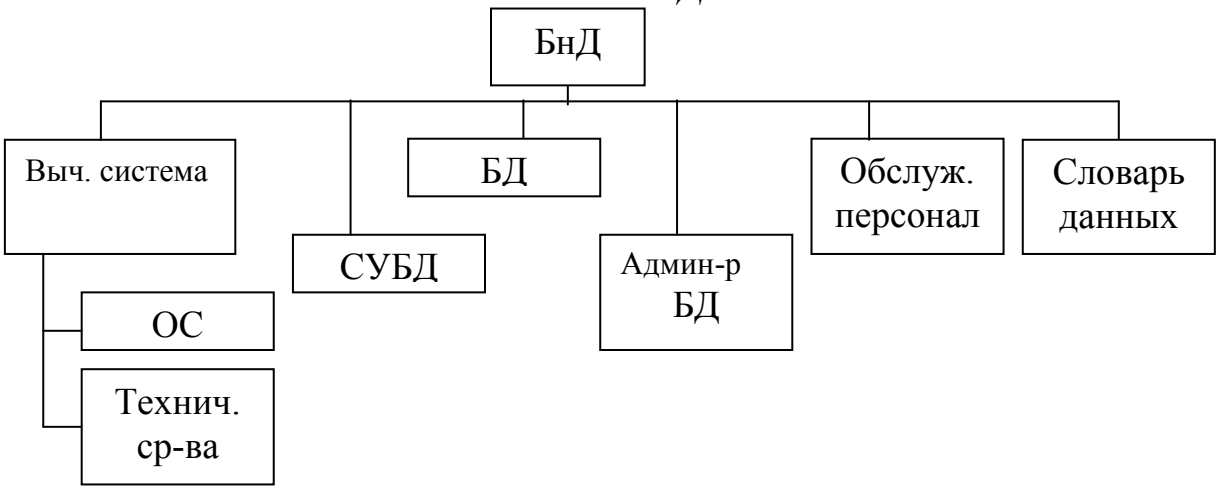
Опр. Данные, выражающие семантику данных, называются ***метаданными***.

Опр. ***Банк данных*** (БнД) – это система специальным образом организованных данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования этих данных.

Опр. ***Банк данных*** (БнД) – это автоматизированная информационная система, включающая в свой состав комплекс специальных методов и средств (математических, информационных, программных, языковых, организационных и технических) для поддержания динамической информационной модели ПО с целью удовлетворения информационных запросов пользователей.

№2. Состав БнД. Предпосылки создания БнД. Пользователи БнД.

Состав БнД:



Основными компонентами БнД являются БД и СУБД. Существует довольно много определений БД. Наиболее общее дано Коддом:

Опр. ***База данных*** (БД) – это поименованная, структурированная совокупность взаимосвязанных данных, относящихся к конкретной ПО.

Опр. ***Система управления баз данных*** (СУБД) представляет собой специальный пакет программ, с помощью которых реализуется централизованное управление БД и обеспечивается доступ к данным.

Функционирование БнД невозможно без участия специалистов, обеспечивающих создание и поддержку БД, организованный и контролируемый доступ к данным различных пользователей. Поэтому в состав БнД включен обслуживающий персонал и администратор БД.

Говоря о БнД, различают администратора баз данных (АБД) и АБнД.

Опр. ***Администратор баз данных*** (не обязательно специалист в области СУБД) должен принимать решения, какие данные нужно вносить в БД в первую очередь, обеспечивать порядок при обслуживании данных и использовании их после занесения в БД. АбД работает как управляющий, а не специалист по техническим вопросам.

Опр. ***АБнД*** – это профессиональный специалист в области информационных технологий. Он должен:

- создавать БД;
- осуществлять технический контроль;
- обеспечивать необходимое быстродействие системы и ее техническое обслуживание.

Функции АБнД выполняются несколькими специалистами (системными программистами).

Опр. Словарь данных – централизованное хранилище информации о самом БнД. Состоит из:

- базы метаданных (т.е. определение других объектов системы)
- набора программ, разработанных для решения задач, связанных с ведением и использованием этих данных (СУБмД)

Предпосылками создания БнД

1) Объекты ПО находятся в сложной взаимосвязи между собой
2) Информационные потребности различных конечных пользователей пересекаются;

3) При решении любой задачи, поставленной конечным пользователем, выполняется отбор данных, предварительно собранных и зафиксированных в информационной системе. Функции предоставления нужной информации конечному пользователю являются общими для разных задач.

С БнД в процессе его создания и эксплуатации взаимодействуют **пользователи** разных категорий:

4) ***Конечные пользователи*** – это специалисты, работающие в данной ПО, для удовлетворения информационных потребностей которых и создаются БнД. Понятием конечный пользователь определяется не только отдельное лицо или группа лиц, но и вычислительные процессы, задачи, а иногда и целые системы, взаимодействующие с БнД.
5) ***Программисты*** (прикладные), которые играют роль посредников между БнД и конечными пользователями. Современные тенденции в развитии информационных систем должны привести к существенному сокращению численности этих пользователей БнД, вплоть до полного их устранения.
6) ***Администраторы*** БнД и обслуживающие персонал. Службой администратора БнД называют структурное подразделение организации вычислительных центров или отделов АСУ, которое несет ответственность за создание БнД и его надежное функционирование, за соблюдение регламента к хранимым данным.

№3. Классификация банков данных

1. **По используемому языку общения.**

а) с базовым языком (замкнутая система);

б) с включаемым языком (открытая система).

Замкнутые системы имеют собственный самостоятельный язык общения с БД, который кроме операций манипулирования данными может выполнять арифметические операции, операции ввода-вывода. С включаемым языком – в качестве последнего выбран один из общепринятых алгоритмических языков (Ассемблер, Кобол), на котором пишется прикладная программа.

Жесткой границы между открытыми и замкнутыми системами нет. В настоящее время разрабатываемые системы все в большей степени наделяются свойством замкнутых систем.

2. По типу используемых моделей (по структурированности):

а) со структурированными БД – ориентированы на предварительную классификацию объектов реального мира, на установление свойств и связей между ними, которые будут фиксироваться в БД (БД с детерминированной схемой);

б) с не структурированными – совокупность видов свойств и видов взаимосвязей объекта с другими объектами определяется только в момент появления каждого реального объекта;

в) с частично структурированными.

Среди детерминированных систем в зависимости от типа МД, поддерживаемой СУБД, различают **иерархические**, **сетевые** и **реляционные** БнД.

Опр. Системы, которые поддерживают одновременно несколько различных моделей, называются ***мультимодельными***.

3. По выполняемым функциям.

а) информационные – позволяют организовать хранение данных, поиск и выдачу нужных данных из БД, поддерживают их целостность и актуальность;

б) операционные – кроме того, осуществляется иная обработка по получению информации, не хранящейся в явном виде в БД. Свойства «операционности» могут быть заложены в СУБД (автоматически получать сводные показатели, выполнять различные группировки) или обеспечиваться ПП.

4. **По сфере применимости.**

а) универсальные – настраиваются на ту или иную ПО путем создания соответствующих БД и ПП;

б) проблемно-ориентированные – могут быть обусловлены различными причинами: особенностями используемых языковых средств, включением в СУБД процедур обработки данных, присущих определенной области применения.

5. **По допустимым режимам работы.**

а) режим пакетной обработки;

б) диалоговый режим.

6. **По характеру хранимой информации. БД для:**

а) экономической;

б) научно-технической;

в) социально-политической;

г) технологической;

б) библиографической и т.п..

7. **По способу организации обработки данных (по степени распределенности).**

а) БД с локальным доступом;

б) с сетевым доступом:

1. клиент-сервер;
2. файл-сервер;

в) распределенные БД (РБД).

В системах с РБнД кроме понятия схема вводится понятие «суперсхема» – описание РБД как логически целой информационной совокупности.

б) – БнД реализуется на нескольких ЭВМ в сети (ЛВС). Файлы БД находятся на сервере, а программы обработки на рабочей станции.

в) – если файлы БД произвольно распределены по компьютерам ЛВС.

Существует много способов распределения данных по узлам сети. Крайними вариантами являются ***полностью избыточные системы***, в которых информация дублируется в каждом узле; ***разделенные системы*** – информация не хранится более чем в одном узле. Промежуточное положение занимает ***частично-избыточные системы***.

№4. Преимущества (7) и недостатки (4) организации БД

Преимущества организации СУБД.

1. Сокращение избыточности хранимых данных. Может быть обеспечена минимальная необходимая избыточность (дублирование) хранимых данных. При использовании несколькими программами одинаковых данных, такие данные интегрируются и хранятся в единственном экземпляре.
2. Устранение противоречивости данных. Это является следствием устранения избыточности данных.
3. Многоаспектное использование данных. Центральное управление позволяет реализовать однократный ввод данных и многократное (многоаспектное) использование данных.
4. Комплексная оптимизация. В максимальной степени устраняются противоречивые требования, предъявляемые конечными пользователями к хранимым данным.
5. Обеспечение возможности стандартизации данных, что упрощает эксплуатацию БнД.
6. Обеспечение возможности санкционированного доступа к данным, т.е. доступ к определенным группам данных может совершаться только пользователями с соответствующими полномочиями.
7. Обеспечение целостности данных. Задача целостности заключается в обеспечении правильности и точности данных в БД.

Наряду с достоинствами БнД присущи и недостатки:

Недостатки организации БД.

1. Увеличивается сложность создаваемых ИС; проектирование БнД требует выполнения большого числа ручных операций и высокой квалификации разработчиков.
2. Применение сложных структур данных увеличивает долю служебной информации в общем объеме хранимых данных. БнД предъявляет повышенные требования к применяемым в системе техническим и программным средствам. Часть ресурсов ПЭВМ, иногда довольно значительная, расходуется собственно на нужды самой системы управления БнД.
3. Последствия сбоев становятся более чувствительными и их труднее исправлять по сравнению с традиционной файловой обработкой.
4. Обеспечение независимости прикладных программ от изменений в хранимых данных становится насущной необходимостью. В противном случае требуется выполнять трудоемкие ручные операции по внесению соответствующих изменений в прикладные программы.

№5. Проектирование БнД. Два уровня независимости данных

ПРОЕКТИРОВАНИЕ БД. (общий подход)

I. Концептуальное проектирование

1) Описание предметной области
2) Отображение предметной области в некоторую модель данных, называемую ***концептуальной моделью***. В зависимости от ширины охвата ПО различают глобальные и локальные модели: ***глобальные*** – это описание ПО в целом; ***локальные*** – описание ПО с точки зрения конкретного пользователя.

Данный этап проектирования БД соответствует инфологическому аспекту рассмотрения БнД и соответственно называется инфологическим проектированием.

II. Логическое проектирование. Далее модель ПО отображается на МД, совместимую с выбранной СУБД. Такие модели также называются также ***логическими*** МД. Описание состава и логической организации БД называется ***схемой***; а соответствующий язык называется ***языком описания данных*** (ЯОД). Описание части БД, представляющей интерес для конкретных пользователей, называется ***подсхемой*** или ***внешней моделью***.

Любая СУБД поддерживает конкретную логическую МД, эту модель определяют в совокупности с ЯОД и ЯМД данной СУБД. ЯМД (или язык запросов) – это средство, позволяющее обращаться к БД. Данный этап есть даталогическое проектирование; когда учитываются возможности имеющихся технических и программных средств.

III. Физическое проектирование. И последний этап – отображение логической модели в физическую модель, специфицирующую размещение данных и методы доступа к ним. Эти модели называются ***внутренними моделями***. Внутренняя модель строится с учетом ограничений СУБД и ОС.

2 Уровня независимости данных

1) Физическая независимость (или независимость данных) – иммунитет приложений к изменениям в структурах хранения и методах доступа к ним. То есть логическая модель имеет иммунитет к изменениям в физической
2) Логическая независимость – независимость логической модели (а, следовательно, и программного обеспечения) от изменений в концептуальной модели.

Основные требования, которые необходимо выполнить при проектировании БнД

1. Адекватность отображения ПО (полнота данных, динамичность модели, актуальность информации, т.е. соответствие состоянию ПО на данный момент времени).
2. Возможность взаимодействия с пользователями разных категорий и в разных режимах.
3. Обеспечение конфиденциальности данных, надежности, целостности, защиты от случайного или преднамеренного разрушения БД.
4. Обеспечение взаимной независимости программ и данных.
5. Технологичность обработки данных, приемлемые характеристики функционирования БнД (стоимость обработки, время реакции системы на запрос, требуемый объем памяти).

2,5 – обеспечивается выбором СУБД, остальные обеспечиваются проектированием.

№6. Концептуальное проектирование БнД. Определение модели данных. Бинарная модель

Опр. ***Модель данных*** представим в виде тройки элементов:

M = < O, X, C >, где

O – объекты ПО;

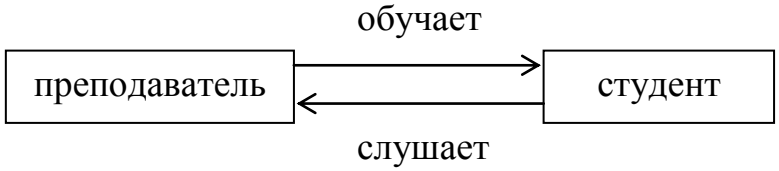
X – характеристики;

C – связи между объектами.

Опр. ***Объект*** – это то, о чем в информационной системе должна накапливаться информация. Объекты могут рассматриваться как **атомарные** и **составные**; причем один и тот же объект в различных ситуациях может быть рассмотрен как атомарный или как составной. Для **составного объекта** должна быть определена его **внутренняя структура** – как множество объектов, которые в свою очередь также могут быть составными и атомарными.

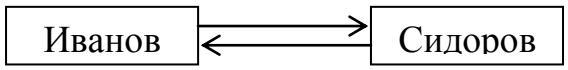
Таким образом, чтобы отобразить ПО в информационную систему (в МПО) нужно описать объекты ПО и их связи.

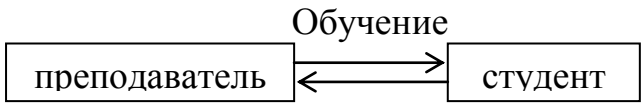
Бинарная модель – рассматривает пару объектов, которые называются ***категориями***:



Обоим направлениям бинарного отношения присваиваются уникальные имена, которые называются ***функциями доступа.***

Обобщенная бинарная модель – в качестве категории берется конкретный объект:





Расширение семантического бинарного графа состоит из реализации категорий и связей между ними. Реализации категорий могут быть ***конкретными*** или ***абстрактными***. Абстрактные категории всегда существуют, конкретные могут появляться и исчезать в описании реального мира средствами МД.

№7. Концептуальное проектирование. Модель сущность-связь. Примеры модели

ER-модель (модель «сущность-связь»).

Модель «сущность-связь» позволяет моделировать объекты ПО, в которой применяется БнД, взаимоотношения этих объектов. В модели «сущность-связь» используются 3 основных понятия: ***сущность, атрибут, связь*.**

Опр. ***Сущность*** – собирательное понятие, некоторая абстракция реально существующего объекта, процесса или явления, о котором необходимо хранить информацию в системе. В качестве сущностей могут рассматриваться как **материальные** объекты (предприятия, изделия), так и **нематериальные** (описание некоторого явления). **Сущность характеризуется** своим **типом** и **экземплярами** (тип – преподаватель, экземпляр – Иванов).

Тип сущности определяет набор однородных объектов, тип сущности должен быть поименован. ***Экземпляр сущности*** относится к конкретному объекту в наборе.

Опр. ***Атрибут*** вводится для описания сущности, и отображает характеристики объекта, которые представлены этой сущностью.

Преподаватель (ФИО; группа; дисциплина)

Студент (ФИО; группа; № зачетной книжки)

Атрибуты характеризуются **наименованием** – в скобках, и принимают значение из некоторого множества (Иванов).

Для того чтобы **задать атрибут в модели**, нужно

- присвоить ему наименование,
- привести смысловое описание атрибута,
- определить множество его допустимых значений
- указать для чего он используется.

Опр. ***Связь*** показывает взаимодействие объектов рассматриваемой ПО.

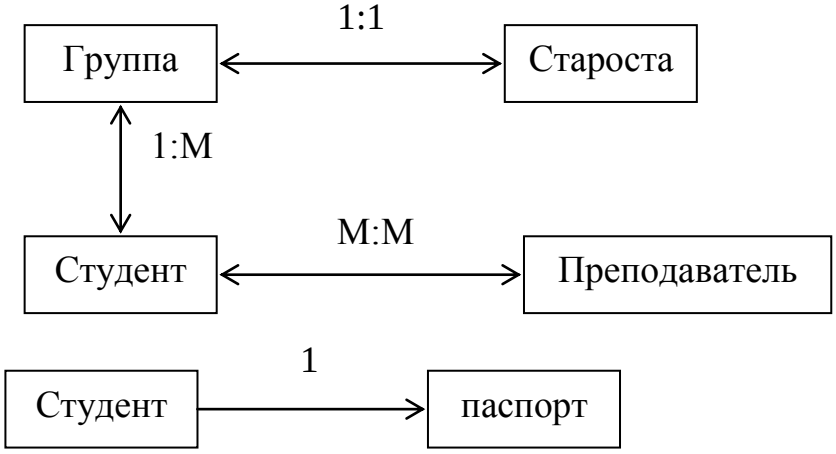
- **Тип связи** рассматривается между **типами сущностей**, а **конкретный экземпляр** связи рассматриваемого типа существует между **конкретными экземплярами** рассматриваемых типов сущностей.
- Существуют также **связи между атрибутами сущностей.**
- Связи могут быть **бинарные, тернарные** и **n-арные.**

Связи различаются по:

- количеству связанных объектов (связи типа 1:1, 1:M, M:1, M:M)
- по направленности связи (односторонняя или ассоциация и двусторонняя или отображение).

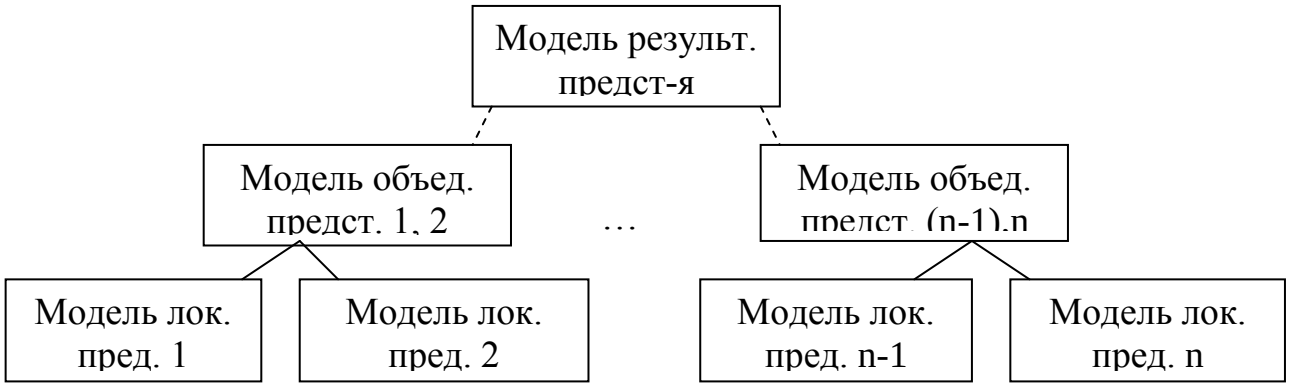
Опр. Условная ассоциация (или M-ассоциация) – одному экземпляру сущности A соответствует только один или ни одного экземпляра сущности B: $\xrightarrow{0,1}$

Пример:



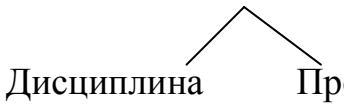
№8. Построение глобальной модели предметной области (ПО) по известным локальным моделям. Идентичность, агрегация, обобщение. Примеры локальной и глобальной моделей.

При проектировании всю ПО разбивают на локальные подобласти и для каждой подобласти строят модель. Затем эти модели объединяют и получают глобальную модель ПО. При объединении двух моделей, состоящих из N_1 и N_2 объектов соответственно, получается модель, содержащая $N_1 + N_2 - X$ объектов, где X – количество идентичных объектов в моделях. Процесс объединения изображают деревом следующего вида:



Стараются объединять достаточно близкие модели. При объединении используются три основополагающие концепции: идентичность, агрегация и обобщение.

Идентичность – если 2 или более объектов предметной области имеют одинаковое смысловое значение, то они объединяются в один объект (сущность).



Агрегация – позволяет рассматривать связь между сущностями модели как новую сущность. Сущность $A(B_1,...,B_n)$ в одной модели рассматриваем A целиком, а в другой $A(B_1, B_2)$.

При объединении моделей агрегация может встретиться в следующих формах:

1) В одном представлении определен сложный объект А как единое целое, а в другом представлении определены объекты B_1, B_2, B_3, которые являются составными частями А. Тогда $A(B_1, B_2, B_3)$.
2) Один и тот же агрегатный объект рассматривается в обоих представлениях, но составляющие различаются: $A(B_1, B_2, B_3)$ и $A(B_1, B_2, B_3, B_4) \Rightarrow A(B_1, B_2, B_3, B_4)$.

Обобщение: некоторым абстрактным понятием заменяется класс однотипных объектов и далее используется как как один поименованный объект. Применение обобщений позволяет организовать для пользователей доступ к данным с использованием различных уровней абстракции, что повышает гибкость системы для совместного использования данных.

Процесс объединения представлений носит **итеративный характер** всвязи с наличием противоречий:

- некорректность требований;
- различие требований в отдельных приложениях и у отдельных пользователей;
- неполнота спецификаций;
- наличие возможных ошибок.

Процесс объединения продолжается до тех пор, пока не будут интегрированы все представления, согласованы и устранены все противоречия.

№9. Логическое проектирование. Иерархическая модель данных. Примеры модели.

В основу ***иерархической*** модели данных положено понятие дерево.

Опр. ***Дерево*** – это неориентированный граф. Вершины графа – сущности, ребра графа – связи между сущностями. Одну из вершин, в которую не ведут никакие другие ребра, называют ***корнем***. Граф будет ориентированным и будет удовлетворять следующим условиям: в каждую вершину может заходить только 1 дуга, а выходить несколько. Это граф, который не имеет цикла.

$$T = \{t_1, ..., t_n\}$$

t_i – корень; $\forall T' \subseteq T$ – тоже является деревом.

Опр. Вершины, из которых не выходит ни одна дуга, называются ***листьями*** дерева.

Говорят о ***степени узла***. Это количество поддеревьев, которые выходят из данного узла. Лист имеет 0-ю степень. Остальные узлы, которые не являются ни листьями, ни корнями являются узлами.

Иерархическая древовидная структура – иерархическая модель:

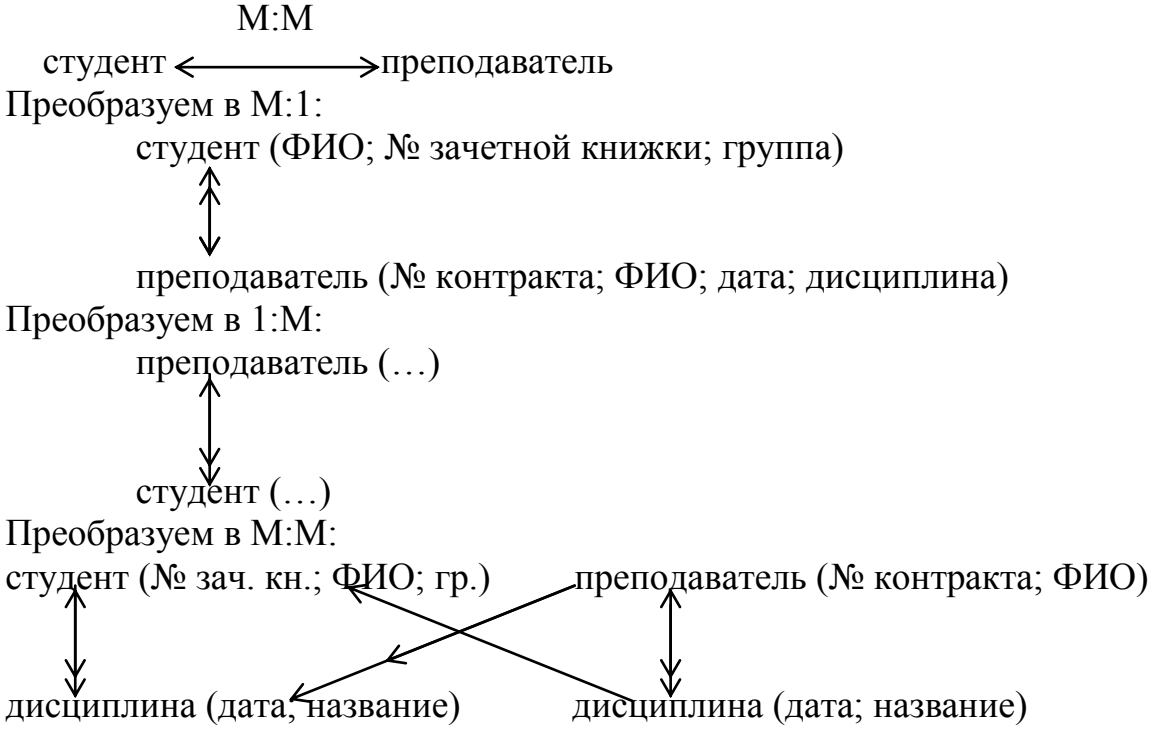
1. Иерархия начинается с корневого узла (1-й или 0-й уровень).
2. На следующих уровнях иерархии находятся порожденные узлы.
3. Каждый порожденный узел, находящийся на i-м уровне иерархии связан только с одним исходным узлом, находящимся на i-1-м уровне иерархии.
4. Каждый узел, кроме листа может иметь несколько порожденных узлов, которые называются ***подобными*** узлами.
5. Доступ к каждому узлу, кроме корневого, возможен только через корневой узел и через те узлы, для которых он является порожденным. Этот путь единственный.

Опр. Графическая интерпретация БД называется ***деревом определения***.

Преимущества: очень простая модель; подходит именно для иерархических структур.

Недостатки: не поддерживает связь M:M; из-за строгой иерархической упорядоченности объектов операции удаления и включения данных являются достаточно сложными; затруднен поиск данных: может быть только последовательный поиск.

Пример:



№10. Логическое проектирование. Сетевая модель данных. Примеры модели.

Опр. ***Элемент данных*** – наименьшая поименованная единица данных.

Опр. ***Запись*** – это поименованная совокупность элементов данных.

Опр. ***Набор*** – поименованная совокупность записей, образующих 2-х уровневую иерархическую структуру.

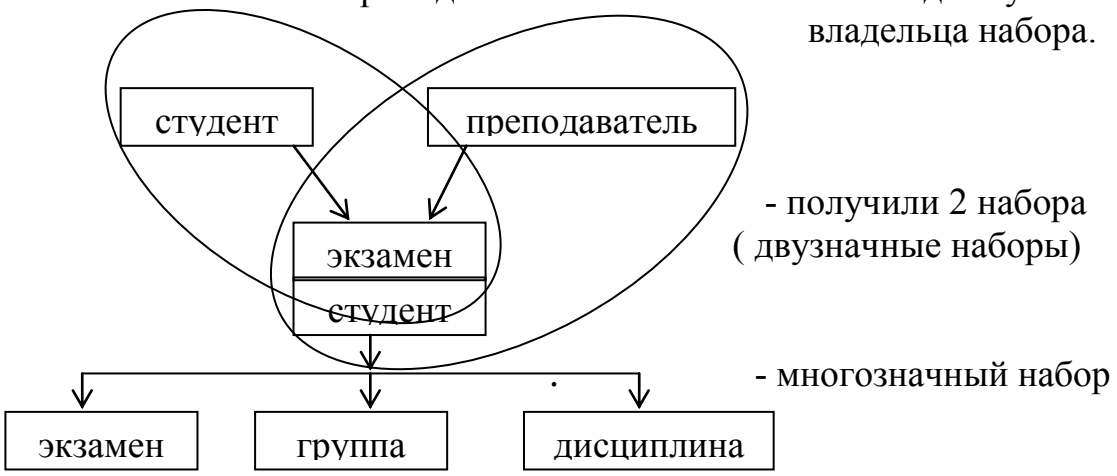
Опр. ***Область*** – это поименованная совокупность данных, содержащая экземпляры записей, наборов или частей одного и того же набора.

Различают тип записи и экземпляр записи. Любой тип записи может создать 1 поле, несколько полей или вообще ни одного поля.

Тип записи, из которой идет стрелка, называется ***владельцем***, а куда идет стрелка – ***членом набора***. Набор характеризуется типом и экземплярами.

Правила построения сетевой модели.

1. БД может содержать любое количество типов записей и любое количество типов наборов.
2. Между любыми 2-мя типами записей может быть установлено любое количество типов наборов.
3. Любой тип записи может быть одновременно владельцем набора и членом другого набора.
4. Экземпляр типа набора состоит из одного экземпляра типа записи владельца и 0, 1, или более экземпляров типа записи членов набора.
5. Между экземпляром типа записи владельца набора и экземплярами типа записи члена набора устанавливается связь типа 1:М.
6. Определенный экземпляр типа записи члена набора не может одновременно принадлежать более чем одному экземпляру типа записи владельца набора.



Опр. Наборы, не содержащие типа записи владельца набора, называются ***сингулярными***.

В любой БД может быть только 1 сингулярный набор, в который можно включать записи, не имеющие естественного владельца; если впоследствии такой владелец появится, то этот экземпляр записи из сингулярного набора исключается и включается в другой.

Преимущества:

- наглядность представления.

Недостатки:

- неоднозначность определения связей;
- непредставление связи М:М.

№11. Логическое проектирование. Реляционная модель данных. Примеры модели

Реляционная модель данных предложена Коддом в 1970 году.

В основу РМД положено понятие ***отношение.***

Опр. ***Отношением*** называется подмножество декартова произведения:

$R \subseteq D_1 \times ... \times D_n$, где D_i - домены:

$D_1 = \{d_{11}, d_{12}, ..., d_{1n_1}\}$

…

$D_n = \{d_{n1}, d_{n2}, ..., d_{nn_n}\}$

$R = \{T_1, ..., T_i, ..., T_k\}$

T_i – называется кортежем: $T_i = \{t_{i1}, t_{i2}, ..., t_{in}\}$, $t_{ij} \in D_j \quad \forall i = \overline{1,k} \quad \forall j = \overline{1,n}$, n – арность.

Опр. ***Таблица*** представляет собой n-арное отношение R, обладающее следующими свойствами:

1. одинаковые строки отсутствуют;
2. порядок строк безразличен;
3. порядок столбцов фиксирован;
4. строки и столбцы могут обрабатываться в любой последовательности.

Для того чтобы не фиксировать столбцы, их именуют, имена столбцов – атрибуты отношения R. Для каждого атрибута ставится в соответствие домен D_i, который содержит значение данного атрибута.

Опр. ***Схема отношения*** – это отношение, заданное вместе со своими атрибутами: $R(A_1, ..., A_n)$.

Опр. ***Ключом*** отношения R называется подмножество схемы отношения такое, что не существует 2-х одинаковых кортежей, принимающих одинаковые значения на данном ключе:

$K \subseteq R(A_1, ..., A_n) \quad \forall T_i, T_j \subseteq R: \; T_i(K) \neq T_j(K)$

Ключи задаются со схемой. Если их несколько, то они перечисляются:

$K = \{A_1A_2, A_1A_3\}$

Один из ключей называется первичным ключом.

Опр. ***Первичный*** ключ (суперключ) – это такой ключ K, что не существует подмножества $K' \subseteq K$ такого, что $\forall T_i, T_j \subseteq R: \; T_i(K) \neq T_j(K)$ и $T_i(K') = T_j(K')$.

Также могут быть неявные ключи, которые образуются каким-то образом из выделенных ключей.

Схему отношений можно описать: $R_{сх}(A, K)$.

Схема реляционной БД.

Пусть задано множество всевозможных атрибутов U (универсум): $A_i \in U \to D_i$.

Опр. ***Схемой реляционной БД*** называется множество всевозможных схем отношений, заданных на данном множестве атрибутов: $\{R^1_{сх}, R^2_{сх}, ..., R^\kappa_{сх}\}$; $R^i_{сх}(A^i, K^i)$ $A^i \subseteq U \quad K^i \subseteq A^i$.

№12. Логическое проектирование. Фактографические и динамические базы данных. Хронологическая модель данных.

Опр. ***Фактографические*** БД отражают текущее состояние предметной области.

Опр. ***Динамические*** отражают состояние предметной области в определенный момент времени или в интервал времени.

Для поддержания динамической БД используется хронологическая модель данных.

Опр. ***Временной ряд*** D^{θ} – это пары: $D^{\theta}{}_n = \left\{\left\langle d_i, t_i \right\rangle\right\}_{i=\overline{1,n}}$, где d_i – один или несколько атрибутов; t_i - момент времени; θ – единица измерения времени.

Пример:

D_5={<лаборант, 17 лет, 1980>, <инженер, 23, 1986>, <ассистент, 27,1990>,...}

Опр. Функция *восстановления*: $F_{\varepsilon}\left(D_n^{\theta}, t\right)$, ε – точность. Позволяет получить информацию об объекте в любой момент времени.

Опр. ***Временная шкала*** – это модель системы учета времени в реальном мире: $\langle \Omega, M, T \rangle$, где

- $\Omega = \{O, S\}$ - множество объектов O и события S, относящиеся к объекту;
- M – множество моментов времени;
- $T: \Omega \to M$ (отображение объектов/событий на моменты времени)

Опр. Состояние, в котором находится объект O между двумя соседними событиями называется ***темпором***. Темпор задает единицу измерения времени, а объект O называется ***времязадающим объектом*** или ***таймером***.

Опр. ***Хронологическая модель данных*** состоит из множества $F_{\varepsilon}\left(D_n^{\theta}, t\right)$ и множеств $\langle \Omega, M, T \rangle$.

№13. Операции над данными. Селекция и действие. Спецификационные и навигационные операции. Процедуры баз данных (БД).

Операции над данными

Опр. В общем случае модель данных: $MD = \langle G, O \rangle$, где

- G – множество правил вхождения (описывает синтаксис данных и соотносится с ЯОД),
- O – множество операций (соотносится с ЯМД).

Действие и селекция.

Опр. ***Действие*** показывает, что с данными нужно сделать:

1. обновление;
2. удаление;
3. добавление;

Опр. С*елекция* – какие данные нужно выбрать для данного действия:

1. по логической позиции – дает выбор 1-й, последней и т.д. записи;
2. по значению данных – выбираем данные, атрибут которых равен какому-либо значению;
3. по связи между данными – выбираем данные, для которых установлена связь.

Операции делятся также на навигационные и спецификационные. ***Навигационные*** – новые объекты не получаем, а ***спецификационные*** – дают в результате новый объект на основе существующих. К спецификационным относятся операции РА и РИ.

Процедуры БД.

Процедура включает в себя:

1) вычисления по данным;
2) вычисление значения атрибута по значениям других атрибутов;
3) получение статистических данных.

№14. Операции над данными. Операции над отношениями как над множествами. Привести примеры каждой из операций.

Так как отношение – это множество, то к нему применяются операции над множествами. Введем отношения одной арности *R* и *S*, заданные какими-то схемами:

R( ), S( ).

1) Объединение: $R \cup S = \{t \mid t \in R \vee t \in S\}$;
2) Пересечение: $R \cap S = \{t \mid t \in R \wedge t \in S\}$;
3) Разность: $R \setminus S = \{t \mid t \in R \wedge t \notin S\}$;
4) Декартово произведение: Пусть *R* – арность n_1, *S* – арность n_2

$R \times S = \{t_{RS}^{(n_1+n_2)} \mid t_R^{n_1} \in R \wedge t_S^{n_2} \in S\}$;

Неясно, какие примеры надо привести

№15. Операции реляционной алгебры: селекция и проекция. Свойства операций. Примеры выполнения операций.

Операция селекции (выбора)

Результатом ее применения к отношению R является другое отношение, которое представляет собой подмножество кортежей отношения R.

$\sigma_F(R)=\{t \mid t\in R \wedge t(F)="T"\}$;

F – формула, в которую входит атрибут, логические связки $\wedge,\vee$, константы, арифметические операторы сравнения $(<,=,>,\le,\ne,\ge)$, а также могут быть использованы скобки.

Пример: Студент(ФИО; год рождения; группа; № зач. книжки)

A_1 A_2 A_3 A_4

$A_3 = 647$ & $A_2 = 1986$

Свойства селекции:

1. операторы выбора коммутативны относительно операции композиции:

$\sigma_{A=a}(\sigma_{B=b}(R))=\sigma_{B=b}(\sigma_{A=a}(R))$, где R – отношение со схемой $R(A\,B); a\in D(A), b\in D(B)$. Т.к. порядок выбора не важен, то $\sigma_{A_1=a_1}\circ\sigma_{A_2=a_2}\circ\ldots\circ\sigma_{A_n=a_n}$ может быть записано как: $\sigma_{A_1=a_1,A_2=a_2,\ldots,A_n=a_n}$ (A_i не обязательно различны).

Доказательство: $\sigma_{A=a}(\sigma_{B=b}(R))=\sigma_{A=a}(\{t\in R|t(B)=b\})=\{t'\in\{t\in R|t(B)=b\}|t'(A)=a\}=$

$=\{t\in R|\ t(A)=a\wedge t(B)=b\}=\{t'\in\{t\in R|\ t(A)=a\}|\ t(B)=b\}=\sigma_{B=b}(\sigma_{A=a}(R))$

2. оператор выбора дистрибутивен относительно бинарных булевых операций:

$\sigma_{A=a}(R\gamma S)=\sigma_{A=a}(R)\gamma\,\sigma_{A=a}(S);\qquad \gamma=\{\cap,\cup,\setminus\}$

Доказательство: $\sigma_{A=a}(R\cap S)=\sigma_{A=a}(\{t|t\in R\wedge t\in S\})=$

$=\{t'\in\{t|t\in R\wedge t\in S\}|t'(A)=a\}=\{t|t\in R\wedge t(A)=a\}\cap\{t|t\in S\wedge t(A)=a\}=$

$=\sigma_{A=a}(\{t|t\in R\})\cap\sigma_{A=a}(\{t|t\in S\})=\sigma_{A=a}(R)\cap\sigma_{A=a}(S)$

Проекция

Пусть R – отношение со схемой $R(\ )$ и X – подмножество из $R(\ )$. Проекция R на X, записанная как $\pi_X(R)$ есть отношение $R'(X)$, полученное вычеркиванием столбцов, соответствующих атрибутам в $R(\ )\setminus X$, и исключением из оставшегося отношения повторяющихся строк.

$\pi_X(R)=\{t \mid t(x)\in R\}$

$R'=\pi_X(R)\qquad R'(\ )=X\qquad X\subseteq R(\ )$;

Пример: $\pi_{A_1A_3}$ (студент)

$\pi_{A_1A_3}(\sigma_{A_2=1979}$ (студент))

Свойства проекции:

1. Операции проекции и выбора коммутативны, если атрибуты, на которые проводится операция выбора, принадлежат множеству атрибутов, на которые осуществляется проекция. Если $A\subseteq X$, $X\subseteq R(\ )$ и R – отношение со схемой $R(\ )$, то: $\pi_X(\sigma_{A=a}(R))=\sigma_{A=a}(\pi_X(R))$.

Доказательство: $\pi_X(\sigma_{A=a}(R))=\pi_X(\{t\in R\mid t(A)=a\})=\{t'(X)|t'\in\{t\in R|t(A)=a\}\}=$

$=\{t(X)|\ t\in R\wedge t(A)=a\}=\sigma_{A=a}(\{t(X)|\ t\in R\})=\sigma_{A=a}(\pi_X(R))$

2. Если две проекции выполняются последовательно и вторая из них – собственная, то она поглощает первую: если π_X применимо к $\pi_X(R)$, то результат будет тем же самым, как при применении π_X непосредственно к R, если первоначальное применение π_X было собственным.

$\pi_{X_1}(\pi_{X_2}(R))=\pi_{X_1}(R),\quad X_1\subseteq X_2$.

3. Связь между проекцией и булевскими операциями:

$\pi_X(R\cup S)=\pi_X(R)\cup\pi_X(S)$

$\pi_X(R\cap S)=\pi_X(R)\cap\pi_X(S)$

$\pi_X(R\setminus S)=\pi_X(R)\setminus\pi_X(S)$.

№16. Операции реляционной алгебры: соединение, эквисоединение, Θ-соединение. Свойства операций. Примеры выполнения операций.

Соединение

Бинарный оператор для комбинирования двух отношений. Пусть заданы отношение R со схемой $R(\ )$, арности n_1, и отношение S со схемой $S(\ )$, арности n_2:

$R \triangleright\!\triangleleft S = \{t_{RS} \mid t_R \in R \wedge t_S \in S\}$

1. Если $R(\ ) \cap S(\ ) = 0 \Rightarrow$ декартово произведение ($R' = R(\ ) \cup S(\ )$).
2. $R(\ ) \cap S(\ ) \neq 0 \Rightarrow$ операция соединения по общим атрибутам (***естественное соединение***; $R' = R(\ ) \cup S(\ )$)

Пример:

R(A	B	C)	S(B	C	D)	$R \triangleright\!\triangleleft S$ (A	B	C	D)
a	b	c	b	c	d	a	b	c	d
d	b	c	b	c	e	a	b	c	e
b	b	f	a	d	b	d	b	c	d
c	a	d				d	b	c	e
						c	a	d	b

Свойства естественного соединения:

1. Операцию соединения можно использовать для определения операции выбора.
2. Операция соединения коммутативна и ассоциативна:
   $R \triangleright\!\triangleleft S = S \triangleright\!\triangleleft R$
   $R \triangleright\!\triangleleft (S \triangleright\!\triangleleft Z) = (R \triangleright\!\triangleleft S) \triangleright\!\triangleleft Z$
3. Операции соединения и проекции образуют дополняющие друг друга функции, хотя и не являются взаимно-обратными.
4. Операция соединения дистрибутивна относительно операции объединения:
   $R \triangleright\!\triangleleft (S \cup Z) = (R \triangleright\!\triangleleft S) \cup (R \triangleright\!\triangleleft Z)$.

Пример:

R(A	B	C)	S(D	E)	$R \triangleright\!\triangleleft S = R \times S$ (A	B	C	D	E)
a_1	b_1	c_1	d_1	e_1	a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_2	d_2	e_2	a_1	b_1	c_1	d_2	e_2
a_2	b_1	c_2			a_1	b_2	c_2	d_1	e_1
					a_1	b_2	c_2	d_2	e_2
					a_2	b_1	c_2	d_1	e_1
					a_2	b_1	c_2	d_2	e_2

Операция Θ – соединения

Пусть R и S – два отношения со схемами $R(A_1, A_2, .., A_n)$, $S(B_1, B_2, .., B_k)$, для которых $R(\ ) \cap S(\ ) = 0$, и пусть A_i и B_j θ-сравнимы (θ - арифметический оператор сравнения: $<, =, >, \leq, \neq, \geq$), тогда

$R \underset{A_i \theta B_j}{\triangleright\!\triangleleft} S = \{t_{RS} \mid$ для $t_R \in R$ и для $t_S \in S$ таких, что $t_R(A) \theta\, t_S(B)\} = \sigma_{1\theta(n+j)}(R \times S)$. $n-$ арность отношения R.

Пример:

$R \underset{A=D}{\triangleright\!\triangleleft} S = R'$ $\qquad R'(\ ) = R(\ ) \cup S(\ )$

$R' = R \underset{A=D}{\triangleright\!\triangleleft} S = \{t_{RS} \mid t_R \in R \wedge t_S \in S, t_R(A) = t_S(D)\}$, $\qquad A \subseteq R(\ ), D \subseteq S(\ )$.

R(A B C)	S(D E)	$R \triangleright\triangleleft S$ (A B C D E)
a_1 b_1 c_1	a_1 e_1	a_1 b_1 c_1 a_1 e_1
a_1 b_2 c_2	a_2 e_2	a_1 b_2 c_2 a_1 e_1
a_2 b_1 c_2		

Опр. Если в операции Θ – соединения присутствует знак "=", то операцию называют ***эквисоединением***. Если присутствуют другие операции – это Θ – соединение.

Вопрос: если Θ есть «<=», то это эквисоединение или же просто Θ – соединение?

Свойства эквисоединения:

1. Можно использовать для моделирования операции выбора

Доказательство.

а) Пусть для отношения $R(\ )$ надо найти $\sigma_{A=a}(R)$. Определим новое отношение $S(A)$ с единственным кортежем таким, что $t(A)=a$. Тогда $R \triangleright\triangleleft S$ есть то же самое, что $\sigma_{A=a}(R)$:

$R \triangleright\triangleleft S = \{t \mid \exists t_R \in R \wedge t_S \in S$ такие, что $t_S = t(A), t(R(\ )) = t_R$ и $t(S(\ )) = t_S\} =$

$= \{t \mid \exists t_R \in R \quad t_R = t(R(\ )) \wedge t(A) = a\} = \{t \in R \mid t(A) = a\} = \sigma_{A=a}(R(\ ))$.

б) Используя соединения можно сконструировать ***обобщенную операцию выбора***. Введем отношение $S(A)$ с k кортежами $t_1, t_2, .., t_k$, где $t_i(A) = a_i$ и $a_i \in dom(A), i = \overline{1,k}$. Тогда

$R \triangleright\triangleleft S = \sigma_{A=a_1}(R) \cup .. \cup \sigma_{A=a_k}(R) = \sigma_{A=a_1,..,A=a_k}(R)$.

в) Если выбрать 2 атрибута A и B из R и взять в качестве S отношение $S(AB)$ с единственным кортежем t таким, что $t(A)=a$ и $t(B)=b$, то

$R \triangleright\triangleleft S = \sigma_{A=a,B=b}(R)$.

2. Операторы соединения и проекции являются взаимодополняющими

Доказательство. Пусть $R(\ )$ и $S(\ )$ – отношения. $Q = R \triangleright\triangleleft S$ со схемой $Q(\ ) = R(\ ) \cup S(\ )$. Пусть $R' = \pi_R(Q)$, $R' \subseteq R$.

$t \in Q \quad \exists t_R$ и t_S, при этом t_R лежит в R или R', а $t_S \in S$. Если $t_R \in R$, то $t \in R \triangleright\triangleleft S$. Если $t_R \in R'$, то $t \in R' \triangleright\triangleleft S$. Тем самым показано, что $Q \subseteq Q'$. Докажем $Q \supseteq Q'$. Выберем $t \in Q' \wedge t \in R \triangleright\triangleleft S$ или $t \in R' \triangleright\triangleleft S$. В обоих случаях $t \in (R \cup R') \triangleright\triangleleft S$.

№17. Дополнительные операции реляционной алгебры (РА): частное и переименование. Примеры выполнения операций

Деление

Пусть R и S отношения арности r и s соответственно, $r > s > 0$. Тогда $R \div S$ есть множество кортежей t длины $r - s$ таких, что для любых кортежей $t_S \in S$: $tt_S \in R$:

$R \div S = \{t \mid \forall t_S \in S : tt_S \in R\}$

$R'(\ ) = R(\ ) \setminus S(\ )$

Пример:	R(A B)	S(B)	T(B)	$R \div S$(A)	$R \div T$(A)
	a_1 b_1	b_1	b_1	a_1	a_1
	a_1 b_2	b_2		a_3	a_2
	a_1 b_3	b_3			a_3
	a_2 b_1				
	a_2 b_2				
	a_3 b_1				
	a_3 b_2				
	a_3 b_3				

Переименование

Пусть задано отношение R, A является атрибутом отношения R, а $B \neq A$ не является атрибутом $R(\ ) - A$. Тогда отношение R с A, переименованным в B, обозначенное $\delta_{A \leftarrow B}(R)$ есть отношение:

$R' = \delta_{A \leftarrow B}(R) = \{t' \mid t(R(\ ) \setminus A) = t'(R'(\ ) \setminus B) \wedge t(A) = t'(B),\ t \in R\}.$

A и B должны иметь один и тот же домен. Что это значит?

Одновременное переименование:

Пусть имеем отношение R; $A_1,..,A_k$ – различные атрибуты, принадлежащие R, а $B_1,..,B_k$ – различные атрибуты, не принадлежащие $R(\ ) - (A_1,..,A_k)$, причем $dom(A_i) = dom(B_i)$ $\forall 1 \le i \le k$. Обозначим одновременное переименование атрибутов $A_1,..,A_k$ в $B_1,..,B_k$ соответственно в отношении R как $\delta_{A_1 \ldots A_k \leftarrow B_1 \ldots B_k}(R)$.

Замечание: Одновременное переименование атрибутов не всегда можно представить в виде последовательности переименований отдельных атрибутов: $\delta_{A,B \leftarrow B,A}$. В этом случае нужно вводить дополнительный атрибутный символ.

№18. Определение РА. Выражения РА. Схема выражений РА и ее свойства.

Опр. Реляционную алгебру определим как:

PA=<U, D, F, R_{cx}, O>,

где U – множество всех возможных атрибутов, называемое универсумом;

D – множество доменов;

F: U → D;

R_{cx} – множество всех возможных схем отношений, заданных на данном множестве атрибутов;

O – множество операций, заданных на отношениях со схемами из R_{cx} :

$$O=\left\{\cup,\cap,\setminus,R^*,\sigma,\pi,\triangleright\!\triangleleft,\delta,\div\right\}$$

Операции РА определил Кодд (за исключением операции переименования). Min O_{PA}< O, т.к. $\triangleright\!\triangleleft, \cap, \div$ можем определить через другие операции.

Цель РА – описать выражение для получения нового отношения (это выражение РА).

Опр. Выражением РА называется любое выражение, правильно построенное (согласующееся с ограничениями, наложенными на операторы) из отношений, использующих операторы из O. В выражении могут быть круглые скобки, именно они определяют порядок выполнения операций, т.к. все операции, за исключением $\cap,\cup$, равноправны.

Пусть записано выражение E: $(R_1, R_2,.., R_k)\to S$.

Схема полученного отношения будет зависеть от схем множества отношений, составляющих алгебру выражения E.

Опр. $sch(E)$ – схема алгебраического выражения E, можно определить рекурсивно:

1. Если выражение E состоит из одного отношения R_i, то $sch(E)=R_i(\ )$;
2. Если выражение $E=\left\{E_1\cup E_2, E_1\cap E_2, E_1\setminus E_2, \sigma_F(E_i),\right\}_{i=\overline{1,2}}$, где F – некоторое множество условий, то $sch(E)=sch(E_i)$;
3. Если $E=\pi_X(E_i)$, то $sch(E)=X$;
4. Если $E=\left\{E_1 \triangleright\!\triangleleft_\theta E_2, E_1 \triangleright\!\triangleleft E_2\right\}$, то $sch(E)=sch(E_1)\cup sch(E_2)$;
5. Если $E=E_1\div E_2$, то $sch(E)=sch(E_1)\setminus sch(E_2)$;
6. Если $E=\delta_{A_1\ldots A_n\leftarrow B_1\ldots B_n}(E_1)$, то $sch(E)=(sch(E_1)\setminus A_1\ldots A_n)\cup B_1\ldots B_n$.

Дейт, ссылаясь на работы Тодда и Дарвена, предложил новые алгебраические операторы: ***расширение*** и ***подведение итогов***.

Оператор расширения обеспечивает возможность горизонтального или построчного вычисления в алгебре (скалярные вычисления):

$$R'=Расш.(R)=\left\{t' \mid t'(R'(\ )\setminus N)=t(R(\ )), t\in R; t'(N)=\varphi(t)\right\},$$

где φ – арифметическое выражение, операндами которого являются атрибуты отношения R. Отношение R' есть исходное R, содержащее дополнительный атрибут. В качестве φ может быть взята функция: *count*, *sum*, max, min, *aog*.

Оператор подведения итогов выполняет аналогичную функцию для вертикальных вычислений:

$$R'=I(R)=\left\{t' \mid t'(R(\ )\setminus Z)=t(R(\ ))\wedge R'(\ )=R\cup Z\wedge t\in R\wedge t(Z)=\varphi(\pi_A(R))\wedge A\subseteq R(\ )\right\}$$

Формула непонятная. Возможно, она неправильна.

№19. Определение реляционного исчисления (РИ) с переменными-кортежами. Свободные и связанные переменные.

Аналогично тому, как РА использует в качестве операндов отношения, РИ кортежей строит свои выражения из кортежей.

Выражение в РИ строятся из атомов. ***Атомы*** могут быть трех типов:

1) $R(t)$, где R – имя отношения, а t – переменная-кортеж. Этот атом означает, что t есть кортеж отношения R.
2) $t_1^i \theta\, t_2^j$, где t_1 и t_2 – переменные-кортежи, θ – арифметический оператор сравнения; i, j – номера столбцов кортежей t_1 и t_2 соответственно. Этот атом означает, что i-й компонент t_1 находится в отношении θ с j-й компонентой t_2.
3) $t^i \theta C$, $C \theta t^i$, где θ и t^i – то же, что в 2); $C = const$. Этот атом означает, что i-й компонент кортежа t^i находится в отношении θ с константой C.

Переменная называется ***связанной***, если этой переменной в формуле предшествует квантор $\exists$ или $\forall$. В противном случае переменная называется ***свободной***.

Выражение определяется рекурсивно следующим образом:

1) Каждый атом является выражением. Переменные в выражении являются свободными или связанными в зависимости от того, были ли они свободными или связанными в атоме.
2) Если φ_1 и φ_2 – выражения РИ, то $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\neg\varphi_i$ – также выражения РИ. В результирующем выражении переменные будут свободными или связанными так же, как они были свободными или связанными в выражениях φ_1 и φ_2.
3) Если φ – формула, то $(\exists s)(\varphi)$ – формула, которая утверждает, что существуют значения s, при подстановке которых вместо всех свободных вхождений переменной s в формулу φ, эта формула становится истинной. Переменная s становится связанной, остальные свободны или связанны в зависимости от того, свободны они или связаны в формуле φ.
4) Если φ – формула, то $(\forall s)(\varphi)$ – формула. Эта формула утверждает, что какое бы значение подходящей арности мы не подставляли вместо свободных вхождений s в φ, формула φ является истинной. Переменная s становится связанной, остальные свободны или связанны в зависимости от того, свободны они или связаны в формуле φ.
5) Формулы могут заключаться в скобки. Если их нет, то приоритет операций: $\forall, \exists, \neg, \wedge, \vee$.
6) Ничто другое не является формулой.

Таким образом, формула в РИ с переменными-кортежами есть выражение вида $\{t \mid \varphi(t)\}$, где t – единственная свободная переменная-кортеж. А само исчисление кортежей определяется так же, как РИ только с другим О.

Опр. РИ представляет собой набор правил для записи выражений, определяющих некоторое новое отношение в терминах заданной совокупности отношений.

№20. Безопасные выражения РИ.

РИ с переменными-кортежами в том виде, в каком его определили, позволяет представить некоторые бесконечные отношения. Например, $\{t \mid \neg R(t)\}$ означает все возможные не принадлежащие R кортежи. Для того чтобы этого избежать вводится понятие ***безопасных*** выражений, для которых каждый компонент любого кортежа t, удовлетворяющий $\varphi(t)$, должен быть элементом множества $D(\varphi)$.

D(φ) - множество символов, которые явно появляются в выражении φ, либо являются компонентами какого-либо кортежа отношения R, упомянутого в φ. Множество D является функцией отношений, которые должны подставляться вместо переменных отношения в R. Т.к. R – все отношения предполагаются конечными, то и множество D конечно.

$$D(\varphi) = \bigcup_i C_i \bigcup_j \left(\bigcup_{l_j=1}^{n_j} \pi_{A_{l_j}}(R_j) \right)$$

R_j имеет n_j компонент, C_i – множество констант.

Опр. Выражение $\{t \mid \varphi(t)\}$ реляционного исчисления с переменными-кортежами называется ***безопасным***, если выполняются условия:

1. Всякий раз, когда кортеж t удовлетворяет выражению φ, каждая компонента кортежа t принадлежит множеству $D(\varphi)$.
2. Для любого подвыражения вида $(\exists t)(\omega(t))$, входящего в выражение φ, каждый компонент t должен принадлежать $D(\omega)$, если t удовлетворяет w.
3. Если для любого подвыражения вида $(\forall t)(\omega(t))$, входящего в выражение φ, каждый компонент $t \notin D(\omega)$, то t будет удовлетворять выражению ω.

Теорема: Если существует выражение РА, то существует эквивалентное ему безопасное выражение в РИ с переменными-кортежами.

Опр. РИ представляет собой набор правил для записи выражений, определяющих некоторое новое отношение в терминах заданной совокупности отношений.

№21. РИ с переменными на доменах. Безопасные выражения.

Реляционное исчисление с переменными на доменах.

Различие состоит в следующем:

1. В исчислении с переменными на доменах не существует переменных-кортежей; вместо них существуют переменные на доменах, которые представляют компоненты кортежей.
2. Атом имеет вид:

   а) $R(x_1,...,x_n)$ – n-арное отношение, где $x_i = const$ или переменной на домене. Эта запись указывает, что значения тех x_i, которые являются переменными на доменах, должны быть выбраны так, чтобы $x_1, x_2, .., x_n$ было кортежем отношения R.

   б) $x\theta\, y$; $x, y - const$ или переменные на доменах; θ – оператор сравнения; x, y должны быть выбраны так, что $x\theta\, y$ – истинно.
3. Формулы в РИ с переменными на доменах используют связки $\wedge, \vee, \neg$ и кванторы $\exists, \forall$.

Свободные и связанные переменные определяются так же, как и в РИ с переменными кортежами.

Выражение РИ с переменными на доменах имеет вид:

$$\{x_1,...,x_n \mid \varphi(x_1,...,x_n)\}, \qquad (*)$$

где φ – формула такая, что ее свободные переменные на доменах являются различными переменными $x_1,..,x_k$.

Опр. Выражение (*) называется безопасным, если:

1) Из истинности выражения $\varphi(x_1,...,x_n) \to x_i \in D(\varphi)$.
2) Если $(\exists u)((\omega(u))$ – подформула $\varphi)$, то из истинности $\omega(u) \to u \in D(\omega)$.
3) Если $(\forall u)((\omega(u))$ – подформула $\varphi)$, то из истинности $\neg\omega(u) \to u \in D(\omega)$.

Опр. РИ представляет собой набор правил для записи выражений, определяющих некоторое новое отношение в терминах заданной совокупности отношений.

№22. Эквивалентность выражений РА, выражений РИ с переменными-кортежами и выражений РИ с переменными на доменах.

Опр. РИ представляет собой набор правил для записи выражений, определяющих некоторое новое отношение в терминах заданной совокупности отношений.

Выражения исчисления с переменными на доменах эквивалентны выражениям исчисления с переменными-кортежами. Формализуем алгоритм перевода выражений исчисления с переменными-кортежами в выражения исчисления с переменными на доменах:

1. Если t является кортежем арности n, то введем n новых переменных на доменах $x_1, x_2, .., x_n$.
2. Атомы $R(t)$ заменяются атомами $R(x_1, x_2, .., x_n)$.
3. Каждое свободное вхождение t^i заменяется на x_i.
4. Для каждого квантора $\exists u$ или $\forall u$ вводится m новых переменных на доменах $u_1, u_2, .., u_m$, где m –арность u. Делаются замены:
   $R(u)$ на $R(u_1 u_2, .., u_m)$,
   u^i на u_i,
   $(\exists u)$ на $(\exists u_1)(\exists u_2)..(\exists u_m)$,
   $(\forall u)$ на $(\forall u_1)(\forall u_2)..(\forall u_m)$.
5. Выполняется построение выражения $\{x_1, ..., x_n \mid \varphi'(x_1, .., x_n)\}$, где φ' представляет собой φ, в котором проведены замещения 1-4.

Пример: $\{t \mid R_1(t) \vee R_2(t)\}$ перепишется $\{x_1, .., x_n \mid R_1(x_1, .., x_n) \vee R_2(x_1, .., x_n)\}$.

В РИ с переменными на доменах существуют 2 теоремы:

Теорема: Для каждого безопасного выражения РИ с переменными-кортежами существует эквивалентное безопасное выражение РИ с переменными на доменах.

Теорема: Для каждого безопасного выражения с переменными на доменах существует эквивалентное ему выражение РА.

№23. Реляционные языки манипулирования данными. Полный и более чем полные языки манипулирования данными. Примеры (4) языков манипулирования данными.

ЯМД основаны на РА. Кодд предложил язык Alpha, основанный на РИ. Этот язык является скорее абстрактным, т.к. нет систем, которые полностью бы его реализовали. DAMAS, INGRES реализовали этот язык частично. Этот язык обладает очень мощными средствами выборки:

- с упорядочиванием;
- с использованием $\forall$;
- с использованием $\exists$;
- с использованием $\forall,\exists$ (получить имена тех поставщиков, которые поставляют все детали; подсчет числа кортежей в отношении).
- Alpha содержит агрегатные функции count, total, max, min, average.

Опр. Язык, в котором можно (по крайней мере) моделировать исчисление с переменными-кортежами, либо, что равносильно операциям РА, или исчисление с переменными на доменах, называется ***полным***.

Опр. Реальные языки можно реализовывать функциями, не имеющими аналогов ни в РА, ни в РИ; их называют ***более чем полными***. В них возможно:

1) Арифметические вычисления. Атомы в выражениях РИ и формулах селекции в алгебраическом выражении часто могут включать арифметические вычисления, а также сравнения (+ многие другие арифметические операции не использующиеся в РА и РИ).
2) Команды присваивания и печати.
3) Агрегатные функции. К столбцам отношения часто применяют такие операции, как среднее, сумма, min, max.

Замечание. Язык Alpha является более чем полным.

Примеры ЯМД.

1) ISBL – основан на РА.
   a. **Нет** операторов обновления и модификации,
   b. **Не включены** агрегатные функции.
   c. Предусмотрено создание программ на PL/1
2) QUEL (query language) – основан на РИ с переменными-кортежами.
   a. **Имеет** процедуры вставки, удаления, изменения.
   b. **Содержит** агрегатные функции.
   c. Используется в системе INGRES
3) QBE (query by example) – основан на РИ с переменными на доменах.
   a. **Имеет** процедуры вставки, удаления, изменения.
   b. **Содержит** агрегатные функции.
4) SQL (structured query language) – близок к РИ с переменными-кортежами
   a. **Имеет** процедуры вставки, удаления, изменения.
   b. **Содержит** агрегатные функции.
   c. **Имеет** функции РА такие, как селекция и проекция.

№24. Целостность баз данных (БД). Основные виды ограничений целостности. Определение функциональных зависимостей.

ЗАЩИТА БАЗ ДАННЫХ.

Защита данных в БнД – это решение двух проблем:

1) ограничение целостности данных;
2) секретность данных.

Проблема целостности данных заключается в обеспечении соответствия данных, хранящихся в БД, реальному текущему состоянию ПО.

Опр. Логические ограничения, накладываемые на данные, называются ***ограничениями целостности***. Ограничение целостности – это свойство, которое для данного множества или отношения либо истинно, либо ложно. Это значение должно сохраняться для каждого возможного значения, в котором находится объект.

Существует 3 основных вида ограничений целостности:

1) Внутренние (структурные) – те ограничения, которые лежат в основе структуры БД. Например, РМД присуще внутреннее ограничение, что сущности и связи между сущностями представлены в виде отношений (таблиц).
2) Явные (семантические) ограничения целостности или утверждения.
3) Ограничения, вводимые на основе внутренних и явных. Это подразумеваемые.

Явные ограничения делятся на 2 вида:

1) Касается реальных значений, хранящихся в БД. Обычно такие ограничения требуют, чтобы значение атрибута принадлежало некоторому диапазону, или выражалось через какое-то арифметическое соотношение между другими атрибутами.
2) Показывает связь между атрибутами одного отношения. Записывается в ***виде функциональной зависимости***.

Функциональные зависимости.

Опр. Функциональные зависимости описывают ограничения, связанные с совпадением определенных компонентов кортежей.

Опр. Пусть $R(A_1, A_2, ..., A_n)$ – схема отношения, а X, Y – подмножества A (множество атрибутов). Говорят, что *X* ***функционально определяет*** *Y* (*Y* ***функционально зависит от*** *X*), и обозначается $X \to Y$, если в любом отношении *R* не существует 2-х кортежей, компоненты которых совпадают по всем атрибутам, принадлежащим X, но не совпадают по одному или более атрибутам, принадлежащим *Y*.

Функциональные зависимости возникают различными способами:

1) Если, например, X является ключом, то $X \to Y \ \forall Y$.
2) $R : E_1 \overset{M:1}{-} E_2$ – *R* есть отображение $M:1$ набора объектов E_1 к набору E_2. Среди A_i есть атрибуты, образующие ключ X для E_1, и ключ Y для E_2, то справедливо $X \to Y$.
3) $X \to Y$ и $Y \to X$ для $1:1$.

№25. Покрытие множества функциональных зависимостей

Покрытие множества зависимостей.

Пусть F и G – множества функциональных зависимостей. Будем считать, что F и G ***эквивалентны***, если $F^+ = G^+$. В этом случае говорят, что F покрывает G и, наоборот, G покрывает F.

Лемма: Каждое множество функциональных зависимостей F покрывается некоторым множеством функциональных зависимостей G, в которых ни одна из правых частей не состоит более чем из одного атрибута.

Теорема: Каждое множество функциональных зависимостей F эквивалентно некоторому минимальному множеству F'.

Опр. Говорят, что множество функциональных зависимостей F ***минимально***, если:

1) правая часть каждой зависимости в F состоит только из одного атрибута;
2) ни для какой зависимости $X \to A$ в F множество $F \setminus \{X \to A\}$ не эквивалентно F;
3) ни для какой функциональной зависимости $X \to A$ в F и собственного подмножества $Z \subseteq X$, множества $(F \setminus \{X \to A\} \cup \{Z \to A\})$ и F не эквивалентны.

В общем случае для множества функциональных зависимостей можем построить минимальное.

№26. Определение ключа схемы отношений. Правила (6) вывода функциональных зависимостей (ФЗ).

В связи с введением функциональной зависимости можно дать другое определение ключа:

Опр. Если $R(\ )$ – схема отношения с атрибутами $A_1, A_2, ..., A_n$ и множество функциональных зависимостей F, а X – подмножество $A_1, A_2, ..., A_n$, то X называется ***ключом отношения*** R, если

1) $(X \to A_1, ..., A_n) \in F^+$, где F^+ – ***замыкание множества функциональных зависимостей***;
2) Ни для какого собственного подмножества $Y \subseteq X$: $(Y \to A_1, ..., A_n) \notin F^+$. F^+ – все функциональные зависимости, которые могут быть получены для данного отношения с использованием правил вывода.

Правила вывода называются ***аксиомами функциональной зависимости.*** Пусть задана схема отношения $R(\ )$ с полным множеством атрибутов U и множеством функциональных зависимостей F, связывающем только атрибуты из U.

1°. аксиома рефлексивности: $X, Y \subseteq U,\ Y \subseteq X \Rightarrow X \to Y.$ Это правило дает тривиальные зависимости, т.е. зависимости, правая часть которых содержится в левой части. Его использование не зависит от F.

2°. аксиома пополнения: $X \to Y \wedge X, Y \subseteq U \wedge Z \subseteq U \Rightarrow X \cup Z \to Y \cup Z.$

3°. аксиома транзитивности: $X, Y, Z \subseteq U$: $(X \to Y, Y \to Z) \Rightarrow X \to Z.$

Пример: R(индекс, город, адрес)

$$F: \begin{cases} \underbrace{\text{index, city}}_{A_1} \to \underset{B_1}{\text{city}} \\ \underbrace{\text{city, address}}_{A_2} \to \underset{B_2}{\text{index}} \end{cases}$$

Первое получим пополнением функциональной зависимости $A_1 \to B_1$ атрибутом address: index, address $\to$ city, address

Второе объединением **(4°)**:

$$\left.\begin{matrix} \text{city, address} \\ \square \\ \text{index} \end{matrix}\right\} \Rightarrow \text{city, address} \to \text{city, address, index}$$

Третье транзитивностью:

$$\begin{cases} \text{index, address} \to \text{city, address} \\ \text{city, adress} \to \text{index, city, adress} \end{cases} \Rightarrow \text{index, address} \to \text{index, city, address}$$

Запишем в виде замыкания множества функциональных зависимостей.

$$F^+: \begin{cases} \text{index, address} \to \text{city, address} & (1) \\ \text{city, adress} \to \text{index, city, adress} & (2) \\ \text{index, address} \to \text{index, city, address} & (3) \end{cases}$$

Рассмотренные аксиомы являются надежными, т.е. приводят к истинным заключениям. Наряду с ними есть правила вывода, которые являются следствиями из 1°-3°:

4°. правило объединения: $X \to Y, X \to Z \Rightarrow X \to Y \cup Z.$

5°. правило декомпозиции: $X \to Y, Z \subseteq Y \Rightarrow X \to Z.$(вытекает из 1°, 3°).

6°. правило псевдотранзитивности: $X \to Y, Y \cup W \to Z \Rightarrow X \cup W \to Z.$

Правило объединения и декомпозиции порождают важное следствие:

Если $A_1, ..., A_n$ – атрибуты, то зависимость $X \to A_1, ..., A_n$ справедлива т. и т.т., когда $X \to A_i \quad \forall i$.

№27. Определение замыкания множества функциональных зависимостей и замыкания множества атрибутов относительно ФЗ. Алгоритм нахождения замыкания множества атрибутов.

Замыкание множества функциональных зависимостей

Опр. Пусть F – множество функциональных зависимостей на множестве U, тогда F^+ – замыкание множества функциональных зависимостей, если $\mathrm{F} \subset \mathrm{F}^+$ и все функциональные зависимости, полученные из F по правилам вывода, содержатся в F^+.

Замыкание множества атрибутов относительно множества функциональных зависимостей.

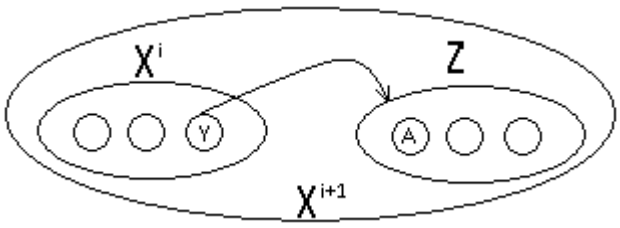
Опр. Пусть F – множество функциональных зависимостей на множестве U, и пусть $X \subseteq U$, тогда X^+ (замыкание X относительно F) есть множество атрибутов A таких, что зависимость $X \to A$ может быть выведена из F по аксиомам 1˚-3˚.

Лемма1: Функциональная зависимость $X \to Y$ следует из аксиом вывода, если $Y \subseteq X^+$.

Вычисление замыканий.

1) Вычисление F^+ для множества F является весьма трудоемкой задачей, т.к. множество F^+ может быть весьма большим, даже при малом F.
   В общем случае: $F = \{A \to B_1, A \to B_2, ..., A \to B_n\}$. Тогда F^+ включает все функциональные зависимости $A \to Y$, где Y – подмножество множества $\{B_1, ..., B_n\}$. Т.к. множеств Y 2^n, то F^+ огромно.
2) Вычисление X^+ для данного множества атрибутов X несложно, т.к. количество вычислений по времени пропорционально мощности множества F.
   Алгоритм вычисления:
   Дано: конечное множество атрибутов U, множество функциональных зависимостей F на U и $X \subseteq U$.
   1˚ $X^0 = X$;
   2˚ $X^{(i+1)} = X^i \cup A$
   A – множество атрибутов таких, что в F существуют функциональные зависимости $Y \to Z$, где $A \in Z$ и $Y \subseteq X^{(i)}$. Т.к. $X = X^0 \subseteq ... \subseteq X^{(i)} \subseteq ... \subseteq U$ и U – конечно, то мы должны в итоге достичь такого i, что $X^{(i)} = X^{(i+1)}$. Отсюда следует признак конца вычислений: $X^{(i)} = X^{(i+1)} = X^{(i+2)} = ...$. Тогда $X^+ = X^{(i)}$.

Выполняем пункт 2) до тех пор, пока не рассмотрим все функциональные зависимости.



№28. Определение декомпозиции схемы отношений.

Декомпозиция схем отношений.

Декомпозиция схем отношений является важным элементом проектирования БД. ***Декомпозицией*** схемы отношений $R(A_1, A_2, .., A_n)$ называется замена ее совокупностью схем $\{R_1(\ ), ..., R_k(\ )\}$ подмножеств R таких, что $R_1 \cup R_2 \cup .. \cup R_k = R$; не обязательно $R_i(\ ) \cap R_j(\ ) = 0$. При построении декомпозиции желательно выполнение двух свойств:

1) свойство соединения без потерь:
   $R = \pi_{R_1(\ )}(R) \triangleright\triangleleft \pi_{R_2(\ )}(R) \triangleright\triangleleft ... \triangleright\triangleleft \pi_{R_n(\ )}(R)$,
   где R является естественным соединением его проекций на все R_i $\forall i = \overline{1,k}$.

Метод, проверяющий сохраняется ли свойство соединения без потерь.

Строится таблица с n столбцами и k строками. Столбец j соответствует атрибуту A_j, а i-я строка отношению R_i. На пересечении строки i и столбца j поместим символ a_j, если $A_j \in R_i$. В противном случае туда поместим символ b_{ij}.

Повторно рассматриваем каждую из зависимостей $X \to Y$ в F до тех пор, пока в таблице невозможно будет сделать какие-либо изменения. Всякий раз, рассматривая зависимости $X \to Y$, мы ищем строки, которые совпадают по всем столбцам, соответствующим атрибутам из *X*. При обнаружении двух таких строк отождествляем их символы в столбцах, соответствующих атрибутам из *Y*. Если при этом один из отождествляемых символов = a_j, приравниваем и другому a_j. В том случае, когда они равны b_{lj} и b_{lj}, делаем их оба равными b_{lj} или b_{lj} по своему усмотрению.

После модификации строк таблицы может обнаружиться, что некоторая строка стала равной $a_1 ... a_k$. Тогда декомпозиция обладает свойством соединения без потерь. В противном случае она не обладает таким свойством.

Пример: $R(NATC) = R_1(NA) \cup R_2(NTC)$

$F = \{N \to A;\ NT \to C\}$

	N	A	T	C
R_1	a_1	a_2	b_{13}	~~b_{14}~~ a_4
R_2	a_1	~~b_{22}~~ a_2	a_3	a_4

Следовательно, декомпозиция обладает свойством соединения без потерь.

2) множество зависимостей F для R должно быть выводимо из проекций F на схемы отношений R_i.

   Формально проекцией F на множество зависимостей $X \to Y$ в F^+ таких, что $X \cup Y \subseteq R_i(\ )$ ($X \to Y$ не обязательно $\in F$). Будем говорить, что декомпозиция сохраняет множество зависимостей F, если из: $F = \bigcup_{i=1}^{k} \pi_{R_i(\ )}(F)$ по правилам вывода следуют все зависимости, принадлежащие F.

   Это следует из того, что F рассматриваются как ограничения целостности. Т.к. если бы из спроецированных зависимостей не следовало бы F, то мы могли бы обнаружить такие текущие значения R_i, представляющего отношение R в декомпозиции, которые не удовлетворяют F.

Может выполняться только одно из свойств. Декомпозиция может сохранять F, не обладая свойством соединения без потерь и наоборот.

№29. Аномалии, возникающие при проектировании БД. Нормализация отношений.

При проектировании БД возможно возникновение проблем

- **избыточности** и
- **аномалий** при выполнении операций
  - включения,
  - удаления,
  - обновления

данных.

Пример: Поставщик (ФИО; номер; адрес; товар; количество)

Избежать избыточности и аномалий позволяет нормализация отношений. Введены 4 уровня нормализации схем отношений, которые называются соответственно 4-мя нормальными формами.

Первая нормальная форма

Опр. Схема отношений $R(\ )$ находится в 1НФ т. и т.т., когда все входящие в нее атрибуты являются атомарными (неделимыми).

Вторая нормальная форма

Опр. $Y-$ ***первичный атрибут***, если он является элементом какого-либо ключа отношения R, т.е. существует ключ X отношения R такой, что $Y \subset X$. Аналогично определяется непервичный атрибут.

Опр. Пусть $A-$ непервичный, Y – первичный атрибут, а $X-$ ключ отношения R. Говорят, что в отношении R имеет место ***частичная зависимость*** (***неполная функциональная зависимость***), когда наблюдается:

$$X \to A \text{ и } Y \to A \qquad (*)$$

Опр. Если это условие (*) не выполняется, то говорят, что атрибут A ***функционально полно*** зависит от X в отношении R: $X \to A \quad \forall A \not\subset X \quad \neg\exists Y \subset X: \ Y \to A$.

Опр. Схема отношений $R(\ )$ находится во 2НФ, если она находится в 1НФ и каждый ее непервичный атрибут функционально полно зависит от ключа, т.е. для однозначной идентификации каждого неключевого атрибута требуется весь непервичный ключ.

Замечание. Отношения, находящиеся во 2НФ могут обладать аномалиями включения, удаления, модификации.

Третья нормальная форма

Опр. Для данной схемы отношений $R(\ )$, подмножества $X \subseteq R(\ )$, атрибута A в R и множества функциональных зависимостей F, атрибут A называется ***транзитивно зависимым*** от X в R, если существует подмножество $Y \subseteq R$ такое, что $X \to Y, \neg Y \to X, Y \to A$ относительно F и $A \notin X \cup Y$.

Другими словами, схема отношений находится в 3НФ относительно множества функциональных зависимостей F, если она находится в 1НФ и ни один из непервичных атрибутов в R не является транзитивно зависимым от ключа для R.

Замечание 1. Если схема в 3НФ относительно F, то она и в 2НФ.

Замечание 2. 3НФ **избавляет** от аномалий включения, удаления, модификации.

Нормальная форма Бойса-Кодда

Опр. Схема отношений $R(\ )$ находится в НФБК относительно F, если она находится в 1НФ и никакой атрибут в R не зависит транзитивно ни от одного ключа R.

Замечание. Если схема в НФБК, то она и в 3НФ.

Общее замечание. 2НФ, 3НФ и НФБК определены не друг через друга (т.е. НФБК есть расширенная 3НФ и т.п.), а как расширенная 1НФ!

№30. Алгоритмы приведения отношений к 3 НФ.

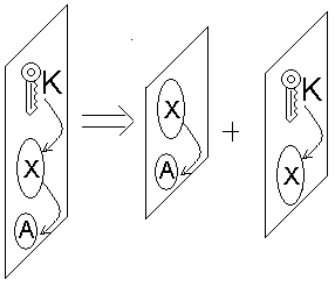
Алгоритм1: пополняющий декомпозицию схем отношений, которая обладает свойством соединения без потерь и приводит к отношениям находящимся в НФБК.

Входные данные: схема отношения $R(\ )$ и функциональные зависимости F.

Выходные данные: декомпозиция R, обладающая свойством соединения без потерь, такая, что каждая схема отношения в декомпозиции находится в НФБК относительно проекции F на эту схему.

Метод: декомпозиция ρ для R конструируется итеративным методом

$\rho = R$. Существует ключ K: $K \to X$. Если $S-$ схема отношения из ρ и S не находится в НФБК, то пусть $X \to A$ – зависимость, имеющая место в S, где X не содержит ключа S, а $A \notin X$.



Тогда в S должен существовать некоторый атрибут, который не принадлежит A и не принадлежит X. В противном случае X содержал бы ключ S. Заменим S на S_1 и S_2, где S_1 состоит из A и атрибутов X, а S_2 – из всех атрибутов S, за исключением A. $S_1 \cap S_2 = X$ и $S \setminus S_2 = A$. (это удаление транзитивной зависимости $K \to X, X \to A$.

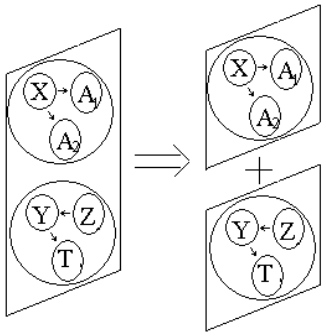
Если в S_1 и S_2 существуют транзитивные зависимости, то делаем декомпозиции и для S_1, S_2.

Алгоритм 2: приведения отношения к 3НФ, использующей декомпозицию, сохраняющую функциональные зависимости.

Входные данные: схема отношения $R(\ )$ и множество функциональных зависимостей F.

Выходные данные: сохраняющая зависимости декомпозиция схемы отношения R такая, что каждая входящая в нее схема отношения находится в 3НФ относительно проекции F на эту схему.

Метод: если существует некоторый атрибут в R, участвующий в левой или правой части какой-либо зависимости из F, то этот атрибут может сам по себе образовать некоторую схему отношения и его можно исключить из R. Если в одной из зависимостей F участвуют все атрибуты R, то выходные данные образуют само R. В противном случае декомпозиция, образующая выходные данные, состоит из схемы XA для каждой зависимости $X \to A$ в F. Если, однако, в F имеются зависимости $X \to A_1, X \to A_2, ..., X \to A_n$, то может быть использована схема $XA_1...A_n$ вместо XA_i $\forall 1 \le i \le n$, и такая подстановка является предпочтительной.



Теорема: Пусть $S-$декомпозиция R, образованная схемами отношений в 3НФ и построенная по алгоритму 2. Пусть $K-$ключ R. Тогда $S' = S \cup \{K\}-$ декомпозиция R такая, что все ее схемы отношений находятся в 3НФ. Эта декомпозиция сохраняет зависимости и обладает свойством соединения без потерь.

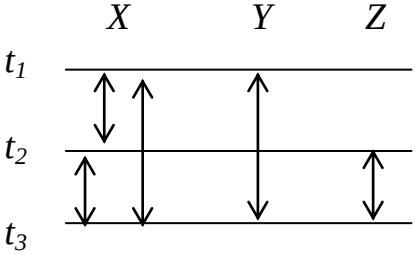
№31. Многозначные зависимости(МЗ). Правила вывода для МЗ. Четвертая НФ.

Опр. Пусть задана схема отношений $R(\ )$, атрибуты $X,Y \subseteq R(\ ),\ X \cap Y = 0$ $Z = R(\ ) \setminus (X \cup Y)$. $X \to\to Y$ (X мультиопределяет Y) – многозначная зависимость, если:

$\forall t_1, t_2 :\ \ t_1(X) = t_2(X) \quad \exists t_3 \in R :\ \ t_3(X) = t_1(X) = t_2(X),\ t_3(Y) = t_1(Y),\ t_3(Z) = t_2(Z)$

Из симметрии определения t_1 и t_2 следует, что в R существует t_4:

$t_4(X) = t_1(X),\ \ t_4(Y) = t_2(Y),\ \ t_4(Z) = t_1(Z).$



Правила вывода (аксиомы) для многозначных зависимостей.

A1. Дополнение

Если $X \to\to Y$, то $X \to\to U \setminus (X \cup Y)$

A2. Пополнение

Если $X \to\to Y$ и $V \subseteq W$, то $WX \to\to VY$

A3. Транзитивность

Если $X \to\to Y$ и $Y \to\to Z$, то $X \to\to Z \setminus Y$

A4. Объединение

Если $X \to\to Y$ и $X \to\to Z$, то $X \to\to Y \cup Z$

A5. Декомпозиция

Если $X \to\to Y$ и $Y \to\to Z$, то

$X \to\to Z \setminus Y$

$X \to\to Y \setminus Z$

$X \to\to Y \cap Z$

A6. Псеводтранзитивность

Если $X \to\to Y, W \cup Y \to\to Z$, то $W \cup X \to\to Z \setminus (W \cup Y)$

Утверждение 1. Пусть $X,Y,Z \subseteq R,\ X \to\to Y,\ Z = R \setminus (X \cup Y)$, тогда $X \to\to Z$

Утверждение 2. Пусть $X,Y,Z \subseteq R,\ Z = R \setminus (X \cup Y)$, то $X \to\to Z$ тогда, когда существует декомпозиция отношения R: $R \subset R_1 \cup R_2$, где $R_1 = X \cup Y,\ R_2 = X \cup Z$.

Она обладает свойством соединения без потерь.

Обозначим F – функциональные зависимости, а M – многозначные зависимости. Пусть $D = \mathrm{F} \cup M$.

Четвертая нормальная форма

Опр. Отношение R находится в 4НФ относительно D, если $\forall X$, содержащего ключ отношения R и мультиопределяющего Y ($X \to\to Y$), выполняется: $Y \neq 0,\ Y \subseteq X$ и $X \cup Y \neq R$.

Теорема. Если R в 4НФ относительно D, то оно в НФБК относительно F.

Для приведения отношения к 4НФ используется декомпозиция, которая обладает свойством соединеная без потерь тогда, когда существуют зависимости:

$R_1() \cap R_2() \to\to R_1() \setminus R_2()$ и $R_1() \cap R_2() \to\to R_2() \setminus R_1()$.

№32. Физическая организация БД. Внутренняя модель БД.

Опр. Физическое проектирование – отображение МД в среду хранения.

Задачи физического проектирования: выбор

- рациональной структуры хранения
- методов доступа к ней.

Опр. Внутренняя модель БД преставляет собой совокупность хранимых файлов с определенной структурой хранимых записей. Во внутренней модели определены:

- служебные поля, реализующие связи между хранимыми записямии
- методы доступа к ним.

Опр. ***Хранимое поле*** – это наименьшая именованная единица данных в БД.

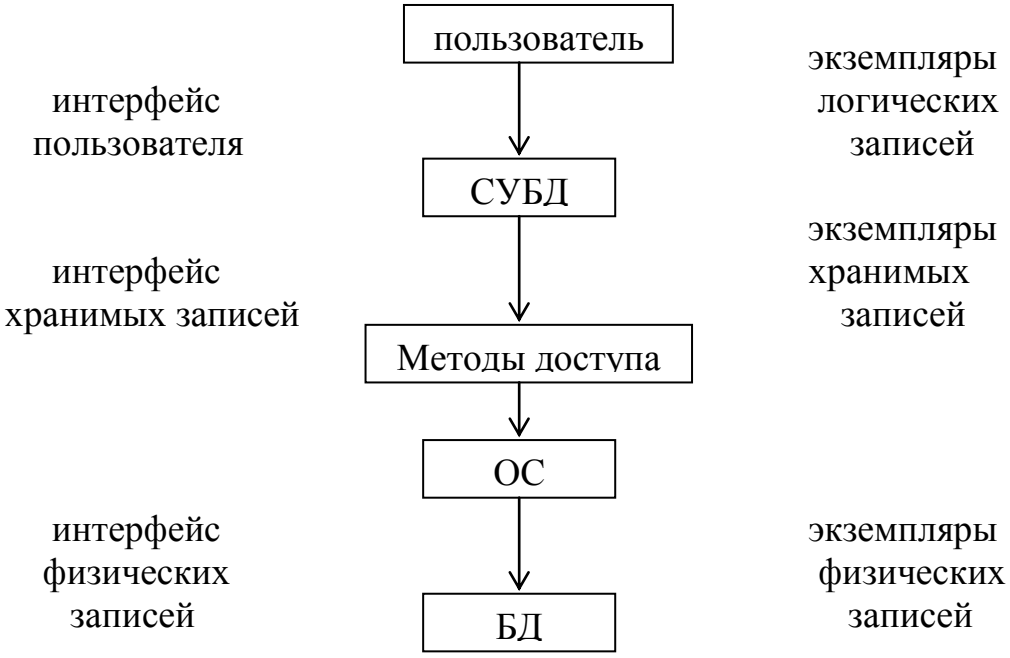
Опр. ***Хранимая запись*** – именованная совокупность связанных хранимых полей. Экземпляр хранимой записи состоит из группы соответствующих экземпляров хранимых полей.

Опр. ***Хранимый файл*** – именованная совокупность хранимых записей с определенной структурой.

Опр. Логические записи соответствуют хранимым записям, но могут быть образованы

- путем объединения двух различных типов хранимых записей;
- путем объединения полей одного типа хранимых записей.

Схема прохождения запроса



СУБД известно, какие существуют хранимые файлы и для каждого из них:

- структура соответствующих хранимых записей;
- хранимые поля упорядочения (если таковые есть);
- хранимые поля, которые могут использоваться как аргументы выборки при прямом доступе (если таковые есть).

СУБД неизвестно ничего о:

- физических записях (блоках);
- как хранимые поля связаны в хранимые записи;
- как осуществляется упорядочение (физическое следование, по индексу, по указателю);
- как выполняется доступ (посредством индекса; последовательным просмотром; хэш-адресация).

№33. Структуры хранения: с единственным хранимым файлом, с организацией индексного файла с произвольным числом указателей.

Рассмотрим возможные структуры хранения на примере РМД (поставщик)

Код поставщика	ФИО	Код товара	Город
1	Иванов	10	Самара
2	Петров	20	Самара
3	Андреев	10	Ульяновск
4	Сидоров	30	Москва
5	Егоров	20	Москва

С единственным хранимым файлом

Предполагает единственный хранимый файл, содержащий 5 экземпляров хранимых записей, по одному экземпляру для каждого поставщика.

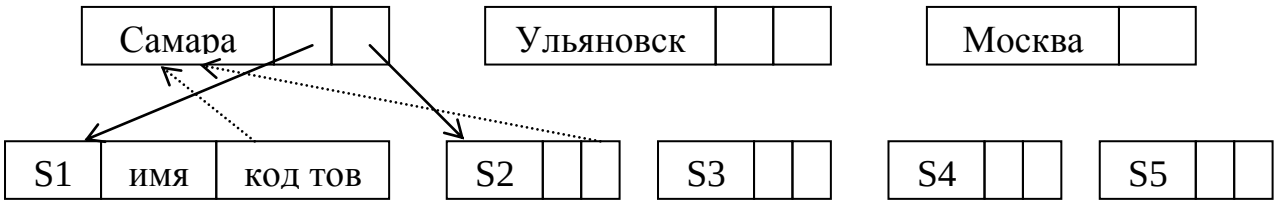
Преимущества:

– простота.

Недостатки:

– избыточность информации. Например, 10000 поставщиков из 10 городов (информация о городах избыточна). Метод доступа – последовательный.

С организацией индексного файла с произвольным числом указателей



Преимущества:

– быстрый поиск.

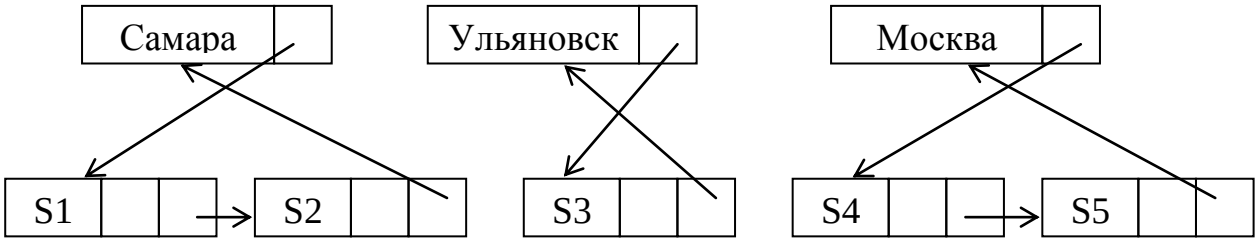
Недостатки:

– потери большого пространства памяти;
– необходимости прилагать больше усилий для поддержания указателей при изменениях;
– непредсказуемое число указателей в каждом элементе индекса, т.е. в каждом экземпляре хранимой записи в индексе.

№34. Структуры хранения: с организацией индексного файла с единственным указателем, с инвертированным индексным файлом.

С организацией индексного файла с единственным указателем

Каждый экземпляр хранимой записи файла «Поставщик» или файла «Город» содержит только один указатель. Каждая запись файла городов указывает на запись первого поставщика в этом городе. Эта запись поставщика указывает на запись второго поставщика того же города и т.д.; последняя запись содержит указатель на город. Таким образом, для каждой записи города мы имеем цепочку записей всех поставщиков в этом городе. Этот способ называется ***многосписочной организацией***.



Преимущества:

– простота изменений.

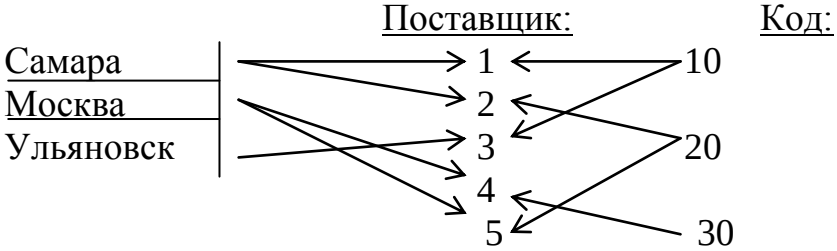
Недостатки:

– для города единственным способом доступа к n-му поставщику является последовательное прохождение по цепочке от 1-го ко 2-му и т.д. до (n-1) поставщика. Если каждая операция доступа включает операцию поиска, то время доступа к n-му поставщику может стать достаточно большим.

Замечание. Развитие этого варианта: двунаправленные цепочки; включение в запись поставщика указателя на соответствующую запись города, для того, чтобы сократить объем просматриваемых ссылок при выполнении некоторых типов запросов.

С инвертированным индексным файлом

Можно сделать произвольное число вторичных индексов. В предельном случае предусмотрен индекс по каждому вторичному полю. Это ***инвертированная организация структуры хранения***.



Преимущества:

– хорошая производительность для запросов о всех поставщиках, обладающих определенным значением атрибута.

Недостатки:

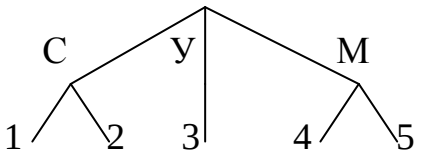
– ответ на запрос о всех атрибутах конкретного поставщика потребует продолжительного времени;

– требует дополнительной памяти для хранения значений индексируемых полей.

№35. Структуры хранения: иерархическая организация, хэш-адресация

Иерархическая организация

Является разновидностью индексированной. Вид хранимой записи зависит от запроса. Один хранимый файл, содержащий три экземпляра хранимой записи, по одному экземпляру на город. Каждый экземпляр хранимой записи содержит список произвольной длины, каждый экземпляр которого содержит номер поставщика, его имя и код товара. В этой же записи есть данные о городе.



Хэш-адресация

Каждый экземпляр хранимой записи размещается в БД по адресу хранимой записи, который может быть вычислен как некоторая функция h (хэш-функция) от значения некоторого ключа в этом экземпляре записи – обычно значения первичного ключа. То есть запись со значением первичного ключа υ хранится по адресу $h(\upsilon)$.

Чтобы заполнить экземпляр, СУБД вычисляет адрес хранимой записи и заставляет метод доступа разместить этот экземпляр в соответствующей позиции; при поиске экземпляра СУБД выполняет те же самые вычисления и запрашивает метод доступа для вызова экземпляра из вычисленной позиции.

Опр. Алгоритм преобразования ключа в адрес называется подпрограммой рандомизации или хэширования.

Преимущества:

- быстрый прямой доступ на основе значений хэшируемого поля.

Недостатки:

- диапазон возможных значений ключа шире диапазона адресов;
- возможно возникновение коллизий, связанное с вычислением одинакового значения для адресов двух различных записей;
- неэкономное расходование памяти;
- последовательность экземпляров хранимых записей в пределах хранимого файла не совпадает с последовательностью, определяемой первичным ключом.

№36. Методы доступа к данным.

Опр. ***Интерфейс физических записей*** – это интерфейс между методом доступа и физической БД. Единицей передачи через этот интерфейс является экземпляр физической записи (один блок). В этом интерфейсе, в отличие от интерфейса хранимых записей, важно физическое следование записей, хранимых в файле.

Метод доступа – это средство, которое использует СУБД для детального управления физическим доступом к БД.

Опр. ***Методы доступа*** совокупность подпрограмм, процедур, которые позволяют:

- скрыть детали управления устройствами от СУБД;
- обеспечить СУБД ***интерфейсом хранимых записей***.

Методы доступа к данным.

1. ***Последовательный МД***. Организация в виде списков и смежная.
   1.1. В случае *смежной организации* операции добавления и удаления происходят по следующей схеме:
   - доступ к данным на время блокируется;
   - в буфере создается файл, в котором отражены изменения;
   - исходный файл перезаписывается с учетом информации буферного файла.

   Недостаток: запрос более долгий.
   1.2. В случае *организации в виде списков* блокировка минимальна, – только на время модификации указателя.
   - При добавлении отыскиваются две записи, между которыми лежит искомая, изменяются указатели.
   - При удалении указатель на удаленную запись переставляется на следующий после удаленной. В результате может возникнуть «разреженный файл», в котором много пустых промежутков между записями.

   В основном файле записи упорядочиваются по возрастанию ключа. Эффективность последовательного МД крайне низка: ля выборки нужной записи нужно просмотреть все предшествующие ей записи БД
2. ***Индексно-последовательный МД***. Предполагает наличие двух файлов, имеющих блочную структуру – основного и индексного. Индексный файл упорядочивается по первичному ключу. Каждый блок индексного файла содержит ссылку не на каждую запись основного, а на группу записей, хранимых в физическом блоке. В физическом блоке записи хранятся в виде цепочки в той же логической последовательности, что и индекс. Недостаток: при частой модификации файла перемещается большое количество записей.
3. ***Индексно-произвольный МД***. 2 файла: индексный и основной. Каждая запись индексного файла содержит ссылку на запись основного. В основном файле записи хранятся в произвольном порядке, а в индексном они упорядочиваются по ключу.
   - Для поиска записи используется бинарный поиск.
   - Удаление реализуется простановкой в найденной записи маркера удаления (например, “*”).
   - В случае изменения содержимого записи, а не ключа, запись остается на месте старой. Если же у записи изменяется ключ, то она удаляется и в БД добавляется запись с новым ключом.
4. ***Прямой МД***. Подразумевает однозначное соответствие ключа и адреса записи. Самый быстрый. Удаление, добавление и изменение происходит за одно обращение к файлу. Недостатки:
   4.1. реальный диапазон ключей может оказаться значительно больше диапазона адресов, что влечет ограничение на диапазон ключей.
   4.2. на практике файл может оказаться плохо заполненным.
5. ***Хэширование***. Отличие от прямого МД состоит в том, что для вычисления адреса используется более сложная функция. Недостатки те же, что и у прямого МД.
6. ***Инвертированный МД***. Организация записей в списке, с совпадающими значениями по одноименным атрибутам. Изменение и модификация требуют длительного времени, зато поиск весьма быстрый.
   УТОЧНИТЬ

№37. Оптимизация запросов. Общие стратегии оптимизации.

Предпосылки оптимизации

Время исполнения запросов может быть сокращено, если процессор ЯЗ перефразирует запрос перед его исполнением.

Опр. Такой процесс называется ***оптимизацией***.

Замечание. Этот перефразированный запрос не обязательно является оптимальным из всех возможных способов его реализации.

Оптимизация запросов равносильна оптимизации выражений РА и РИ. Важное значение имеет исследование выражений, которые содержат соединение или декартово произведение с последующими операциями селекции и проекции.

Общие стратегии оптимизации:

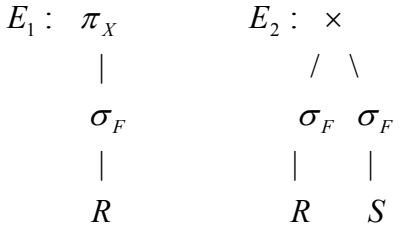
1) Выполнять операции селекции по возможности раньше.
2) Перед выполнением
   - операции × с последующей селекцией
   - соединением

   целесообразно предварительно сортировать или индексировать файл.
3) Искать общие подвыражения в выражении. Если результат общего подвыражения представляет собой небольшие отношения, который можно прочитать из внешней памяти за значительно меньшее время, чем требуется для его вычисления, целесообразно один раз вычислить это подвыражение.
4) Собирать в каскады проекции и селекции. Выполнять последовательность таких операций за один просмотр файла.
5) Комбинировать операции проекции с последующими или предшествующими двуместными операциями.
6) Вместо × с последующей селекцией выполнять операцию соединения.

№38. Оптимизация запросов. Алгебраические законы (10), используемые при организации запросов.

Процессор ЯЗ начинает обработку с построения дерева разбора для выражения РА.

Пример:



Сам язык запросов произвольный. Если запрос написан в виде выражения РИ, то он переводится в выражение РА.

Законы оптимизации.

1) Коммутативности для соединения и ×:
   $E_1 \times E_2 \equiv E_2 \times E_1$, $E_1 \triangleright\!\triangleleft_\theta E_2 \equiv E_2 \triangleright\!\triangleleft_\theta E_1$.
2) Ассоциативности для соединения и ×:
   $(E_1 * E_2) * E_3 \equiv E_1 * (E_2 * E_3)$;
3) Каскад проекций:
   Если $A_1 ... A_n \subseteq B_1 ... B_m$, то $\pi_{A_1 ... A_n}\left(\pi_{B_1 ... B_m}(R)\right) \equiv \pi_{A_1 ... A_n}(R)$.

Следствие: если проекция по всем атрибутам отношения, то она растворяется: $\pi_{R(\ )}(R) = R$.

4) Каскад селекций:
   $\sigma_{\mathrm{F}_1}\left(\sigma_{\mathrm{F}_2}(R)\right) \equiv \sigma_{\mathrm{F}_1 \wedge \mathrm{F}_2}(R)$;
5) Перестановка селекции и проекции:
   $\pi_{A_1 ... A_n}\left(\sigma_{\mathrm{F}}(R)\right) \equiv \sigma_{\mathrm{F}}\left(\pi_{A_1 ... A_n}(R)\right)$;.

   Следствие: Если $B_1 ... B_m \subseteq \mathrm{F}$, то $\pi_{A_1 ... A_n}\left(\sigma_{\mathrm{F}}(R)\right) \equiv \sigma_{\mathrm{F}}\left(\pi_{A_1 ... A_n B_1 ... B_m}(R)\right)$.
6) Перестановка селекции и ×.
   Если $\mathrm{F} = \mathrm{F}_1 \wedge \mathrm{F}_2$ и все атрибуты из F_1 являются атрибутами E_1, а из атрибуты из F_2 – атрибутами E_2, то
   $\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2)$;
   Следствие: Если F_1 содержит атрибуты из E_1, а F_2 из $E_1 \cup E_2$, то $\sigma_{\mathrm{F}_1 \wedge \mathrm{F}_2}(E_1 \times E_2) = \sigma_{\mathrm{F}_2}\left(\sigma_{\mathrm{F}_1}(E_1) \times E_2\right)$
7) Перестановка селекции с объединением:
   $\sigma_{\mathrm{F}}\left(E_1 \cup E_2\right) \equiv \sigma_{\mathrm{F}}\left(E_1\right) \cup \sigma_{\mathrm{F}}\left(E_2\right)$;
8) Перестановка селекции с разностью:
   $\sigma_{\mathrm{F}}\left(E_1 \setminus E_2\right) \equiv \sigma_{\mathrm{F}}\left(E_1\right) \setminus \sigma_{\mathrm{F}}\left(E_2\right)$;
9) Перестановка проекции с ×.
   Пусть E_1 и E_2 – реляционные выражения; $A_1, \ldots, A_n = \{\underbrace{B_1, \ldots, B_m}_{E_1}, \underbrace{C_1, \ldots, C_k}_{E_2}\}$, тогда:

   $\pi_{A_1 ... A_n}\left(E_1 \times E_2\right) \equiv \pi_{B_1 ... B_m}\left(E_1\right) \times \pi_{C_1 ... C_l}\left(E_2\right)$.
10) Перестановка проекции с объединением: $\pi_{A_1 ... A_n}\left(E_1 \cup E_2\right) \equiv \pi_{A_1 ... A_n}\left(E_1\right) \cup \pi_{A_1 ... A_n}\left(E_2\right)$.

№39. Оптимизация запросов. Алгоритм (6) оптимизации выражений РА.

Алгоритм оптимизации выражений РА.

Исходные данные – дерево разбора, представляющее выражения РА. Выходные – тоже дерево, но оптимизированное (программа, для вычисления выражения).

Метод:

1. Представить селекцию в виде каскада;
2. Перемещаем каждую селекцию как можно ниже по дереву;
3. Перемещаем каждую проекцию как можно ниже по дереву;
4. Комбинируем каждый каскад селекций и проекций в одиночную с предшествующими или последующими двухместными операторами.
5. Разбиваем внутренние узлы полученного в результате дерева на группы, в каждую из которых включается двухместный оператор и цепь потомков, помеченных унарными операторами, завершающуюся в листе.
6. Создаем программу из нескольких шагов для вычисления каждой группы в любом порядке, но так, чтобы никакая группа не вычислялась раньше групп – ее потомков.

   Шаги алгоритмов могут быть:

   1. выполнение одиночной селекции или проекции;
   2. выполнение $\times$, $\cup$, $\setminus$ множеств с последующими или предшествующими операциями селекции или проекции.

№40. Точная оптимизация для подмножества запросов, написанных на языке РИ.

Точная оптимизация для подмножества реляционных запросов.

Опр. ***Конъюнктивным*** называется запрос, удовлетворяющий критерию оптимизации.

Критерий оптимизации – минимальное число соединений и произведений.

В основе класса лежат запросы на языке РИ с переменными на доменах.

Формула запросов:

$$\left\{x_1,..,x_n \middle| (\exists y_1)..(\exists y_m)\left(\varphi(x_1,..,x_n,y_1,..,y_m,C_1,..,C_r)\right)\right\}$$

где

- $x_1,..,x_n$ – переменные на доменах или константы;
- $y_1,..,y_m$ – переменные на доменах;
- $C_1,..,C_r$ – константы;

φ имеет вид $R_1(t_1)\wedge..\wedge R_k(t_k)$, где R_i являются именами отношений, а t_i – кортежи, состоящие из x_i, y_i, C_i. Тот факт, что φ есть конъюнкция термов, дает название запросу.

Опр. Два запроса Q_1 и Q_2 ***эквивалентны*** (записывается $Q_1 \equiv Q_2$) т. и т.т., когда $Q_1 \subseteq Q_2 \wedge Q_2 \subseteq Q_1$.).

Опр. Запрос Q_1 ***содержится*** в Q_2, если существует отображение символов запроса Q_2 в Q_1.

Опр. ***Сверткой*** Q_2 в Q_1 называется отображение f символов Q_2 в Q_1 такое, что:

1) $f(x_i') = x_i$;
2) $\mathrm{f}(C_i') = C_i'$;
3) $f(y_i')$ является каким-либо из x_j, y_j, C_j;
4) Если $R(t)$ – какой-либо терм φ_2, то $R(f(t))$ – некоторый терм φ_1, где f, примененное к кортежу $t = t_1..t_j$, определяется равенством $f(t_1..t_j) = f(t_1),..,f(t_j)$.

Теорема: Запрос $Q_1 \subseteq Q_2$, если и только если найдется некоторая свертка Q_2 в Q_1.

№41. Параллельная обработка БД. Основные понятия и определения.

Основные понятия.

Опр. ***Транзакция*** – некоторая последовательность операций, неделимая с точки зрения действия на БД, не нарушающая целостность данных.

Опр. ***Элементы*** – это части БД, которые можно блокировать, при этом выполняемая транзакция может препятствовать доступу другой транзакции к этому элементу до тех пор, пока не разблокирует его.

Опр. ***Блок управления блокировками*** – компонент СУБД, который назначает, регистрирует блокировку.

В РБД могут выбираться большие элементы, включающие целые отношения, и малые – отдельные кортежи и компоненты кортежей, или часть отношения.

Выбор больших элементов сокращает накладные расходы системы по поддержанию блокировок, малых элементов дает возможность параллельного исполнения многих транзакций. Должен быть разумный компромисс.

Считается, что размер элемента выбран правильно, если средняя транзакция требует доступа к малому числу элементов. Так, в РБД если типичная транзакция читает или модифицирует **один кортеж**, то целесообразно **трактовать кортежи как элементы**. Если же типичная транзакция производит **соединение** двух или более отношений и тем самым **требует доступа ко всем кортежам** этих отношений, то в качестве элементов более **уместно выбрать отношения**.

Виды блокировок

1) ***Простая*** – не проводит разницы между операциями чтения / записи, нет доступа к элементам другой транзакции
2) ***Блокировка*** чтения или записи
    a. ***Блокировка по чтению*** – элемент становится readonly.
    b. ***Блокировка по записи*** – полная блокировка (нельзя ни читать, ни писать)

№42. Параллельная обработка БД. Бесконечные ожидания и тупики.

Бесконечные ожидания

Проблема бесконечного ожидания может возникнуть в любой обстановке, предусматривающей параллельное исполнение процессов. Пусть транзакция T_1 проводит работу с элементом A, и доступ к элементу A заблокирован этой транзакцией; T_2 находится в состоянии ожидания; транзакция T_3 также запрашивает блокировку A, и этот запрос исполняется раньше, чем запрос T_2. Далее, в то время, когда блокировку установила транзакция T_3, ее запрашивает T_4, чьё требование удовлетворяется после разблокирования A транзакцией T_3 и т.д. Не исключено, что транзакция T_2 будет пребывать в состоянии бесконечного ожидания.

Решение. Регистрировать все запросы и удовлетворять их в порядке очерёдности.

Тупики

Опр. ***Тупик*** – ситуация, в которую попадает транзакция из некоторого множества S, содержащего не менее двух транзакций, в которых каждый из них желает блокировать элемент, блокированный другой транзакцией в данный момент.

Решение.

1) Потребовать, чтобы каждая транзакция единовременно запрашивала все нужные ей блокировки. Система либо полностью удовлетворяет такой запрос, либо не предоставляет вообще никаких блокировок и заставляет данную транзакцию ждать до освобождение всех необходимых ей элементов.
2) Производится некоторое линейное упорядочивание элементов и все транзакции должны запрашивать блокировки в этом порядке. Для транзакции, заблокировавшей некоторый элемент, гарантируется возможность заблокировать и все следующие за ним элементы.
3) Ничего не предпринимать для предотвращения тупиков, а выполнять проверку на наличие ситуации тупика. Для этого строится граф,
   - вершины графа – транзакции,
   - дуга $T_i \to T_j$ означает, что транзакция T_j ожидает выполнения своего запроса на блокирование элемента, заблокированного в данный момент транзакцией T_i.
   - Если цикла нет, то не существует и тупиков.
   - Если цикл есть, то надо
     - сделать рестарт хотя бы одной из попавших в тупик транзакций (удалить дугу) или
     - сделать откат транзакции (удалить вершину).

№43. Протоколы и расписания. Простая модель транзакции.

Протоколы и расписание.

Каждая транзакция состоит из элементарных шагов (блокировка, чтение записи и т.д.).

Опр. ***Расписанием*** совокупности транзакций называется порядок выполнения элементарных шагов транзакций.

Опр. Расписание считается ***последовательным***, если каждая транзакция выполняет свои шаги, а после этого следующая транзакция работает с тем или иным элементом.

Опр. Расписание считается ***сериализуемым***, если оно эквивалентно эквивалентно некоторому последовательному расписанию.

Если есть совокупность транзакций, которая не исключает тупика, то расписание не сериализуемо.

Для сериализуемости используются протоколы, которые накладывают некоторые ограничения на последовательность выполнения элементарных шагов. Например, протоколом является исключающая тупики стратегия запрашивания блокировок элементов в некотором фиксированном порядке.

Простая модель транзакции.

Опр. ***Простая модель транзакции*** рассматривает транзакцию как последовательность операторов блокирования и разблокирования. Каждый блокированный элемент впоследствии должен быть разблокирован. Будем считать, что при между шагом LOCK A и UNLOCK A транзакция осуществляет изменение значения A.

Замечание. Простая модель транзакции позволяет говорить о сериализуемости.

Поставим каждой паре шагов LOCK и UNLOCK однозначную функцию *f*. Любой элемент A может быть заблокирован и разблокирован большое число раз, поэтому будем рассматривать последовательность функций $f_1(f_2(...f_n(A_0)))$, где A_0 – начальное значение A до выполнения любых транзакций.

Пример:

	A	B	C
T_1: LOCK A			
T_2: LOCK B	…		
T_2: LOCK C	…		
T_2: UNLOCK B	$f_1(B)$		
T_1: LOCK B			
T_1: UNLOCK A	$f_2(A)$		
T_2: LOCK A			
T_2: UNLOCK C			$f_3(C)$
T_2: UNLOCK A	$f_4(f_2(A))$		
T_3: LOCK A			
T_3: LOCK C			
T_4: UNLOCK B		$f_5(f_1(B))$	
T_3: UNLOCK C			$f_6(f_3(C))$
T_3: UNLOCK A	$f_7(f_4(f_2(A)))$		

Эта модель используется в алгоритмах проверки сериализуемости расписания.

№44. Алгоритм проверки сериализуемости расписания.

Метод, позволяющий определить сериализуемость расписания.

1) Строится ориентированный граф G, который называется ***графом предшествований***, узлы которого соответствуют транзакциям.
2) Рассматривается некоторое множенство $S = a_1, a_2, ..., a_n$, где
   a. $a_i = \text{LOCK(A)}$ либо
   b. $a_i = \text{UNLOCK(A)}$
3) Строится дуга от T_i к T_j в том случае, когда T_i – транзакция выполняет действие UNLOCK(A), а T_j – LOCK(A).
4) Если в графе ***есть циклы***, то ***расписание не сериализуемо***, т.е. есть тупики.
5) Если граф ***ацикличен***, то с помощью ***топологической сортировки*** определяется *порядок выполнения транзакций*.

Утверждение. ***Расписание сериализуемо***, если транзакция выполняет все свои команды блокирования и разблокирования, и после этого выполняется другая транзакция.

Опр.Транзакции, в которых все операции блокирования предшествуют всем операциям разблокирования, называются ***двухфазными***. Первая фаза называется фазой блокировки, а вторая – фазой разблокирования.

Теорема: Любое расписание двухфазных транзакций является сериализуемым.

Модель с блокировками для чтения и записи.

Если элемент блокирован для записи, то на его значение воздействует функция f. Если для чтения – значение элемента не изменяется.

Метод проверки сериализуемости расписания:

Строится граф предшествования. Узлы – транзакции. Дуги определяются следующим образом:

1) Если T_i блокирует элемент A для чтения, а T_j, следующая за T_i, – для записи, то строится дуга из T_i в T_j.
2) Если T_i блокирует элемент A для записи, а T_j, следующая за T_i, – для записи, то строится дуга из T_i в T_j
3) Если T_m – блокирует A для чтения после того, как T_i снимет свою блокировку для записи, но перед тем, как T_j установит свою блокировку для записи. Если T_j не существует, то T_m – любая транзакция, которая блокирует A для чтения после того, как T_i разблокирует его. Тогда строим дугу из T_i в T_m.

Если граф имеет циклы, то расписание не сериализуемо.

№45. Экспертные системы. Основные понятия и определения.

Опр. ***Редактор*** – позволяет редактировать БЗ (программа)

Опр. ***Эксперт*** – это высококвалифицированный специалист в данной предметной области

Опр. ***Инженер по знаниям*** – специалист в области искуственного интеллекта, который осуществляет перевод знаний эксперта в некоторую модель, в соответствии с которой правила помещаются в БЗ.

Опр. ***ЭС*** – это система, которая относится к классу интеллектуальных систем, направленная на тиражирование опыта высококвалифицированных специалистов в области, где качество принятия решения зависит от уровня квалификации эксперта.

№46. Характеристики экспертных систем. Виды ЭС.

Классификация экспертных систем.

Классификация по решаемой задаче

1. Интерпретация данных

Интерпретация данных – следует понимать процесс определения смысла данных, результаты которого должны быть согласованным и корректными.

Для таких систем используется многоспектный анализ.

Пример: определение с помощью космических или аэросредств различных типов судов.

2. Диагностика

Она представляет собой соотношение объектов некоторых объектов некоторых классов объектов или обнаружение неисправности в системе.

Под неисправностью понимают отклонение от нормы. Такая трактовка позволяет распознавать неисправности в механических системах и определение заболеваний в медицине и различных природных аномалий.

Пример: определение неисправности в автомобиле.

3. Мониторный

Происходит непрерывное интерпретирование данных и сигнализация о выходе параметров за допустимые пределы.

Существует две проблемы:

1) Пропустить проблему
2) Инверсная проблема (ложное срабатывание)

Пример: на АТС, гидростанциях

4. Проектирование

Состоит в определении спецификации на создание объекта с определенным свойствами. В таких ЭС не только применяется происходное решение, но и существует система объяс, которая позволяет говорить о целесообразности выбора того или иного решения.

Пример: проектирование интегральных схем, электрической цепи.

5. Прогнозирование.

Позволяет представить последствия некоторых событий явлений на основе имеющихся данных. В таких ЭС используются параметричесая динамическая модель, параметры которой подгоняются под конкретную ситуацию. Следствие, вытекающее из этой модели составляет основу прогнозов с использованием вероятностных оценок.

Пример: системы предсказания погоды.

6. Планирование

Понимают нахождение планов действий, относящихся к объектам и выполняющих некоторые функции.

Пример: системы управления роботами.

7. Обучение.

Используется компьютер для обучения некоторому предмету или дисциплине. Такие ЭС выявляют ошибки обучаемого, помогают выбрать правильное решение от диалога с обучаемым, такие системы позволяют определить уровень подготовки обучаемого и те знания, которые необходимы для повышения уровня.

8. Управление

Понимается функция организованной системы, поддерживающая определенный режим деятельности. Такие ЭС управляют поведением сложных систем в соответствии с определенной спецификацией.

Пример: ЭС, используемые в газовых котельнях.

9. Поддержка принятия решения

Такая система представляет собой совокупность процедур, позволяющая индивидууму прининимаемого решения облегчить процесс принятия решения за счет предоставления информации и рекомендаций.

Пример: Выбор страхового общества, выбор стратегии в критических ситуациях.

Данные классы можно разделить на задачи анализа и задачи синтеза.

В задачах анализа множество решений может быть перечислено и включено в систему

В задачах синтеза множество решений потециально не ограничено и общее решение находится через решение подпроблем.

Опр. ***Задача анализа*** – это система интерпретации, диагностики данных, решения проблем.

Опр.*Задача синтеза* – проектирование, управление, планирование.

Другие – это их комбинация, то есть комбинированные системы.

Классификация по средствам реального времени

1. Статические ЭС
2. Квазидинамические ЭС
3. Динамические ЭС

Опр. ***Статические ЭС*** – системы, в которых БЗ и интерпретируемые данные не изменяются во времени (поиск неисправности в автомобиле).

Опр. ***Квазидинамические ЭС*** – ситуация, которая изменяется с некоторым интервалом времени (микробиологические ЭС).

Опр. ***Динамические ЭС*** – системы, в которых осуществляется непрерывная интерпретация данных (задачи мониторинга).

Классификация по типу ЭВМ.

1. Использование супеЭВМ (суперкомпьютер CRAY, многопроцессорная система Эльбрус)
2. Использование символических процессоров и рабочих станций (Sun, Silicon)
3. Применение ЭВМ средней производительности
4. Использование ПК

Классификация по степени интеграции.

1. Автономные системы
2. Гибридные ЭС.

Опр. ***Автономные системы*** – не требуется привлекать периодические методы обработки данных (не решают задач моделирования).

Опр. ***Гибридные системы*** – это программный комплекс, агрегирующий стандартизированные пакеты прикладных программ (программа математической статистики).

№47. Представление знаний. Продукционные модели.

Классификация моделей представления данных

1. Продукционные модели
2. Семантические сети
3. Фреймы
4. Формальные и логические модели

Продукционная модель основана на правилах вида:

if <условие> **then** <действие>

Опр. <условие> называется атцетендентом,

Опр. <действие> называется консиквентом.

Есть база знаний, в которой хранятся правила. В соответствии с правилом отыскивается условие и производится соответствующее действие. В качестве консиквента может быть новое правило.

В данной модели используется прямой вывод, то есть от цели идут к конкретным фактам или действиям. Справедливо и обратное: имея факты, подтверждаем цель.

Данная модель простая, наглядная, у нее высокая модульность. В настоящее время большинство ЭС основано именно на ней.

№48. Представление знаний в ЭС. Семантические сети.

Опр. ***Семантическая сеть*** представляет собой граф, вершины которого – понятия, а дуги – отношения. Понятия – абстрактный или конкретный объект. Отношения – связи между объектами.

Условно модель можно разделить по признакам:

1) Однородные СС (один тип отношений)
2) Неоднородные (по числу типов отношений делят на бинарные и т.д.)

Выделяют **отношения** для данной модели:

1. Элемент класса
2. Иметь свойства
3. Значения свойства
4. Часть – целое
5. Функциональные
6. Количественные
7. Пространственные
8. Временные
9. Логические
10. Примеры элементов класса

Преимущества модели:

– Соответствует концептуальной модели организации памяти человека.

Недостатки:

– Трудность выполнения запросов, так как в этом случае приходится выделять некоторый фрагмент сети, который соответствует подсети.

Существуют определенные языки, которые позволяют реализовать эти запросы.

№49. Представление знаний. Фреймы.

Опр. ***Фрейм*** – это абстрактный образ для представления стереотипа явления объекта, понятия.

Представление:

(Имя фрейма:)
(имя 1-го слота: значение слота)
…
(имя n-го слота: значение n-го слота)

Значение может быть

- функцией;
- именем другого фрейма, таким образом могут образовываться сети фреймов.

Преимущества модели:

- Наибольшее соответствие концептуальной организации памяти человека.

Недостатки:

- Более сложная реализация

№50. Этапы проектирования ЭС.

1) Выбор проблемы
2) Разработка прототипа
3) Доработка
4) Оценка
5) Стыковка
6) Поддержка

Этап 1.

Определение проблемной области;

- наличие эксперта;
- назначение коллектива разработчиков;
- анализ прибыли и расходов;
- подготовка плана разработки.

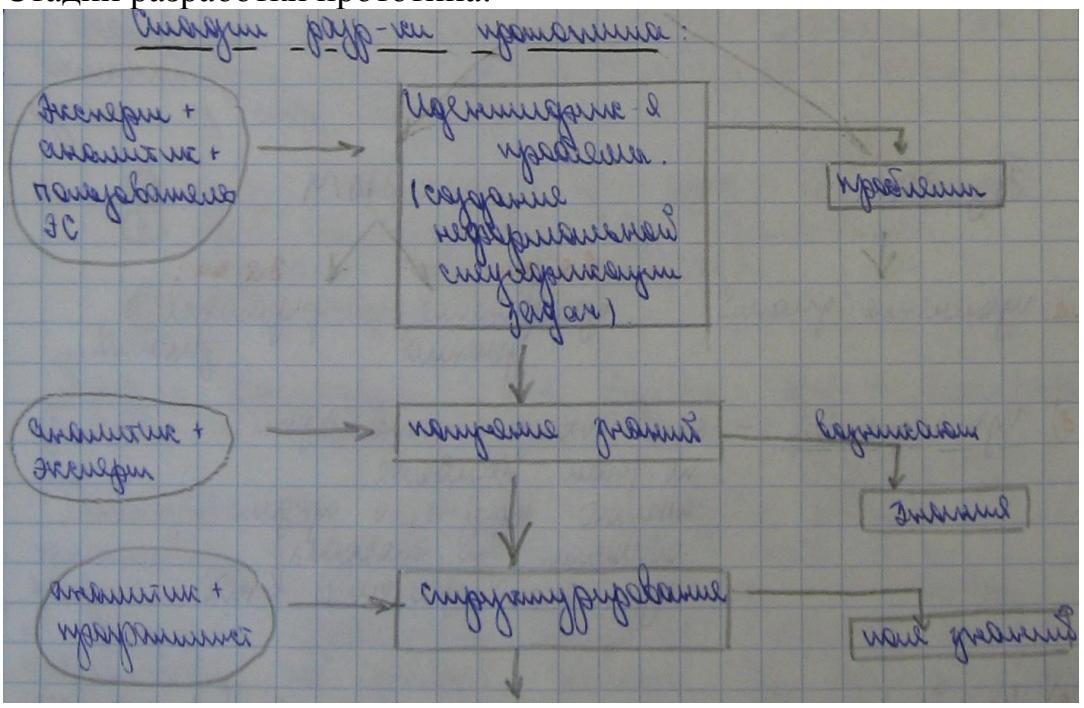
1. Задача решаема;
2. На рынке имеются необходимы программные средства;
3. Имеется подходящий эксперт;
4. Критерия производителя приемлемы для заказчика;
5. Приемлемые затраты и сроки окупаемости.

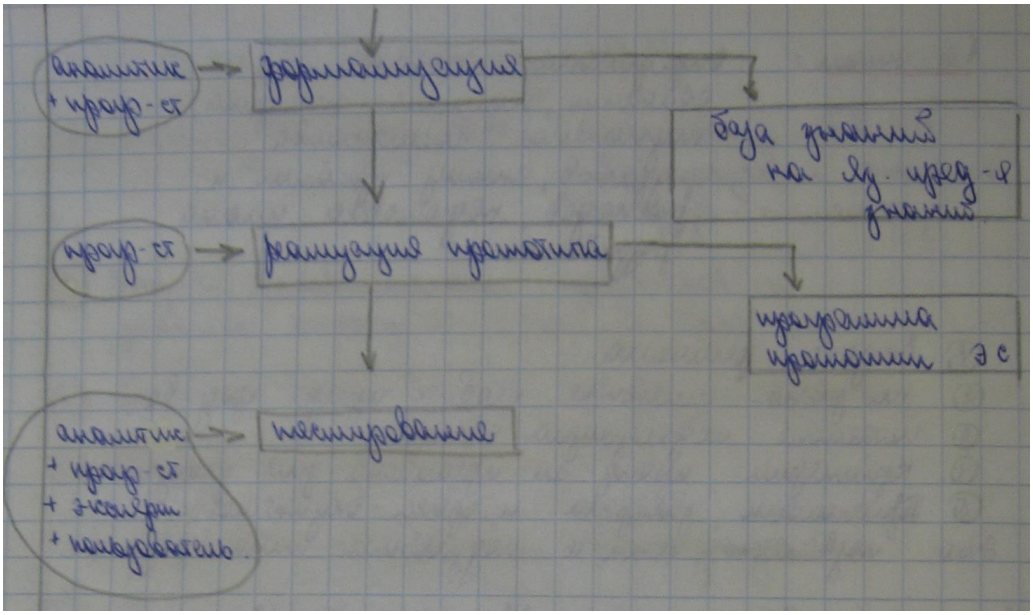
Это позволяет составить подробный план разработки.

Этап 2.

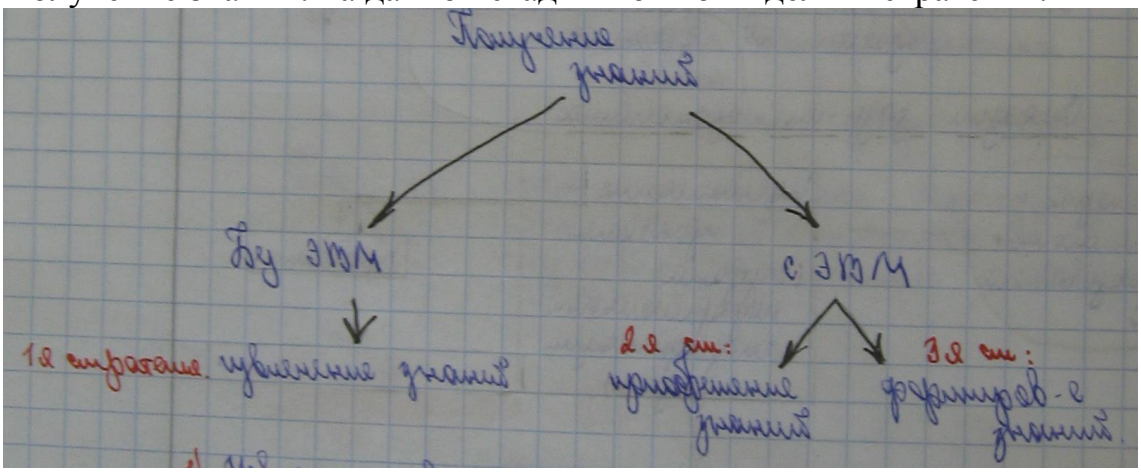
Опр. ***Прототип ЭС*** – ЭС, проектируемая для проверки правильности кодирования фактов, рассуждений или связей. Объем прототипа – несколько действий, правил и фреймов.

Стадии разработки прототипа:





Получение знаний: на данной стадии можно выделить стратегии:



1) ***Извлечение знаний*** – это получение нинженером по знаниям наиболее полного представления о предметной области и способах принятия решений в ней.
2) ***Приобретение знаний*** – это процесс дополнения БЗ экспертом с использованием специальных программных средств.
3) ***Формирование знаний*** – процесс анализа данных и выявления скрытых закономерностей с использованием специального математического аппарата и программных средств.

Этап 3.

Опр. ***Структурирование*** – это обработка ПО и представление ее в виде графа, таблицы, текста.

На данном этапе выделяются объекты, их атрибуты, определяется их терминология, отношение между объектами структура вход и выходной информации. Стратегия принятия решеня, ограничения стратегий.

Опр. ***Формализация данных*** – построение формального представления ПО на основе выбранной модели ЯПЗ (фреймы, продукционные модели, сети и т.д.).

Опр. ***Реализация прототипа*** – создаение прототипа ЭС, включающего БЗ и остальные блоки одним из следующих способов:

а) Программирование на традиционных языках программирования (C++, Pascal)

б) Программирование на специальных языках, применяемых в задачах ИИ;

в) Использование пустых ЭС.

Опр. ***Программная реализация*** – это разработка программного комплекса, обеспечивающая жизнеспособность системы в целом.

Опр. ***Тестирование*** – оценка и проверка работы прототипа с целью приведения его в соответствие с запросами пользователей. Выявляются ошибки в подходе и реализации прототипа, вырабатываются рекомендации подготовки по доводке системы до производственного варианта.

Этап 4.

Оценка – оценка эффективности ЭС: достигнуты ли задачи, поставленные на выше описанных этапах.

Система может быть оценена пользователем, экспертом и разработчиком:

- ***Критерий пользователя***: дает ли пояснения, удобный ли интерфейс.
- ***Критерий эксперта***: проверка решения.
- ***Критерий разработчиков***: сложность создания, возможность внесения изменений в данную систему.

Этап 5.

Все пользователи данной системы оценивают ее на выполнение функций в интересах разных категорий. Формируется интеграция ЭС с другими программными пакетами.

Этап 6.

Поддержка – в связи с изменениями, происходящими в ПО возможно внесение изменений в ЭС.