СМОЛЕНСКИЙ ГУМАНИТАРНЫЙ УНИВЕРСИТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Г.Н.МАКАРОВ

ТЕОРИЯ ЭКОНОМИЧЕСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебное пособие для студентов ВУЗов,
обучающихся заочно по специальности
«ПРИКЛАДНАЯ ИНФОРМАТИКА В ЭКОНОМИКЕ»

СМОЛЕНСК 2008 г.

ОГЛАВЛЕНИЕ

1.Теория курса экономических информационных систем

Предметом изучения курса теории экономических информационных систем является экономическая информация в планировании, учете и анализе.

Цель курса - уметь представлять информацию о предметной области с помощью различных моделей данных.

Структура курса

1. Основные понятия экономической информационной системы (ЭИС):
Компоненты ЭИС;
Единицы информации;
Жизненный цикл ЭИС;
Классификация ЭИС;

2. Модели данных.

- 2.1 Информационное пространство и понятие структурированности информационного пространства;
- 2.2 Элементы теории отношений;
- 2.3 Иерархическая модель данных;
- 2.4 Сетевая модель данных;
- 2.5 Реляционная модель данных;
- 2.6 Модели представления знаний;

3. Методы организации данных в памяти ЭВМ

- 3.1 Сортировка;
- 3.2 Поиск;
- 3.3 Связанные списки;

4. Приложения, обучающие программные продукты по курсу теории экономической информационной системы .

2.Предметная область теории экономической информационной системы

Любая экономическая система состоит из совокупности связанных ресурсов и процессов. К ресурсам относятся: рабочие, служащие, сырье и материалы, деньги, средства производства. Процесс – это преобразование одного набора ресурсов в другой набор ресурсов.

Взаимосвязанные ресурсы и процессы образуют информационное пространство соответствующей предметной области. Информация о предметной области хранится и обрабатывается ЭИС.

Итак, информационное пространство – это множество всех компонентов предметной области (объекта) независимо от способов и средств отображения этих компонентов.

Один из важнейших характеристик информационного пространства является степень его структурированности.

Под структурированностью понимается такое свойство информационного пространства, при котором его компоненты и взаимосвязи между ними выраженные в явном виде.

Различают пять степеней структурированности информационного пространства:

1. ***Неструктурированное информационное пространство*** (например, разговорная речь о компонентах информационного пространства);
2. ***Слабоструктурированное информационное пространство***, т.е. структурированная только часть компонентов информационного пространства в письменной форме;
3. ***Структурированное информационное пространство***, где информация документирована, понятия кодированы.
4. ***Формализовано – структурированное информационное пространство***, т.е. определение связи между компонентами информационного пространства и алгоритм обработки элементов данных (например, сортировка данных, поиск, размещение данных, вызов данных).
5. ***Машинно-структурированное информационное пространство***, т.е. вся информация, реализована в виде базы данных в машинной проблеме.

Таким образом, степень структурированности информационного пространства предметной области характеризует упорядоченность информации об объекте.

Последовательность перехода от одного вида структурированности информационного пространства через промежуточные модели к системе машинной обработки данных является предметом изучения курса теории экономической информационной системы. При этом важной особенностью процесса последовательности перехода от одного вида структурированности к другому виду является соблюдение синтаксической и семантической непротиворечивости процессов переходов, обеспечивающее полноту, корректность и адекватность информационного пространства реальным условиям.

Переход от одного вида структурированности информационного пространства, формирующего информационную модель объекта к последующим моделям, образуют этапы создания системы машинной обработки данных по запросам со стороны приложений.

Эти этапы определяют:

1. Этап машинной интерпретации информационной модели, содержащий визуальные компоненты информационного пространства, т.е. объекты и методы их поведения;
2. Этап сопряжения машинной модели со средствами управления данными (т.е. программно-алгоритмическая модель со средствами управления данными).
3. Система обработки данных по запросам приложения.

Переход от информационного пространства предметной области к системе обработки данных, регламентирует три уровня представления информации, а именно:

1. Внешний уровень;
2. Концептуальный;
3. Внутренний;

Внешний уровень представления информации- это описание информационных потребностей конечного пользователя-(т.е. первый вид структурированности информационного пространства).
Концептуальный уровень-это описание компонентов информационного пространства на уровне понятий ЭИС, т.е. описание представлений пользователя в абстрактной форме, выраженной в определении структуры информационного пространства, описание методов контроля информации, доступа и ограничений на доступы к информации и.т.д.
Внутренний уровень-реализация концептуального уровня в вид физической модели, реализуемой в памяти ЭВМ.

3.Основные понятия экономических информационных систем

Информационная система и ее назначение

В основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются информационные системы (ИС).

Автоматизированными называют ИС, в которых применяют технические средства, в частности ЭВМ. В дальнейшем под ИС будем понимать автоматизированную ИС.

По области применения ИС будем рассматривать обработку информации для объектов экономического характера. Исходя из конкретизации использования ИС, введем понятие «экономическая информационная система» (ЭИС). По целевой функции ЭИС можно условно разделить на следующие основные категории: управляющие, информационно-справочные, поддержки принятия решений.

Детальное изучение ЭИС включает понятия «экономическая информация» и «система».

Под экономической информацией будем понимать набор сведений, принятых и понятых конечным потребителям, имеющих отношение к экономике, т.е. к процессам производства, распределения, обмена и

потребления материальных благ, включая финансовые процессы. Иными словами, экономическая информация характеризует производственные отношения в обществе.

Понятие системы охватывает комплекс взаимосвязанных элементов, действующих как единое целое в интересах достижения поставленных целей. Каждая система включает в себя следующие компоненты:

1. структуру системы – множество элементов и взаимосвязей между ними;
2. функции каждого элемента системы (управленческие функции, функции обработки данных и т.д.);
3. вход и выход каждого элемента и системы в целом;
4. цели и ограничения системы и ее отдельных элементов (финансовые ограничения, материальные и т.д.)

Каждая система обладает свойствами делимости и целостности. Делимость означает, что систему можно представлять состоящей из относительно самостоятельных частей – подсистем, каждая из которых может рассматриваться как система.

Свойство целостности указывает согласованность цели функционирования всей системы с целями функционирования ее подсистем и элементов.

Таким образом, система – это сложный объект с определенным образом упорядоченной внутренней структурой, обеспечивающей связность составных частей в единое целое. При этом система, как правило, имеет больше свойств, чем составляющих ее элементов.

Под экономической информационной системой будем понимать взаимосвязанную совокупность технических средств (средств вычислительной техники), методов сбора, хранения, обработки и распространения информации о деятельности некоторого экономического объекта реального мира в интересах достижения поставленных целей пользователями системы.

Экономическая информационная система, как правило, создается для конкретного экономического объекта и должна в определенной мере фиксировать взаимосвязи элементов объекта.

В целом экономическую информационную систему следует воспринимать как человека – компьютерную систему обработки информации, ибо современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации персонального компьютера (ПК) или большой ЭВМ.

Эффективность функционирования ЭИС во многом зависит от ее архитектуры. В настоящее время перспективой является архитектура клиент-сервер. В достаточно распространенном варианте она предполагает наличие компьютерной сети с распределенной базой данных и приложений, реализуемых на персональных компьютерах пользователей.

Локальный вариант ЭИС – это приложение, решающее экономическую задачу, и плюс база данных с источником входных данных для приложения на компьютере пользователя.
Современные системы управления базами данных (СУБД) позволяют решать широкий круг задач по работе с базами данных без разработки приложений. Тем не менее есть случаи, когда целесообразно разработать программу-приложение, в частности средствами визуального программирования. Разработанное приложение обычно состоит из одного или нескольких файлов операционной системы.
Если основным файлом приложения является исполняемый файл (например, exe-файл), то это приложение, скорее всего, является независимым приложением, которое выполняется автономно от среды СУБД.
Во многих случаях приложение не может исполняться без среды СУБД. Для этого обычно, например, с помощью средств визуального программирования Delphi привлекают процессор баз данных BDE (Borland Database Engine), играющий роль ядра СУБД. Процессор базы данных анализирует содержимое файлов приложения и автоматически строит необходимые исполняемые машинные команды.

При работе приложения пользователя с базой данных над ее содержимым выполняются следующие основные операции: выбор, добавление, замена и удаление данных. Цикл взаимодействия приложения пользователя с БД для операции выбор можно разделить на следующие основные этапы:

1. пользователь через терминал формулирует запрос на некоторые данные из БД;
2. приложение на программном уровне средствами языка манипулирования данными формулирует через программный интерфейс запрос, с которым обращается к СУБД;
3. используя свои системные управляющие блоки и таблицы, СУБД с помощью словаря данных определяет местоположение требуемых данных и обращается за ними к операционной системе (ОС);
4. программы методов доступа файловой системы ОС считывают из БД на внешнем носителе искомые данные и помещают их в системные буферы СУБД;
5. преобразуя полученные данные к требуемому формату, СУБД пересылает их в соответствующую область программы-приложения и сигнализирует о завершении операции, например, кодом возврата;
6. результаты выбора данных из базы приложения отображаются на терминале пользователя.

В случае работы пользователя в диалоговом режиме с СУБД (без приложения) цикл взаимодействия пользователя с БД упрощается. Его можно представить следующими этапами:

1. пользователь терминала через терминальный интерфейс формулирует на языке запросов СУБД, требование на выборку некоторых данных из БД;
2. СУБД определяет местоположение требуемых данных и обращается за ними к ОС, которая считывает из БД на внешнем носителе искомые данные и помещает их в системный буфер СУБД;
3. информация из системного буфера преобразуется к требуемому формату, после чего отображается через терминальный интерфейс на терминале пользователя.

Если компьютер и ОС поддерживает многопользовательский режим работы, то в такой вычислительной машине может функционировать многопользовательская СУБД.

4.Структура информационной системы

Структура любой ИС, независимо от сферы применения, может быть представлена совокупностью обеспечивающих ее подсистем. Среди обеспечивающих подсистем обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечения.

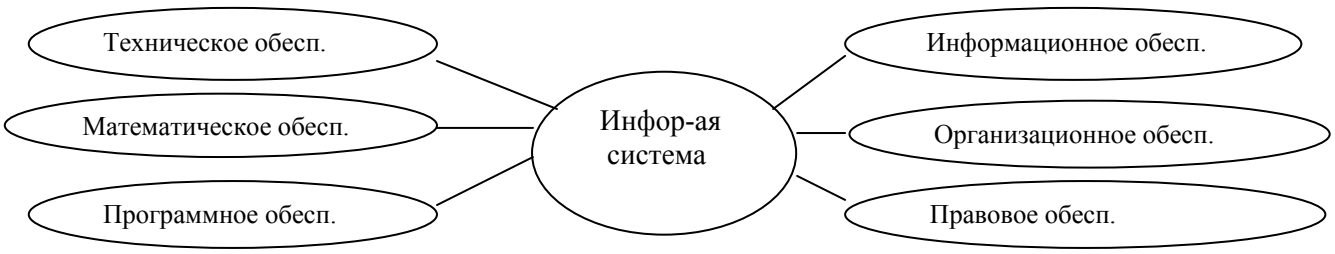


Рис.1. Структура ИС

1. Информационное обеспечение **– это совокупность методов и средств по размещению и организации информации, включая методологическое построение баз данных.**
2. **Программное обеспечение** – это совокупность программных средств для создания и эксплуатации системы обработки данных, включая операционную систему, систему управления базой данных, сервисные программы, трансляторы языков программирования и прикладные программные продукты.
3. **Техническое обеспечение** – комплекс технических средств, применяемых для функционирования систем обработки данных (компьютеры, устройства сбора, накопления и вывода информации, средства телекоммуникации и связи).

4. **Математическое обеспечение** – совокупность математических методов, алгоритмов для реализации целей и задач.
5. **Организационное обеспечение** – совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой.
6. **Правовое обеспечение** – совокупность правовых норм, определяющих юридический статус ИС, права и обязанности персонала.

5. Компоненты экономической информационной системы

Практически все ИС включают один и тот же набор компонентов, а именно:

- функциональные компоненты;
- компоненты системы обработки данных;
- организационные компоненты.

Функциональные компоненты – это задачи предметной области, методы и алгоритмы их решения. Задача включает набор исходных данных в виде атрибутов, отношения, показателей, базы данных.

Компоненты системы обработки данных – это совокупность систем, обеспечивающих функционирование всех структур ИС.

Организационные компоненты – состав и структура организаций, обеспечивающих эффективное функционирование экономического объекта.

Информационные компоненты экономических информационных систем.

Экономическая информация представляет собой специфический вид социальной знаковой информации и имеет отношение к науке о знаках и знаковых систем семиотики.

Экономическая информация существует в документах.

Документ – это основной носитель информации, имеющий юридическую силу.

Информация в экономике с точки зрения управления имеет три аспекта:

1. Синтаксический
2. Семантический(смысловой)
3. Прагматический

Синтаксический аспект, с точки зрения семиотики(науки о знаках и знаковых системах) – это правило образования и употребления слов естественного языка об показателях объекта, как предмета обсуждения.

Семантический аспект – это смысловое содержание тех или иных понятий, выраженных в различных контекстах(т.е. в виде знаков, знаковых систем или слов).

Прагматический аспект – это уровень информации, отражающий полезность сообщения.

Каждый реальный объект имеет ряд характерных для него свойств, которые выражены в параметрах, характеристиках, признаках.

Свойства объекта или сущности отображаются с помощью величин, являющимися элементарными единицами информации, которые принято называть ***реквизитами.***

Реквизиту присущи два существенные свойства:

1. Отдельный реквизит может входить в состав различных составных единиц информации(CEU).
2. Отдельно взятый реквизит не может полностью характеризовать объект.

Таким образом, ***реквизит*** – это информационное отображение отдельного свойства объекта реального мира.

Реквизит характеризуется – областью определения (например, температура, цена, фамилия, имя и т.д.);

- Реквизит имеет длину, т.е. число символов, образующих его значение;
- Реквизит характеризуется типом, а именно: тип числовой, текстовый, логический.

Реквизиты подразделяются на реквизиты признаки – т.е. определяют место действия, и реквизиты основания, т.е. определяют меру действия.

Синонимы реквизита: поле, атрибут, признак, термин.

Второй уровень единиц информации – показатель.

Показатель – это минимальная по информационному содержанию единица, но достаточная для образования документа, т.е. частный случай совокупной единицы информации(CEU).

Для исчерпывающей характеристики объекта необходима некоторая совокупность реквизитов, описывающих количественные и качественные свойства объекта. Эти исчерпывающая характеристика получила название ***составная единица информации***(CEU).

Совокупная единица информации – это объединенная совокупность разных реквизитов, связанных между собой некоторыми отношениями.

Последний, в порядке возрастания синтаксической сложности, уровень единицы информации – база данных.

База данных – это единица информации, задающая информационное отображение множества разнородных свойств объекта.

6.Информационные отношения

Всякая экономическая система имеет дело с множеством объектов или элементов. Для того чтобы описать такую систему, нужны не только сами эти элементы, но и отношения между ними.

Отношения, в котором участвуют два объекта, называют бинарными. Отношения между тремя объектами называют тернарными. Если отношение касается одного объекта, то оно называется унарным.

Пример. Известно, что кровь каждого человека одной из четырёх групп. Причём кровь первой группы можно переливать любому человеку, кровь второй и третьей группы можно переливать людям с той же группой крови, либо людям с четвёртой группой и, наконец, кровь четвёртой группы может переливаться только её обладателям. Отношение совместимости групп

крови человека можно записать следующим образом. **X можно переливать Y**

Характер этого отношения раскрывает приведённая ниже на рисунке схема в виде решетки

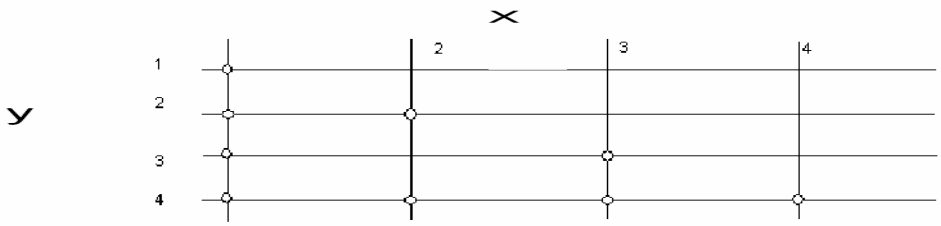


Рис.2. Схема совместимости групп крови человека.

Это же отношение полностью характеризуется числовыми парами.
(1,1),(1,2),(1,3),(1,4),(2,2),(2,3),(2,4),(3,3),(3,4),(4,4).

Из приведённого примера напрашивается обобщённая форма записи для любого бинарного отношения “XRY.”, где R- множество, заданное перечислением упорядоченных пар элементов или решёткой.

СВОЙСТВА ОТНОШЕНИЙж1.Рефлективность. Некоторый элемент, а отношения R находится в отношении R к самому себе, т.е. aRa,т.е. отношение R выполняется при любом а. Например, каждая прямая а считается параллельной самой себе.

2.Симметричность. Для некоторой пары элементов а и b имеет место два отношения:

aRb и bRa.

Пример. отношение параллельности, отношение перпендикулярности и отношение равенства.

3.Транзитивность. Из двух условий aRb и Rc вытекает aRc**.**

Пример, отношение параллельных прямых, отношение равенства (a=b), отношение а больше чем b (a>b);

Отношение эквивалентности. Отношение называется эквивалентностью если соблюдаются этого отношения все три, указанные свойства отношений, т.е. рефлексивность, симметричность, транзитивность. Отношение называется толерантностью, если соблюдаются свойства рефлексивности и симметричности. Отношение называется квазипорядком, если соблюдаются свойства рефлективности и транзитивности.

Отношение упорядоченности.

Для любых двух несовпадающих элементов а и b всегда выполняется одно из двух соотношений a>b или b<a, т.е. если все элементы множества сравнимы между собой и образуют отношение, то такое отношение называют упорядоченным.)

Пример. Слова в словарях упорядочиваются лексиграфически, т.е. в алфавитном порядке; ученики могут быть упорядочены по росту. Процесс упорядочения называется сортировкой. В процессе сортировки осуществляется сравнение пар значений признаков. Минимальное возможное число таких сравнений $N=M\log_2 M$, где M-число сортируемых признаков.

Признаки, значения которых упорядочиваются в пределах некоторого участка массива, называются ключами сортировки.

7.Логические отношения

Это отношение причинно-следственной связи совокупных единиц и информации (СЕИ). Две СЕИ называются логическими эквивалентами, если одна из них может заменить другую, причём замена не приводит к потере информации.

Пример. Задана таблица истинности для функции двух аргументов

X_1	X_2	(X_1,X_2)
0	0	0
0	1	1
1	0	1
1	1	0

Составляя аналитическое выражение получим две логические эквивалентные функции.

$f(x_1,x_2)=NOT(x_1)\&x_2 U x_1\&NOT(x_2)$ и

$f(x_1,x_2)=(x_1 U x_2)\&(NOT(x_1)UNOT(x_2))$

где знак &-логической функции “И”;U-логической функции “ИЛИ”.Символ NOT(.)-логическая функция “НЕ” или инверсия.

8.Множественные отношения

Это бинарные отношения, определённые на единицах информации.

1.Отношение равенства. Две СЕИ S_i и S_J образуют элемент отношения равенства если множества их значений равны.

2.Отношение включения. Две СЕИ S_i и S_j образуют элемент отношения включения, если все значения СЕИ S_i содержатся в множестве значений СЕИ S_j.

3.Частичное включение. Две СЕИ S_i и S_J образуют элемент отношения частичного включения, если множества их значений содержат общие элементы.

4.Непересечение. Две СЕИ S_i и S_J образуют элемент отношения непересечения, если множества их значений не содержат общих элементов.

Множества значений используется для поддержания достоверности информации в ЭИС и выбора пути доступа к данным.

9.Реляционная модель данных

Модель данных – это некоторая информационная конструкция, которая описывает множество допустимых операций над данными и множество ограничений для хранения значений данных, которые отражают свойства некоторой предметной области, имеющей интерес для пользователя.

Предметная область – это элементы материальной среды, информация о которой хранится и обрабатывается в модели данных (базе данных).

Реляционная модель данных (РМД) некоторой предметной области представляет собой набор отношений, изменяющихся во времени.

Совокупность отношений позволяет хранить данные об объектах предметной области и устанавливать связи между ними. Элементы РМД и формы их представления приведены в таблице.1.1.

Элементы реляционной модели

Таблица 1.1

Элементы реляционной модели	Формы представления
Отношение	Таблица
Схема отношения	Заголовок таблицы (столбцов таблицы)
Кортеж	Строка таблицы
Сущность	Описание свойств объекта
Атрибут	Заголовок столбца таблицы
Домен	Множество допустимых значений атрибута
Значение атрибута	Значения поля в записи
Первичный ключ	Один или несколько атрибутов
Тип данных	Тип значений элементов таблицы

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные.

Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.

Атрибуты представляют собой свойства, характеризующие сущность в структуре таблицы. Каждый атрибут именуется и ему соответствует заголовок некоторого столбца таблицы.

Домен представляет собой множество всех возможных значений определенного атрибута отношения. Каждый домен образует значения одного типа данных, например, числовые или символьные.

Кортеж представляет собой строку таблицы. Множество упорядоченных кортежей образуют отношение. Отношение не содержит одинаковых кортежей.

Схема отношения (заголовок отношения) представляет собой список имен атрибутов. Например, схема отношения имеет вид сотрудник (ФИО, отдел, должность).

Первичным ключом называется атрибут отношения, однозначно идентифицирующий каждый из его кортежей. Например, в отношении сотрудник (ФИО, отдел, должность) ключевым является атрибут «ФИО».

Ключ может быть составным (сложным), т.е. состоять из нескольких атрибутов. Ключи обычно используются для достижения следующих целей:

1. исключения дублирования значений в ключевых атрибутах;
2. упорядочения кортежей по возрастанию или убыванию значений всех ключевых атрибутов;
3. ускорения работы с кортежами отношения;
4. организации связывания таблицы.

Пусть в отношении R_1 имеется не ключевой атрибут A_1, значения которого являются значениями ключевого атрибута B другого отношения R_2. Тогда атрибут B отношения R_2 есть внешний ключ. С помощью внешних ключей устанавливаются связи между отношениями.

Реляционная модель накладывает на внешние ключи ограничения для обеспечения целостности данных, называемых **ссылочной целостностью**. Это означает, что каждому значению внешнего ключа должны соответствовать строки в связываемых отношениях.

Таблица считается отношением, если:

- все строки таблицы уникальны, т.е. нет одинаковых строк;
- имена столбцов таблицы должны быть различны, а значения их простыми, т.е. недопустимо составное значение в одном столбце таблицы;
- все строки одной таблицы должны иметь одну структуру, соответствующую именам и типам столбцов;
- порядок размещения строк в таблице может быть произвольным.

Наиболее часто таблица с отношение) размещается в отдельном файле. Отдельная таблица (отношение) может считаться базой данных, или база данных может состоять из нескольких таблиц.

Реляционная модель данных представляет собой множество отношений. Множество отношений и операций над ними образуют реляционную алгебру, основу которой создали два логика – американец Чарльз Седер Пирс (1839-1914) и немец Эрнст Шредер (1841-1902). Американский математик Э.Ф. Кодд в 1970 г. на основе этих работ сформулировал основные понятия и ограничения реляционной модели.

10. Реляционная алгебра

Множество отношений и отрицаний над ними образуют реляционную алгебру.

Вариант реляционной алгебры, предложенный Коддом, включает следующие основные операции:

1. объединение;
2. разность (вычитание);
3. пересечение;
4. декартово (прямое) произведение;
5. выборка (ограничение);
6. проекция;
7. деление;

8. соединение.

Операции реляционной алгебры могут выполняться над одним отношением или над двумя, т.е. операция может быть унарной или бинарной.

Операция объединение: $R=R_1 \cup R_2$.

Отношение R содержит все элементы исходных отношений (с исключением повторений). Отношения R_1 и R_2 являются одинаковой размерности.

Операция вычитание: $R=R_1 \setminus R_2$.

R – отношение содержит множество кортежей, принадлежащих R_1, без кортежей, принадлежащих R_2. Результат операции вычитания зависит от порядка следования аргументов R_1 и R_2.

Операция пересечение двух совместных отношений R_1 и R_2 одинаковой размерности порождает отношение R, включающее в себя кортежи, одновременно принадлежащие обоим исходным отношениям, т.е.

$$R=R_1 \cap R_2.$$

Операция произведение отношения R_1 степени $К_1$ и отношения R_2 степени $К_2$ есть такое отношение R степени $К_1+К_2$, заголовок которого представляет сцепление заголовков R_1 и R_2, а тело – имеет кортежи такие, что первые $К_1$ элементов кортежей принадлежат отношению R_2, а последние $К_2$ элементов - отношению R_2.

Операция выборка отношения R по формуле F представляет собой новое отношение, состоящее из таких кортежей отношения R_1, которые удовлетворяют истинности логического выражения, заданного формулой F. Для записи формулы F используются имена атрибутов, логические операции U, OR, NOT, операции сравнения и скобки.

Операция проекция отношения A переносит в результирующее отношение R те же столбцы исходного отношения A, которые указаны в условиях операции, т.е.:

$$R=A\,[X],$$

где X – список атрибутов в структуре отношения A.

Выражения вида R=A [ ] означает пустую проекцию, результатом которой является пустое отношение R.

Операция деления отношения R_1 с атрибутами A и B на отношение R_2 с атрибутом B, где A и B – простые или составные атрибуты, причем атрибут B – общий атрибут, определенный на одном и том же домене, является отношением R, состоящим из кортежей ***r*** таких, что в отношении R_1 имеются кортежи ***(r, s)***, причем ***s*** включает множество значений атрибутов B отношения R_2.

Иными словами, определяется образ значения ***а*** атрибута A отношения R_1, выраженный через множество значений ***в*** атрибута B, причем каждый элемент ***в*** образует вместе с ***а*** некоторую строку отношения R_1. Задача решается путем пересечения найденных образов.

Пример.

Отношение R_1 **Отношение R_2**

X	Y	Z	U

X_1	Y_1	Z_1	U_1
X_1	Y_2	Z_2	U_2
X_2	Y_3	Z_1	U_1
X_3	Y_1	Z_1	U_1
X_2	Y_3	Z_2	U_2
X_3	Y_1	Z_2	U_2

Z	U
Z_1	U_1
Z_2	U_2

Вычисляем образы множества значений атрибутов X и Y отношения R_1.
Образами являются множество значений атрибутов Z и U отношений R_1 и R_2.

$$im\ (Z_1,U_1)=\{(X_1,Y_1), (X_2,Y_3),(X_3,Y_1)\}$$
$$im\ (Z_2,U_2)=\{(X_1,Y_2), (X_2,Y_3),(X_3,Y_1)\}$$

Пересечение образов приводит к отношению R.

Отношение R

X	Y
X_2	Y_3
X_3	Y_1

Таким образом, результатом операции деления является отношение, содержащее пересечение образов отношения делителя, вычисленных на основе отношения делимого:

$$R=R_1\ [a, b \div b]\ R_2 \text{ или } R=D\ [R_1, R_2].$$

Операция соединение отношений R_1 и R_2 по условию, заданному формулой F, представляет собой отношение R_1, которое можно получить путем Декартова произведения отношений R_1 и R_2 с последующим применением к результату операции выборки по формуле F.

Иными словами, каждая строка первого исходного отношения сопоставляется по очереди со всеми строками второго отношения, и если для этой пары строк соблюдается условие соединения, то они сцепляются и образуют очередную строку в результирующем отношении.

Частный случай соединения называется **натуральным соединением** (эквисоединением). Если знаком сравнения в условии эквисоединения является равенство («=»), то в результате эквисоединения столбцы из R_2 подавляются.

Если R_1 и R_2 не содержат общих реквизитов, то результирующее отношение определяется произведением исходных отношений, т.е.:

$$R=R_1 * R_2.$$

Если отношения R_1 и R_2 являются синонимами, то эквисоединение определяется пересечением списков атрибутов исходных отношений, т.е.:

$$R=R_1 \cap R_2.$$

11. Нормализация отношений

Основной задачей, решаемой в процессе проектирования баз данных, является задача нормализации ее отношений. Метод нормализации базируется на понятии зависимости между атрибутами отношения.
Нормализация отношения должна отвечать следующим требованиям:

1. множество отношений должно обеспечивать минимальную избыточность представления информации;
2. перестройка набора отношений при добавлении в базу данных новых атрибутов должна быть минимальной.

Процесс преобразования отношений базы данных к той или иной нормальной форме называется **нормализацией отношений.**
Цель нормализации отношений – получать отношения с более простой структурой, у которых функциональные зависимости выражены простейшим способом.

Зависимость между атрибутами

Между атрибутами отношений существуют зависимости:

- функциональные;
- транзитивные;
- многозначные.

Понятие **функциональной зависимости** является базовым в теории нормализации отношений.
Например, атрибут ***B*** функционально зависит от атрибута ***A***, если каждому значению ***a*** атрибута ***A*** соответствует в точности одно значение ***в*** атрибута ***B***.
Математически функциональная зависимость ***B*** от ***A*** обозначается записью:

$$A \rightarrow B, \text{ т.е. } B=F(A).$$

Атрибуты A и B могут быть составными, т.е. состоять из двух и более атрибутов.
Наличие функциональной зависимости между атрибутами отношения определяется природой вещей, информация о которых представлена кортежами отношения.
Иными словами, основной способ определения наличия функциональных зависимостей – внимательный анализ семантики атрибутов. Для каждого отношения может существовать определенное множество функциональных зависимостей между атрибутами.
Пример. См. табл. 1.1. **Исходное отношение R**

Таблица 1.1

ФИО	Должность	Оклад	Стаж	КАФ	Предмет	Группа	Вид

							занятий
Иванов	преподаватель	5000	5	2	СУБД	36	практика
Петров	ст. препод.	6000	6	2	Паскаль	45	лекции
Егоров	ассистент	3000	3	2	ТЭИС	36	практика
Сидоров	преподаватель	4000	4	3	ЭИС	45	практика
Егоров	ассистент	3000	4	3	ЭВМ	36	лекции

В отношении R можно выделить функциональные зависимости, исходя из смыслового содержания атрибутов.

ФИО→должность

ФИО→предмет

Должность-оклад

ФИО→КАФ

и другие. Ключевыми атрибутами отношения R являются ФИО, предмет.

Функциональная взаимозависимость. Если существует A→B и B→A, то между A и B имеется взаимно однозначное соответствие. Наличие функциональной взаимозависимости можно выразить в виде A↔B или B↔A. Пример, номер паспорта и ФИО его владельца.

Частичная функциональная зависимость – это зависимость неключевого атрибута от части составного ключа.

Пример. Из таблицы 1.1. следует, что ее ключевыми атрибутами являются ФИО и предмет. В данном случае частичная функциональная зависимость выражается: неключевой атрибут «должность» зависит от части составного ключа «ФИО» и не зависит от ключевого атрибута «предмет».

Полная функциональная зависимость – это зависимость неключевого атрибута от всего составного ключа. В нашем примере атрибут «вид занятий» находится в полной зависимости от составного ключа, т.е. зависит от атрибутов «ФИО» и «предмет».

Транзитивная зависимость. Если атрибут ***C*** зависит от атрибута ***A***, т.е. A→C и атрибут ***B*** зависит от атрибута ***C***, т.е. C→B, тогда A→B, т.е. атрибут ***B*** функционально зависит от атрибута ***A*** или ***B*** =F (A). В отношении R транзитивной зависимостью связаны атрибуты:

ФИО→Должность→Оклад.

Многозначная зависимость означает, что, если атрибут ***B*** многозначно зависит от атрибута ***A***, то каждому значению ***A*** соответствует множество значений ***B***, не связанных с другими атрибутами этого отношения. В отношении R многозначная зависимость проявляется между атрибутами:

ФИО $\Leftarrow$ группа(M:1), ФИО $\Rightarrow$ вид занятий(1:M).

Зависимость «многие ко многим» (M:M) обозначается (A $\Leftrightarrow$ B).

Взаимно независимые атрибуты. Два или более атрибута называются взаимно независимыми, если ни один из этих атрибутов не является функционально зависимым от других атрибутов и обозначается A→B или B→A.

Резюме. Между атрибутами отношения могут существовать функциональные зависимости:

1. один к одному, т.е. 1:1;
2. один ко многим, т.е. 1:M;
3. многие к одному, т.е. M:1;
4. многие к многим, т.е. M:M.

Поскольку зависимость между атрибутами в отношении является нежелательной с точки зрения потери целостности при ее перестройке наборов атрибутов, то стараются расчленить отношения с неполными функциональными зависимостями атрибутов на несколько отношений, между атрибутами которых сохраняется лишь полная (сильная) функциональная зависимость. Иными словами, функциональные зависимости должны сохраняться внутри отношения и отсутствовать между отношениями.

12. Нормальные формы

Процесс нормализации отношений является итерационным процессом и заключается в последовательном переводе отношений из первой нормальной формы в нормальные формы более высокого порядка по определенным правилам. При этом каждая следующая нормальная форма устраняет определенный тип функциональных зависимостей и сохраняет свойства предшествующих нормальных форм.

Первая нормальная форма. **Определение.** Отношение находится в первой нормальной форме тогда и только тогда, когда на пересечении каждого столбца и каждой строки находятся только элементарные значения атрибутов, т.е. не должно быть кортежей. Как правило, исходное отношение строится таким образом, чтобы оно было в 1НФ. Перевод отношения в следующую нормальную форму осуществляется методом «декомпозиции». Основной операцией декомпозиции является операция проекции. Например, если в отношении R (A,B,C,D,E) выделяем функциональную зависимость C→D, то путем взятия проекции по атрибутам ***C*** и ***D***, получим два новых отношения R_1 (A,B,C,E) и R_2 (C,D).

Вторая нормальная форма. Определение. Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от первичного составного ключа или, другими словами, не содержит неполных функциональных зависимостей, не первичных атрибутов от атрибутов первичного ключа.

Для перевода отношения в 2НФ, если оно находится в 1НФ, используется операция проекции, разделяющая исходные отношения на несколько отношений (R_1,R_2) следующим образом:

1. построить проекцию без атрибутов, находящихся в частичной функциональной зависимости от первичного ключа;
2. построить проекции на части составного первичного ключа и атрибуты, зависящие от этих частей.

Так в отношении R первичным ключом является составной ключ, состоящий из атрибутов: «ФИО», «предмет». Неключевые атрибуты «должность», «оклад» функционально не полно зависят от ФИО, Предмет.

R_1 (ФИО, Предмет, Группа, Вид занятия);

R_2 (ФИО, Должность, Оклад, Стаж, Кафедра);

Следовательно, для перехода во 2 НФ необходимо исключить эту неполную функциональную зависимость и перенести атрибуты в новое отношение, разделив отношение R на два отношения.

Таким образом, переход к 2 НФ исключает явную избыточность в отношении R. С созданием двух отношений, каждое из которых содержит атрибутные сохраняющие полную зависимость. Примечательно, если вероятный ключ отношения одно атрибутный, то 2 НФ автоматически соблюдается. Отсюда следует, что бинарное отношение всегда находится во 2 НФ.

Третья нормальная форма(3 НФ). Определение1. Отношение находится в 3 НФ, если оно находится во 2 НФ и не содержит транзитивных зависимостей.

Определение 2. Отношение находится в 3НФ в том и только в том случае, если все неключевые атрибуты отношения взаимно независимые и полностью зависят от первичного ключа.

Из предыдущего примера следует, что транзитивные зависимости существуют в отношении R_2, а именно:

ФИО-ДОЛЖНОСТЬ-ОКЛАД

Транзитивные зависимости также порождают избыточное дублирование информации в отношении. Устраним их. Для этого используя операцию проекции, преобразуем отношение R_2 в отношения R_3 и R_4, и каждое из которых находится в 3НФ.

Тогда R_3 (ФИО, Должность, Стаж, Кафедра);

R_4 (Должность, Оклад);

На практике построение 3НФ схем отношений являются достаточным и приведение к ней, проектирование баз данных заканчивается. Таким образом, в данном примере результат проектирования базы данных состоит из отношений R_1 R_3 и R_4.

Если в отношении имеется зависимость атрибутов, то переходят к нормальной форме Бойса-Кодда (БКНФ), в которой отсутствуют зависимости атрибутов составного ключа от некоторых атрибутов.

Четвертая нормальная форма (ЧНС). Определение. Отношение находится в ЧНФ в том и только в том случае, когда существует многозначная

зависимость. A→B, а все остальные атрибуты отношения функционально зависят от A.
Пример: ФИО, Номер з.к., Группа, Дисциплина.

13. **Сетевая модель данных**

Стандарт сетевой модели впервые был определён в 1975 году организацией CODASYL (Conference of Data System Languages).
Стандарт содержал базовые понятие модели и формальный язык описания сетевой модели данных.
Информационными конструкциями в сетевой модели данных являются отношения и веерные отношения.

Сетевая модель представляется как множество отношений и веерных отношений. Отношения разделяются на основные и зависимые, потомок, предок.

Под отношениями в сетевой модели будем понимать запись или агрегат данных, имеющих имя. Тогда под веерным отношением понимается связь одного отношения с другим отношением. Иными словами, запись-потомок может иметь связь с записью-предком. Причём, если в иерархических структурах запись –потомок могла иметь только одну связь с записью-предка, то в сетевых структурах запись- потомок может иметь произвольное число связей с записью- предка.

Следовательно, сетевая модель данных позволяет отображать разнообразие взаимосвязей отношений или элементов данных, образующих отношения, в виде произвольного графика (см. рис. 3.)

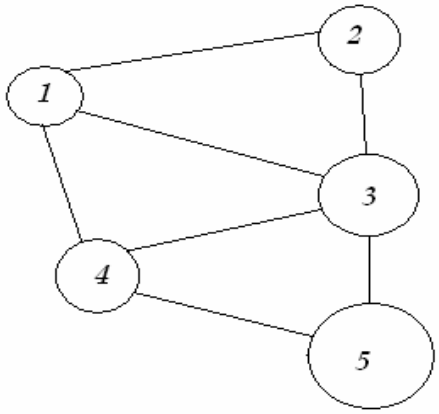


Рис.3 Представление связей в сетевой модели.

Сетевые модели данных в зависимости от ограничений на вхождение отношений в веерные отношения разделяются на многоуровневые сети и двухуровневые сети.
Наибольшее распространение получили двухуровневые сети, обладающие свойством ацикличности.
Для двухуровневых сетевых систем управления базами данных (СУБД) обычно накладывается два ограничения, а именно:

1. Первичные ключ основного отношения, т. е. записи- предка, может быть только одноатрибутным;

2. Веерное отношение существует, если первичный ключ основного отношения является частью первичного ключа зависимого отношения, т. е. записи потомка.

Физическое размещение данных в базах сетевого типа может быть организованно практически теми- же методами, что и в иерархических базах данных, но с некоторыми нововведениями, относящимися к приёмам построения динамических структур данных, образующих понятие списки.

К числу важнейших операций манипулирования данными базы данных сетевого типа можно отнести следующие:

1. Поиск записей в базе данных (Б. Д.);
2. Переход от предка к первому потомку;
3. Переход от потомка к предку;
4. Создание новой записи;
5. Удаление текущей записи;
6. Обновление текущей записи;
7. Включение записи в связь;
8. Исключение записи из связи;
9. Изменение связей и т. д.

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности в силу допустимости образования произвольных связей между записями.

Недостатками сетевой модели данных является высокая сложность схемы Б. Д. и слабость контроля целостности связей в следствии допустимости установления произвольных связей между записями. Наиболее известными СУБД являются сеть, СЕТОР и КОМПАС.

13.1 Организация веерных отношений сетевой модели данных в памяти ЭВМ

Структура взаимосвязей основных и зависимых отношений строится с использованием институтов программирования известных под названием списки.

Простейшая форма списка- это группа объектов над которыми можно производить различные действия, заложенные в программе. Веерные отношения в сетевой модели данных создаются с помощью связанных списков (Linked list), которые используют указатели для создания гибких структур данных. В связанные списки можно добавлять или удалять элементы из любой части связанного списка с минимальными усилиями.

Связанные списки.

Связанный список хранит элементы в структурах данных, называемых ячейками(cells). Каждая ячейка это своего рода запись, состоящая из двух частей: информационной и адресной. Информационная часть содержит подлежащую обработки информации, а в адресной части записи хранится указатель на следующий элемент списка(см. рис 4.), т. е. 1-й элемент списка содержит ссылку на 2-й элемент(или указывает на 2-й элемент)
2-й элемент –ссылку на 3-й и т. д.

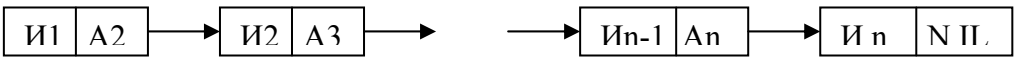


Рис.4 Расположение элементов в списке.

Здесь И1, И2, … И n –указатели (адресные части) на соответствующие элемента списка: А2-адрес 2-го элемента, А3-адрес 3-го элемента и т. д.
Таким образом, каждая ячейка-запись содержит указатель на следующую ячейку в списке. Для этого в записи должна быть переменная Next eel, которая указывает на следующую ячейку в списке. В нём также должны быть определенны переменные для хранения любых данных, с которыми будет работать программа.
Например, в связанном списке с записями о сотрудниках. Эти поля могли бы содержать имя служащего, номер страхового полиса, должность и т. д..
Структура связанного списка будет выглядеть таким образом:

Описание записи

```
Type PEmpcell=  ^TEmpcell;
     TEmpcell=record;
          Имя: String[20];
     Номер - страховки: String[10];
          Должность: String[11];
          NextCell: PEmpcell;
          end;
```

Для создания новых ячеек в программе используется оператор New, выделяя под них необходимое количество памяти.

Программа должна сохранять указатель на начало списка.
Для того, чтобы определить, где заканчивается список, она устанавливает значение Next Cell для последнего элемента в nil.
Например, следующий фрагмент кода создаёт список, содержащий информацию о трёх служащих:

```
Var
     Top-Cell, cell1, cell2, cell3: PempCell;
     begin
        // Построение ячеек.
        New(cell1);
           cell1^.имя:=’Иванов’;
```

```
            cell1^.номер_страховки:=’123-45-6789’;
            cell1^.должность:=’инженер’;
        New(cell2);
            cell2^.имя:=’Петров’;
            cell2^.номер_страховки:=’154-66-4936’;
            cell21^.должность:=’мастер’;
        New(cell3);
            cell3^.имя:=’Жуков’;
            cell3^.номер_страховки:=’163-47-3567’;
            cell3^.должность:=’менеджер’;
    // Связывание элементов списка для построения связанного
списка
            Cell1^.Nextcell:=Cell2;
            Cell2^.Nextcell:=Cell3;
            Cell3^.Nextcell:=nil;
    // Установка указателя на начало списка
            Top_cell:= cell1;
```

На рис.5 изображено схематическое представление этого связанного списка. Прямоугольники соответственной ячейкам, а стрелки- указателями на объекты. Маленький перечёркнутый квадрат представляет значение nil, которое указывает на конец списка.

Переменные top_cell, cell1, cell2, cell3- это не фактические объекты, а только указатели на них.

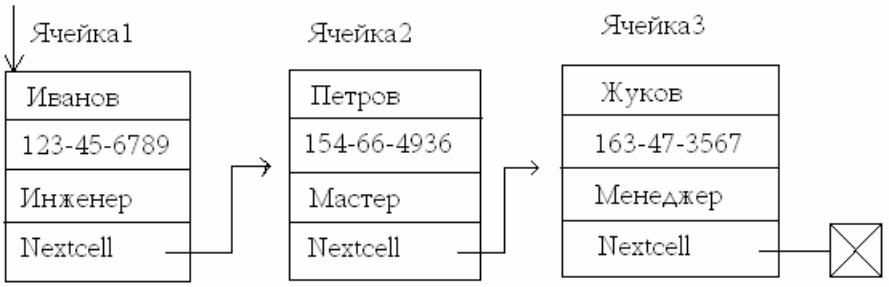


Рис.5 Связанный список

Поиск записи в БД.

Введём переменную ptz, которая будет выполнять функцию указателя на элементы списка. В цикле While реализуется перемещение через весь список до тех пор, пока значение ptz не достигнет конца списка. Во время каждого шага цикла процедура выводит поле “имя” для ячейки, указанной переменной ptz.

Затем программа передвигает ptz, чтобы указать следующую ячейку списка. В конечном итоге ptz достигнет конца списка и получит значение nil, после чего цикл останавливается.

Фрагмент программы поиска записи

```
Var
        ptz: PEmcell;
        Begin
          ptz:=top_cell;
        While (ptz<>nil) do
                Begin
        ShowMessage (ptz^.имя);
                ptz:= ptz^.NextCell;
                  end;
                  end. ,
```

где top_cell- указание на вершину списков.

Связанные списки обладают важными свойствами, позволяющими добавлять новые ячейки в начало списка, в середину списка, в конец списка. удалять ячейки из списка. При этом следует не забывать об освобождении памяти под удаляемую ячейку.

Существуют стандартные соглашения о способах включения и исключения ячеек из списка.

Веерные отношения между записями могут быть установлены из смыслового содержания функциональной зависимости элементов одной записи осей элементов другой записи. В теоретическом плане веерные отношение существующей, если ключ основного отношения функционально определяем ключ зависимого отношения, т. е. каждое значение основного отношения связано с единственным значением зависимость отношения. Иными словами, в составной элемент зависимого отношения входит в элемент основного отношения.

Например, элемент основного отношения –код группы и элемент зависимого отношения –код зачётной книжки студента этой группы

F1(Код группы#, Курс)

F2(Код зачётной книжки#, ФИО студента#, кафедра)

Сетевая модель данных является ациклической, если между любыми двумя вершинами графы существует не более одного пути.

Алгоритм получения двухуровневой структуры сети.

1. Для каждой функциональной зависимости вида $A \to B$ создаётся файл $F_i(A,B)$ или отношение, соответствующее 3НФ.
2. У всех пар файлов, на шаге 1 проверяется условие для ключей, т. е. ключ основного отношения $К_i$ является частью ключей Кj < Ki зависимого отношения.

Если условие 2 соблюдается, то из соответствующих отношений создаётся веерные отношения $W_{ij}(F_i,Fj)$

3. Если на шаге 2 будут. получены два веерных отношения W_{ij} и Wj_k , то все атрибуты отношения Fi передаются в отношение Fj , а F_i уничтожается.
4. Атрибуты, не вошедшие в состав веерных отношений на шаге 2, добавляются в те отношения F_n , где они будут не ключевыми. При

этом предпочтение в приёме атрибутов отдаётся основным отношениям. Если такие отношения отсутствуют, то создаётся новое отношение

5. Запрещается существование отношения в качестве основного в одном веерном отношении и одновременно в качестве зависимого в другом веерном отношении

Пример. Атрибуты объекта: Фамилия, Стаж, Дата, Сумма основная, дополнительная, зарплата.

F1(Фамилия*, стаж;)

F2(Фамилия*, Дата*, сумма, основная зарплата)

F3(Фамилия*, Дата*, сумма, дополнительная зарплата)

На рис.6. Показана структура соответствующей двухуровневой сетевой базы данных

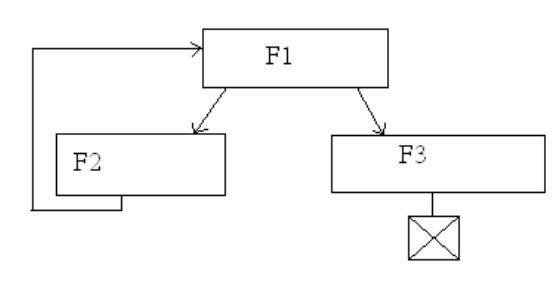


Рис.6 . Сетевая база данных о сотруднике

Структура основных отношений показана в верхней части рисунка, а структура зависимых отношений – внизу.

Таким образом, в сетевых моделях непосредственный доступ по ключу может обеспечиваться к любому объёму независимо от уровня, на котором он находится в модели. Возможен также доступ по связям от любой точки доступа.

Фрагмент кода для рис. будет иметь вид

1. // связывания элементов списка

```
F1^.next:=F2;
F2^.next:=F1;
F3^.next:=nil;
```

2. //Установка указателя на начало в списке

```
Top_cell:=F1;
Ptz:=top_Cell;
While (ptz<>nil)
Begin
:
Ptz:=ptz^.Next;
F1^.next:=F3;
end;
```

Пример. Пусть сведени.я об объекте зафиксированы в виде совокупности следующих атрибутов:

Цех, план, склад, продукция, выдача.

Решение. По п.1 алгоритма получения двухуровневой структуры сети составили отношения Fi из атрибутов функционально связанных между собой (Ключи помечены знаком #).

F1(цех#, продукция, план);
F2(цех#, продукция#, склад);
F3(склад#, продукция#, выдача);
F4(склад#, продукция);
F5(продукция#, выдача);

По п.2. у всех пар отношений проверяется условие для ключей и создаётся веерное отношение Wij(Fi,Fj)

В частности, в нашем примере получим

$W_{12}(F_1, F_2)$, $W_{43}(F_4, F_3)$, $W_{53}(F_5, F_3)$, $W_{52}(F_5, F_2)$

На рис.7. показана структура соответствующей сетевой модели.

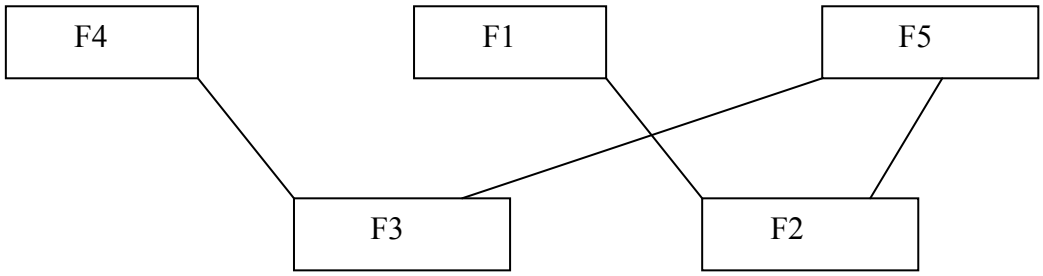


Рис. 7. Сетевая модель данных о цехе

Здесь структура основных отношений показана в верхней части рисунка, а структура зависимых отношений- внизу рисунка.

14.Иерархическая модель данных

Иерархическая модель имеет свою логическую структуру, в которой связи между данными можно описать с помощью упорядоченного графа (или дерева).

Упрощено представление связей между данными в иерархической модели показано на рис.8

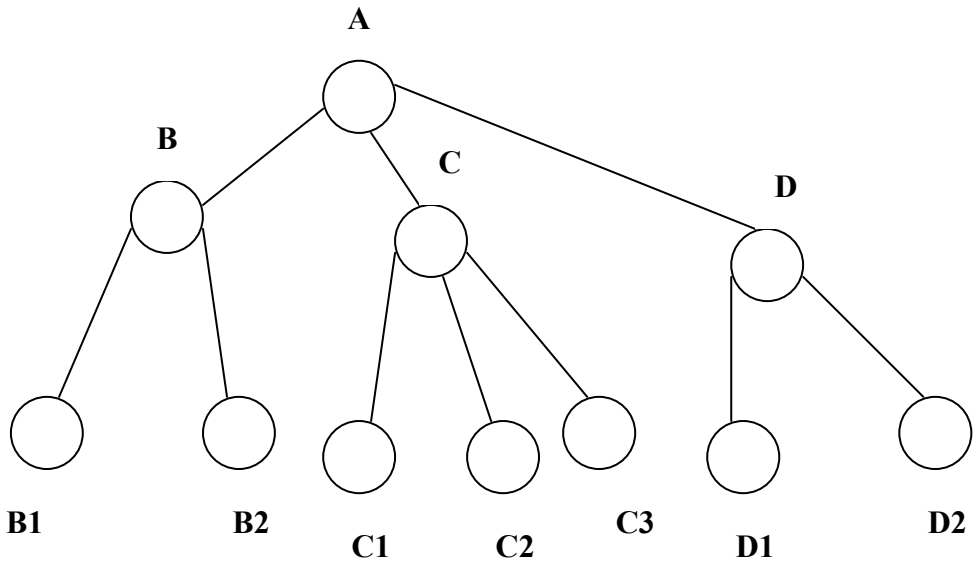


Рис.8. Представление связей в иерархической модели.

Для описания структуры иерархической модели на некотором языке программирования используется тип данных «дерево». Тип «дерево» схож с типом данных «запись» на языке программирования Турбо Паскаль. Деревом называется конечный связанный граф без циклов, имеющих не более двух вершин. Для каждой пары его вершин существует единственная соединяющая цепь.

Тип «дерево» является составным. Он включает в себя подтипы, каждый из которых, в свою очередь, является типом «дерево» для других подтипов. Каждый из типов «дерево» состоит из одного «корневого» типа и упорядоченного набора подчиненных типов. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Простая запись состоит из одного типа, а составная «запись» объединяет некоторую совокупность типов, включая и указатель.

Корневым называется тип, который имеет подчиненные типы и сам не является подтипом. Подчиненный тип (подтип) является потомком по отношению к типу, который выступает для него в роли предка (родителя). Потомки одно и того же типа являются близнецами по отношению друг к другу.

В целом тип «дерево» представляет собой иерархически (подчиненно) организованный набор типов.

«Запись», обычно отношение родства между типами переносятся между самими записями.

На рис. Корневой узел A соединен с тремя поддеревьями B,C,D. Эти узлы соединены соответственно с поддеревьями B1 и B2, C1, C2 и C3, а также D1 и D2.
Узел (node) – это точка, где может возникнуть ветвь. Ветвь описывает связь между двумя узлами, лист (leaf) – это узел откуда не выходит другая ветвь.
Из генеалогии пришли термины, описывающие отношения. Если узел находится непосредственно над другим узлом, то он называется родительским (parent), а нижний узел называется дочерним (child). Узлы на пути вверх от узла до корня принято считать предками узла. Например, на рис. Предками узла C2 являются узлы C и A. Все узлы, расположенные ниже какого-либо узла, называются его потомками. На рис. Потомками узла B являются узлы B1 и B2.
Внутренний узел (internal node) – это узел, не являющийся листом. Порядком узла (node degree) чем его степень называется количество его дочерних узлов.
Степень дерева – это максимальный порядок его узлов. На рис. Степень дерева равна трем, так как узел C имеет три дочерних узла.
Глубина (depth) узла равна числу его предков плюс 1. На рис. B1 имеет глубину три.
Глубина дерева – это наибольшая глубина всех узлов. Глубина дерева, изображенного на рис., равна трем.
Дерево степени два, называется двоичным деревом. Аналогично дерево степени N называется N-иным деревом.
Итак, иерархическая модель – представляется как множество отношений и верных отношений. Отношения в верных отношениях разделяется на основные и зависимые, предки и потомки, родительские и дочерние.
Связь между отношениями существует (т.е. имеет место верное отношение), если ключ отношения функционально определяет ключ отношения.
Например, пусть отношение A является основным, а отношение B-зависимым. Тогда между отношениями A и B существует верное (зависимое) отношение, если ключ «b» отношение A функционально определяет ключ «b» отношения A, т.е. a стремится к b. Иными словами b=F(a), т.е. ключ «b» есть функции от ключа a.
И наоборот, если ключ первого отношения функционально определяет ключ второго отношения, то первое отношение может быть основным, а второе – зависимым в их верном отношении.

Алгоритм получения двухуровневой структуры иерархической модели данных.

1. Для каждой функциональной зависимости вида A $\longrightarrow$ B создается отношение S(A,B), т.е. группа атрибутов между которыми существует связь по содержанию (по смыслу), образует отношение из набора этих атрибутов.
2. Отношения делятся на группы по признаку: два отношения находятся в общей группе, если их ключи функционально определяют хотя бы

один общий атрибут, т.е. отношения образуют группу, если каждое отношение имеет хотя бы один общий атрибут.

3. У всех пар отношений группы проверяется условие для ключей $K_j \longrightarrow K_i$. Если оно соблюдается, то из соответствующих отношений создается верные отношения $W_{ij}(S_i,S_j)$, где отношение S_i является основным, а S_j – зависимым.
4. По верным отношениям образуется иерархия отношений, т.е. дерево отношений.
5. Отношения вне верных отношений добавляются в структуру тех отношений, где их атрибуты будут не ключевыми.

Внутренняя структура иерархической модели.

На языке программирования Turbo Pascal составим структуру записей в соответствии с рис. Корневой тип A имеет три подтипа B, C и D. Подтипы B, C и D содержат поля записей

```
Type B Rec = record
             B1: тип
             B2: тип
             end:

C Rec = record
             C1: тип
             C2: тип
             C3: тип
             end:
D Rec = record
             D1: тип
             D2: тип
             end:
A Rec = record
             B: B Rec
             C: C Rec
             D: D Rec
             end:
 Var X: A Rec;
             begon
             .
             .
             end.
```

Поле записей хранят числовые или символические значения, составляющие основное содержание модели данных. Обход всех элементов иерархической модели обычно производится сверху вниз и слева направо. К основным операциям манипулирования иерархически организованными данными относятся следующие:

1. Поиск указанного экземпляра базы данных;

2. Переход от одного дерева к другому, т.е. в иерархической модели допускается отображение одной переменной области в нескольких иерархических баз данных;
3. Переход от одной записи к другой внутри дерева;
4. Вставка новой записи в указанную позицию;
5. Удаление текущей записи и т.д.

Пример: объект описывается набором атрибутов:

Владелец, Адрес, Недвижимость, Договор, Дата, Площадь, Цена, Арендатор;

Решение:

1. Составим отношение по функциональной зависимости между атрибутами (ключи помечены значком #)
   F1 (Владелец #, недвижимость, Адрес)
   F2 (Договор #, недвижимость, Дата #)
   F3 (Арендатор #, Адрес, Договор #)
   F4 (Недвижимость #, Адрес #, Площадь)
   F5 (Недвижимость #, Адрес #, Цена)
2. Определим верные отношения без разделения их на группы. Иными словами разработаем многоуровневую структуру базы данных. Для этого используем призыск подчинения (старшинства) по объему понятия отношений.

Верные отношения: W_{12} (F1, F2), W_{23} (F2, F3), W_{24} (F2, F4); W_{45} (F4, F5).
На рис.9. Показана структура соответствующей иерархической модели.

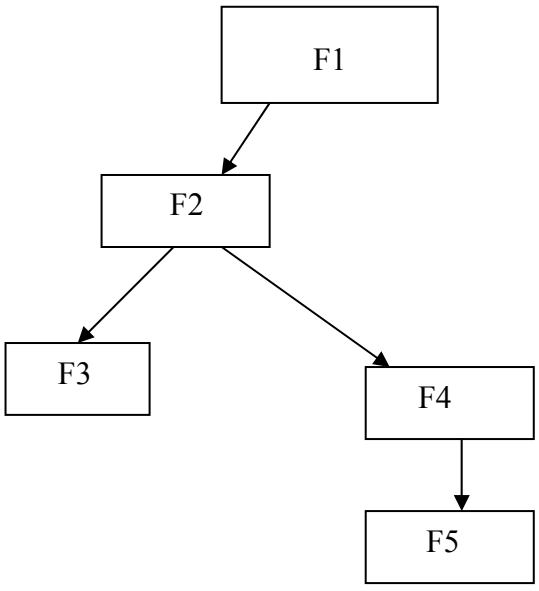


Рис.9. Иерархическая модель данных о договоре.

Структура основных отношений, обладающих большим объемом понятия отношения показаны на верхних относительных уровнях, а структура зависимых отношений относительно на низких уровнях.
Таким образом, физическое представление логической структуры иерархической модели – это совокупность записей различных типов, каждая из которых ссылается только на следующую. Следовательно, поиск информации возможен сверху вниз, т.е. доступ по связям от объекта на вершины модели.
Для получения двухуровневой модели данных следует ориентироваться на приведенный алгоритм. Основное правило контроля целостности иерархической модели формируется следующим образом: потомок не может существовать без родителя, а родитель не может не иметь потомка.
Между предками и потомками автоматически поддерживается контроль целостности связей. Механизм поддерживания целостности связей между записями различных деревьев отсутствует.
К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ, а также удобная и понятная для пользователя работа с иерархически упорядоченной информацией.
Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями. Примерами отечественных иерархических моделей являются системы: ОКА, ИНЭС, МИРИС

15. Ациклические базы данных

Отношения могут содержать не только функциональные зависимости между атрибутами, но и многозначные и взаимнозначные зависимости.

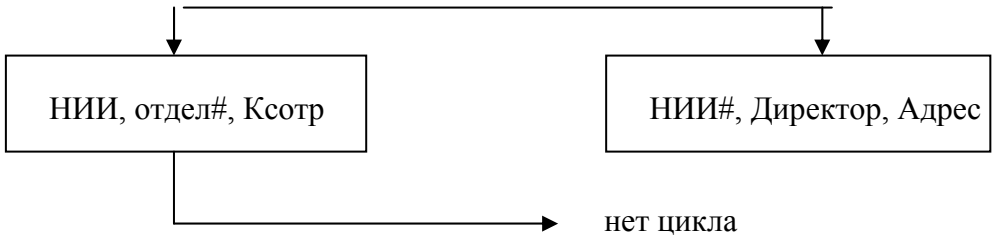
Наличие в отношении большого числа функциональных зависимостей обычно приводит к усложнению процесса корректировки в случае замены или исключения одного реквизита, который функционально зависит от многих других реквизитов. В этом случае приходится исключать или заменять определенное количество строк в отношении. Поэтому все отношения представляют в более простой структуре, у которой функциональные зависимости взаимосвязаны простейшим способом. Такой процесс преобразования отношений называется нормализацией, а отношение, для которого не допускаются те или иные варианты функциональных зависимостей, называется отношением с нормальной формой. (НФ)

Многозначная зависимость реквизитов фиксируется в определении четвертой нормальной форме (4НФ) отношения. Отношение R (x, y, z) находится в 4НФ, если атрибуты X и У образуют многозначную зависимость X→→У, в которой X и У либо содержат все реквизиты отношения R, либо ключ отношения R содержится в X.
Приведение отношения R (x, y, z) к 4НФ сводится к разделению его на две проекции: R_1 (x, y) и R_2 (x, y).

Пример. R (цех, продукция, сырье): где X – цех, У – продукция, Z – сырье. Здесь операция соединения отношений R_1 <> R_2 позволяет получить исходное отношение R.

Необходимость избежать в отношении многозначной функциональной зависимости, приводит к декомпозиции отношения, которая может содержать серьезные недостатки, приводящие к циклической базе данных. Однозначная декомпозиция отношения характерна для ациклических баз данных, т. е. баз данных не образующих циклы между отношениями. Например, база данных, состоящая из отношений S_1, S_2, S_3 является ациклической, если в графе соединений можно разорвать любую из дуг и превратить граф в дерево. Пусть имеет место R_1 (НИИ #, Директор, Адрес); R_2 (НИИ, отдел #, Ксотр). Отношения R_1 и R_2 имеют одно атрибутный ключ, следовательно, эта база данных является ациклической.

Действительно,



Алгоритм проверки структуры БД на ацикличность

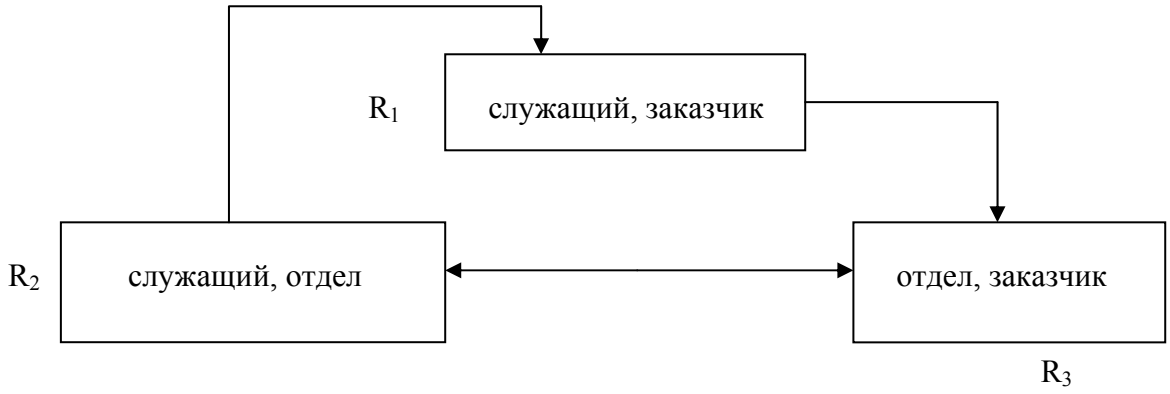
База данных с циклическим графом соединений может давать некорректные ответы на запросы из-за существования неравноценных путей доступа к данным при реализации запросов.

Метод 1. Если некоторый атрибут встречается только в одном отношении, вычеркнуть данный атрибут из этого отношения;

Метод 2. Если все атрибуты некоторого отношения находятся среди атрибутов другого отношения, то первое отношение вычеркивается из списка.

Пример циклической базы данных.

R_1 (служащий, заказчик); R_2 (служащий, отдел); R_3 (отдел, заказчик)

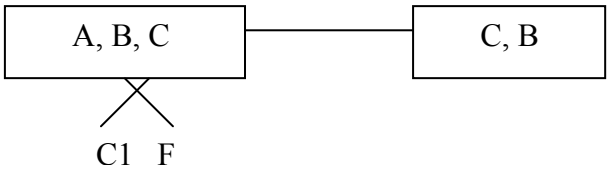


Восстановление свойств ацикличности БД может быть проведено двумя способами:

1. Добавление в БД нового отношения с атрибутами, равными объединению двух отношений, образующих цикл.
2. Добавление новых атрибутов, переименование и разделение атрибутов.

Алгоритм разделения атрибута отношения с атрибутами A, B, C и A, E, C. Заменяем в отношениях атрибут C на C_1 и C_2, т. е. A, B, C_1 и A, E, C_2.

Алгоритм добавления атрибута. Отношения с атрибутами A, B, C и C,B. Заменяем атрибут B отношении A, B, C на атрибут F.



Рекомендации при проведении проектирования структуры реляционной модели данных.

1. Каждый входной документ привести к ЗНФ и установить первичный ключ в каждом случае.
2. Для полученного множества отношений построить граф соединений. Если граф соединений можно преобразовать в дерево соединений, то база данных в целом является ациклической и соответствует ЗНФ.

Критерии, которым соответствует база данных в ЗНФ и ациклической БД ,не совпадают.

Завод, изделие, план

Заменяем изделие на материал

Завод, изделие

R1(студент, группа)
R2(Группа, вып каф)
R3(студ, куратор)
R4(курат, кафедра)

Студ, группа – группа,ВК
Студ, куратор – куратор, ВК5

Ациклическая структура базы данных

Доступ к реляционной базе данных

Реализацию запросов к базе данных с помощью операторов реляционной алгебры можно описать следующими правилами.

1. В формулировке запроса выделить имена атрибутов, вход и выход запроса и условия выборки.
2. Зафиксировать множество запрашиваемых атрибутов. Если все необходимые атрибуты находятся в каком-то одном отношении, то последующие операции выборки и проекции проводить только с ним.
   Если требуемые атрибуты распределены по нескольким отношениям, то эти отношения необходимо соединить. Каждая пара отношений соединяется по условию равенства атрибутов с совпадающими именами. После каждого соединения с помощью проекции можно отсечь ненужные атрибуты.
3. Полученное единственное отношение далее обрабатывается операциями выборки и проекции. Выборка по значениям атрибута должна предшествовать проекции, в которой этот атрибут выводится из отношения.
4. Если запрос можно разделить на части (подзапросы), то его реализация также разделяется на части, где результатом каждого подзапроса является отдельное отношение.
5. Указанная последовательность действий является стандартной.

6. Если создается отношение большего размера, то операции выборки и проекции могут предшествовать операции соединения.

Для оптимизации запроса к реляционной базе данных используется следующие преобразования:

1. объединение операций выборки с составным условием;
2. если операция проекции применяется к результату выполнения пересечения, объединения, вычитания или соединения двух отношений, то она может быть предварительно применена к каждому исходному отношению;
3. если операция выборки применяется к результату выполнения пересечения, объединения, вычитания или соединения двух отношений, то она может быть предварительно применена к каждому исходному отношению.

В настоящее время распространенным языком запросов является SQL (структурированный язык запросов). Центральным средством доступа к БД в SQL являются команды Select и ее параметры From, Where, Group by, Having, Order by.

В команде Select указываются имена выводимых атрибутов или знак *, если надо выводить все атрибуты.

Параметр From является обязательным и содержит имена требуемых для выполнения запроса отношений.

Параметр Where определяет условия, которым должны удовлетворять выводимые данные.

Параметр Group by объединяет записи с одинаковым значением атрибута-ключа.

Параметр Having проверяет условия в группе, выделенной с помощью Group by.

Параметр Order by определяет сортировку имен атрибутов.

Классификация запросов к реляционной б.д.

Рассмотрим три класса запросов:

1. Найти значение атрибута по известному значению ключа
2. Найти записи с заданным значением не ключевого атрибута
3. Перечислить значения данного атрибута для каждой записи.

Условия запроса реализуются с помощью логических связок НЕ, И, ИЛИ

16. Модели данных в экономике

Для описания данных используют две формы представления информации о них:

1. Синтаксические модели данных, т. е. информация о данных представляется по условно принятым правилам естественного языка.
2. Семантические модели данных – это такие модели, в которых информация о данных выводится из их смыслового содержания.

Семантические модели данных в общем виде призваны отображать структуру предметной области, таким образом, чтобы понятия модели были бы понятны как специалисту, так и пользователю.

Семантические модели данных объединяют две подмодели:

- модель «сущность – связь»;
- модель семантических сетей.

Между этими двумя подмоделями имеется много общего, но есть и отличия. Общая черта этих подмоделей – это графическое представление всех компонентов предметной области в виде диаграммы, отражающей «сущность», т.е. тип физического объекта и тип связи.

Допускаются следующие типы связей:

1) N-е связи;
2) рекурсивные связи;
3) несколько связей для пары объектов.

Отличие модели «сущность – связь» от модели семантической сети в том, что первая модель не отражает уровня иерархии управления и характер действий компонентов модели по уровням управления.

Модель семантических систем

Повышение выразительной силы изобразительных средств достигается в модели семантической сети.

Семантическая сеть – это ориентированный граф с помеченными именованными дугами. Осмысленность структуры предметной области достигается тем, что вершины графа могут отражать физический объект в виде его понятия, события, происходящего в этом объекте, его свойства и значения. Иными словами, изобразительные средства семантической сети изображают сценарий поведения объекта, включающий набор понятий, событий причинно-следственных связей. Вершины графа могут обозначать экземпляры объектов и классы объектов, т.е. один и тот же экземпляр объекта может быть соотнесен к нескольким классам. Имена вершин и метки на дугах совпадают с терминами профессионального языка предметной области.

Различие между вершинами сети приводит к существованию трех типов дуг:

- дуга, соединяющая два экземпляра, соответствует утверждению;
- дуга между классом и экземпляром – показывает принадлежность экземпляра к классу;
- дуга, связывающая два класса – определяет бинарное отношение (см. рис 10).

-
-

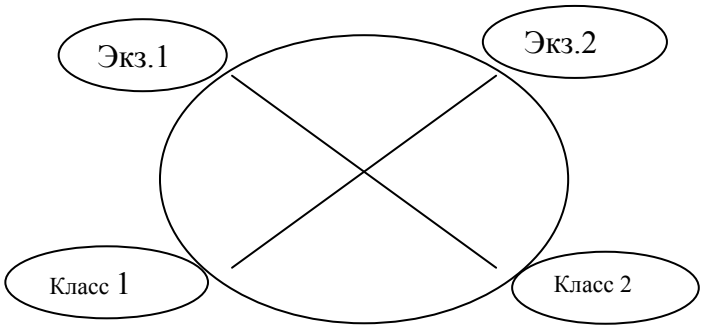


Рис. 10. Типы дуг между вершинами сети

В семантической сети существуют две обязательные связи при установлении структуры понятий:

Связь «есть – некоторый», т.е. экземпляр принадлежит некоторому (нек) классу.

Связь «есть – часть», т.е. объект содержит разнородные экземпляры или принадлежит часть к целому.

Представления событий и действий с помощью семантической сети выполняется в следующей последовательности. Предварительно выделяются простые отношения, которые характеризуют основные компоненты объекта. Затем из события выделяется действие, которое обычно описывается глаголом. Далее определяются объекты, которые действуют, объекты над которыми эти действия производятся и т.д.

В сети указываются:

- объект – предмет, инициирующий действие;
- объект – предмет, подвергающийся действию;
- источник – размещение предмета перед действием;
- приемник – размещение предмета после действия;
- время – когда происходит действие и проявляется событие;
- место – где происходит событие;
- цель – указание цели действия.

Пример. Структура объекта. Директор завода остановил 05.01.2005 цех № 5, чтобы заменить оборудование (см.рис.11.).

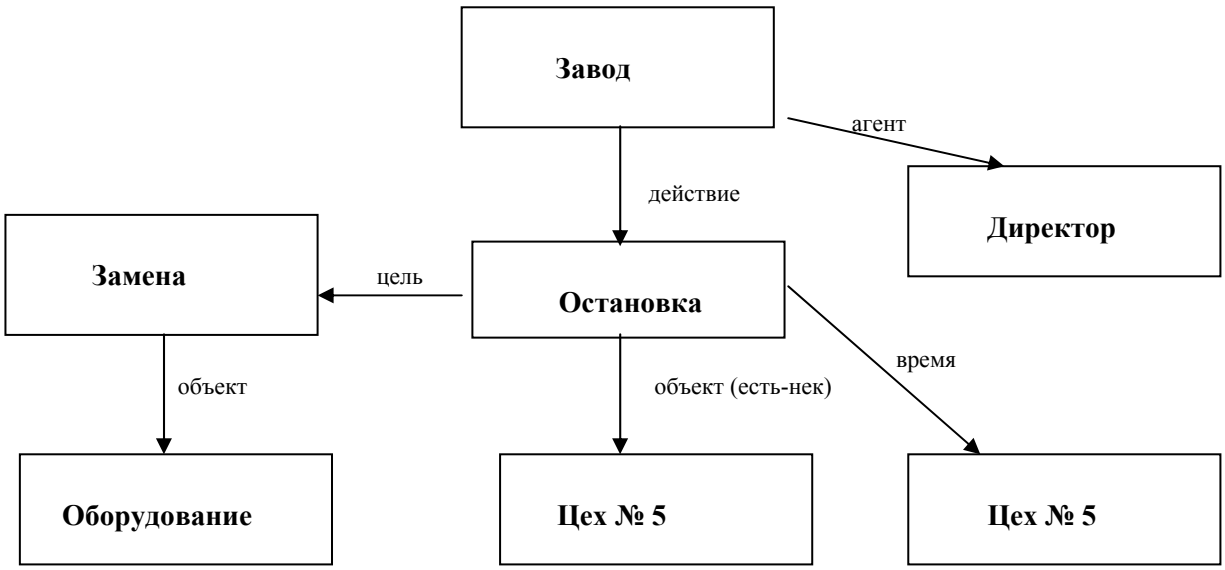


Рис. 11. Пример семантической сети

17.Методы ускорения доступа к данным

1. Хеширование

Ускорение доступа к данным достигается применением различных методов размещения информации и её поиска, путем создания массивов вспомогательной информации о хранимых данных.

Доступ к требуемым записям может осуществляться не только путем сравнения искомого значения ключа с ключами записей, извлекаемых из массива по определенному алгоритму, но и в результате вычисления местоположения требуемой записи.

Расстановка записей происходит по некоторой функции, которая вводит соответствие между искомым значением и индексом позиции, где он должен находиться. Сравнивая искомые значения со значением в известных позициях, алгоритм находит искомые значения. В алгоритме хеширования используется подобный принцип для преобразования элементов в хеш-таблицу.

Для реализации хеширования необходимы три вещи:

1.структура данных, называемой хеш-таблицей, для хранения данных;
2.хеш-функция для отображения ключевых значений на ячейки таблицы
3.алгоритм разрешения конфликтных ситуаций, если ключи отображаются на одну и ту же позицию.

Один из методов разрешения конфликтов состоит в том,чтобы сохранять записи, отображаемые на одну позицию таблицы, в связанных списках. Для вставки новой записи с помощью хеш-функции выбирается связанный список, в котором будет находиться эта запись. Затем запись добавляется в этот список.

Пример.

Допустим хеш-таблица содержит 5 ячеек. Хеш-функция отображает ключ К на позицию массива по функции К mod 5.

Каждая позиция массива содержит указатель на первый элемент связанного списка. Чтобы вставить элемент в таблицу, добавляют его в соответствующий список.

Пусть в хеш-таблицу добавляются следующие элементы в указанном порядке:50,13,10,72,25,46,68,30,99,93.Связывание списков представим на рисунке 12.

0	1	2	3	4
50	46	72	13	99
10			68	
25			93	
30				

Рис. 12. Связывание списков по хеш-функции К mod 5

1) Позиция 0 в хеш-таблице 0=50 mod 5
2) Позиция 1 в хеш-таблице 1=46 mod 5
3) Позиция 2 в хеш-таблице 2=72 mod 5
4) Позиция 3 в хеш-таблице 3=13 mod 5
5) Позиция 4 в хеш-таблице 4=99 mod 5
6) Позиция 0 в хеш-таблице 0=10 mod 5,а также 25 mod 5 и 30 mod 5
7) Позиция 3 в хеш-таблице 3=68 mod 5 и 93 mod 5.

Известно достаточно много хеш-функций. Простейшая из них имеет вид

$$i = p\text{-}a$$

где i- номер(адрес) списка

p-значение ключево атрибута записи.

a-константа

Если известно минимальное значение ключевого атрибута P_1, и номер списка $i_{список}$, соответствующего ключу P_1, то тогда константа $a=P_1 - i_{список}$.

Индексы

Для ускорения поиска записей в массиве используется дополнительная информация, организованная в виде массива индексов.

Индексом называется набор ключей и адресов записей, которые выбираются из основного массива по определенному алгоритму.

Пример1. Запись с ключом q попадает в индекс с номером

$$n = \frac{q - P(1)}{Z} + 1,$$

где P(1) – значение ключа первой записи;

Z – константа (шаг арифметической прогрессии)

Пример 2: Запись с номером q попадает в индекс с номером

$$n = P(1) + z(q - 1)$$

Над исходными отношениями до проведения соединения меняя взаимный порядок требуемых соединений.

Индексный способ доступа.

Индекс представляет собой механизм быстрого доступа к данным.

Сущность индексов состоит в том, что они хранят значения индексных полей и указатель на запись в таблице.Например.

Таблица 1.

Порядковый номер записи	Дата	Товар	Количество
1	10.01	сахар	10
2	12.01	картофель	20
3	13.01	свекла	10
4	14.01	сахар	15
5	16.01	слива	20

Индексы таблицы

Таблица 2.

По дате		По товару		По количеству	
Дата	№ записи	Товар	№ записи	Колич.	№ записи
10.01	1	Картоф.	2	20	
12.01	2	Сахар	1	10	
13.01	3	Сахар	4	15	
14.01	4	Свекла	3	20	

Хэш-таблица (hashtables)- одно из величайших изобретений информатики. Сочетание массивов и списков с небольшой добавкой математики позволила создать эффективную структуру для хранения и получения динамических данных.. Типичное применение хэш-таблиц – символьная таблица, которая связывает некоторые значения (данные) с каждым членом динамической таблицы строк. Для получения хэш-значения,

необходимо пропустить ключ. через хэш-функцию, которая используется как индекс в таблице, где храниться информация.

Хэш-таблица – это массив списков, который сцепляет вместе элементы, имеющие общее хэш-значение.

С помощью схемы хеширования отображают большее кол-во возможных ключей в относительно компактной хеш-таблице, т.е устанавливают соответствие между N – количеством записей и K – позицией таблицы.

18. СОРТИРОВКА

Сортировка (sorting) – один из наиболее сложных для изучения алгоритмов. Во-первых, сортировка – это общая задача многих компьютерных приложений. Практически любой список данных ценнее, когда он отсортирован по какому-либо определенному принципу. Часто требуется, чтобы данные были упорядочены несколькими различными способами.

Во-вторых, многие алгоритмы сортировки являются интересными примерами программирования, поскольку демонстрируют важные методы, такие как частное упорядочение, рекурсия, объединение списков и сокращение двоичных деревьев в массивах.

У каждого алгоритма сортировки есть свои преимущества и недостатки. Производительность различных алгоритмов зависит от типа данных, начального расположения, размера и значений данных и, наконец, сортировка – одна из немногих задач с точными теоретическими границами производительности. Любой алгоритм сортировки, который использует сравнения, занимает, по крайней мере, N log N времени, где N-количество сортируемых данных.

Существуют алгоритмы, которые осуществляют сортировку не с помощью сравнений, и при этом работают быстрее чем N log N.

Общие принципы сортировки

При сортировке элементов программа перестраивает их в некоторую структуру данных. Скорость этого процесса зависит от типа элементов. Перемещение целого числа на новую позицию в массиве может произойти намного быстрее, чем перемещение структуры данных, определяемой пользователем. Если структура данных являются записью, содержащей тысячи байт данных, то перемещение одного элемента может занять достаточно много времени. Гораздо проще сортировать указатели из одной части массивы в другую.

Чтобы отсортировать массив объектов в определенном порядке, можно создать массив указателей на объекты. Затем сортировать указатели с помощью значений в соответствующих записях данных.

Пример. Требуется отсортировать записи о служащих, определенных следующей структурой

```
type
    PEmployee = ^ TEmployee;
```

```
        TEmployee = Record
            ID: Integer;
            LastName : String [yo];
            ForstName : String [yo];
                end;
var
EmployeeData : array [I …N] of TEmployee;
```

Создадим массив указателей на данные служащего.

```
var
IDIndex : array [ '..N] OF PEmployee;
```

Составим цикл, так чтобы первый элемент индекса указывал на первую запись данных, второй – на вторую запись данных и т.д

.

```
For i: = 1 to N do IDIndex [i]: = @EmployeeData [i];
```

Сортируя массив индексов по идентификационному номеру, получим возможность по индексному элементу указывать на соответствующую запись данных в заданном порядке.

Иногда для сортировки удобнее хранить ключи списка в комбинированной форме. Например, можно объединить (combine) ключевые элементы списка в виде переменной.

```
combine_name: = Format ('%S, %S', [new_Last_name, new_First_name]);
```

Популярные методы сортировки строк – кодирование их целыми числами или данными другого числового формата.

Числовые типы данных занимают меньше места, и компьютер может сравнивать два числовых значения намного быстрее, чем две строки. Обычные строковые операции не выполняют числового кодирования, поэтому необходимо перевести строку в кодированную форму и обратно при необходимости. Например, кодирование прописных английских букв из 26 букв алфавита выполняются по алгоритму.- Будем считать, что каждый символ – это число по основанию 27 (т.е. 26+1), где 1 – отметка конца слова.

1. Если в строке три символа и слово FOX, то код слова FOX равен $27^2\times(F-A+1)+27\times(O-A+1)+(X-A+1)=4803$
2. Если в строке два символа и слово NO $27^2\times(N-A+1)+27\times(0-A+1)+(0)=10611$

Таким образом, можно кодировать строки из n прописных букв в длинное целое (Longint) и в двойное число с плавающей точкой (Double).

Пример преобразования строки в числа формата Double и обратно. (27 – это основание равное 26+1, каждый символ – это число с основанием 27)

```
    String_Base=27;
    ASC_A=65;                    //ASCII код для 'A'
    full_len=3;
Function TEncode Form.. StringToDbl (txt: String; full_len : integer) : Double;
    var
        len, i : integer
        cn : Char;
```

```
begin
len : = length (txt);
IF (len > full_len) then len : = full_len;
     Result : = 0.0;
For I : = 1 to len do
  begin
  ch : = txt [i];
Result : = Result × String_Base + ord (ch) – ASC_A+1; end; lnd;
```

Преобразование кода в строку

```
Function  TEncodeForm.. DblToString (value : Double) : String;
  var
    ch : Integer;
    new_value : Double;
    begin
    Result : = ‘ ‘;
    While (value > 0) do
    begin
    new_value : = Round (value/STRING_BASE);
    ch : = Round (value_new_value*STRING_BASE);
    IF (ch < > 0) then
    Result : = chr (ch + ASC_A-1)+Result;
    Value : = new_Value;
    end;
    lnd.
```

Исходная программа позволяет создавать список из строк и сортировать их с помощью числового кодирования.

Методы сортировки

1. **Сортировка выбором**. Задача этого метода – искать наименьший элемент, который затем меняется местами с элементами из начала списка. Затем находится наименьший из оставшихся элементов и меняется местами со вторым элементом. Процесс продолжается до тех пор, пока все элементы не займут свое конечное положение.
2. **Сортировка вставкой**. Задача состоит в том, чтобы сформировать новый сортированный список, присматривая все элементы в исходном списке в обратном порядке. Алгоритм присматривает исходный список в порядке возрастания и ищет место, где необходимо вставить новый элемент. Затем он помещает новый элемент в найденную позицию.
3. **Вставка в связанных списках**. Метод позволяет упорядочивать элементы не в массиве, а в связанном списке.

Алгоритм ищет позицию нового элемента в возрастающем связанном списке и затем помещает туда новый элемент, используя операции работ со связанными списками. Версию для связанных списков лучше использовать, когда программа уже хранит элементы в связанном списке. Другие принципы

действия заложены в алгоритмах пузырьковой, быстрой, пирамидальной и т.д. сортировках.

Резюме. Зная структуру данных и различные алгоритмы сортировки, можно выбрать тот алгоритм, который лучше всего подходит для решения конкретной задачи.

19. Базы данных и знаний

Система понятий для представления знаний существенно отличается от понятий для представления данных, поэтому знания отображаются в базе знаний, а данные хранятся в базе данных. Вместе с тем база знаний способна хранить данные как простую разновидность знаний.

Итак, данные – это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Знания могут храниться в памяти человека, на материальных носителях (в учебниках) и на машинных носителях.

База знаний – основа любой интеллектуальной системы. Знания могут быть поверхностными – то есть знания о видимых взаимосвязях между отдельными событиями и фактами.

Знания глубинные – это абстракции, схемы. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объекта.

Модели представления знаний

Основные модели:

1. Продукционная модель;
2. Семантическая сеть;
3. Фреймы;
4. Формальные логические модели.

Продукционная модель основана на правилах вида:
Если (условие), то (действие).Под условием понимается некоторое предложение – образец, по которому осуществляется поиск в базе знаний, а под действием – действия, выполняемые при успешном исходе поиска.

Семантическая сеть – это ориентированный граф, вершины которого понятия, а дуги – отношения между ними. В качестве понятий выступают объекты, а отношения – это связи типа: это является частью, или это принадлежит. Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети,

соответствующего некоторой подсети, отражающей поставленный вопрос к базе знаний.

Фреймы – это вид структуры знаний для восприятия пространственных сцен и обозначают «Каркас» или «Рамку» сцены. Фрейм – это абстрактный образ для представления некоего стереотипа восприятия или формализованная модель для отображения образа.
Например. Произнесенное слово «комната» порождает у слушающих образ комнаты, как жилого помещения с четырьмя стенами, полом, потолком, окнами и дверью, то есть образ комнаты – это фрейм комнаты.
Различают фреймы – образцы или прототипы, хранящиеся в базе знаний и фреймы – экземпляры, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных.
Для обозначения объектов и понятий используются фреймы – структуры, для действий – фреймы роли, фреймы - ситуации, фреймы - сценарии и т.д.

Формальные логические модели основаны на классическом исчислении предикатов 1-го порядка, когда предметная область или задача описывается в виде набора аксиом.

20. Метод сущность-связь

Метод сущность – связь называют также методом «ER - диаграмм» , где E – essence(сущность), R – relation (связь).

Основные понятия метода

Основными понятиями являются следующие**:**

1. Сущность
2. Атрибут сущности
3. Ключ сущности
4. Связь между сущностями
5. Степень связи
6. Диаграммы ER – типа

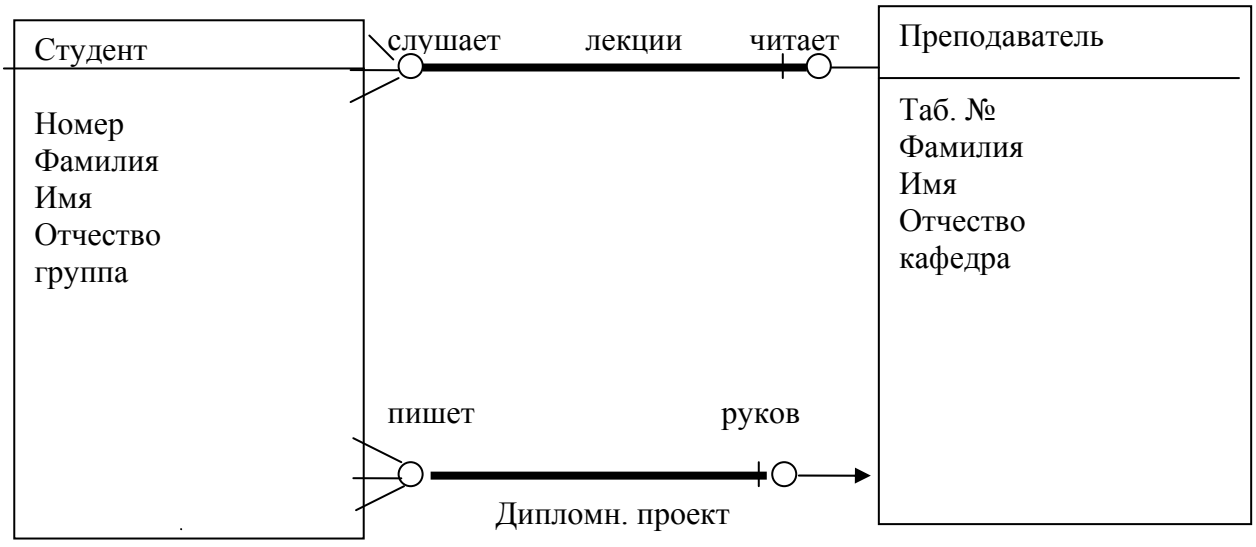
Сущность – это объект, информация о котором хранится в БД. Названиями сущностей являются, как правило, существительные.
Атрибут сущности представляет собой свойства сущности.
Ключ сущности – это атрибут или набор атрибутов, используемый для идентификации экземпляра сущности т. е совпадает с определением ключа отношения .Связь между сущностями предполагает зависимость между атрибутами этих сущностей. Название связи обычно представляется глаголом.

С целью повышения наглядности и удобства проектирования для представления сущностей и связей между ними используются графические средства типа ER – диаграмм.(см.рис.13)

Рис.13. Диаграмма ER – типа

На основе диаграмм ER – типа формулируется отношения проектируемой БД.
Степень связи является характеристикой связи между сущностями, которая может быть типа:
1:1; 1:M; M:1; M:M.
Класс принадлежности сущности может быть обязательным, если все экземпляры этой сущности обязательно участвуют в рассматриваемой связи, в противном случае класс принадлежности сущности является необязательным.

Пример отношения «один – ко – многим» при связывани сущностей – Студент и преподаватель.



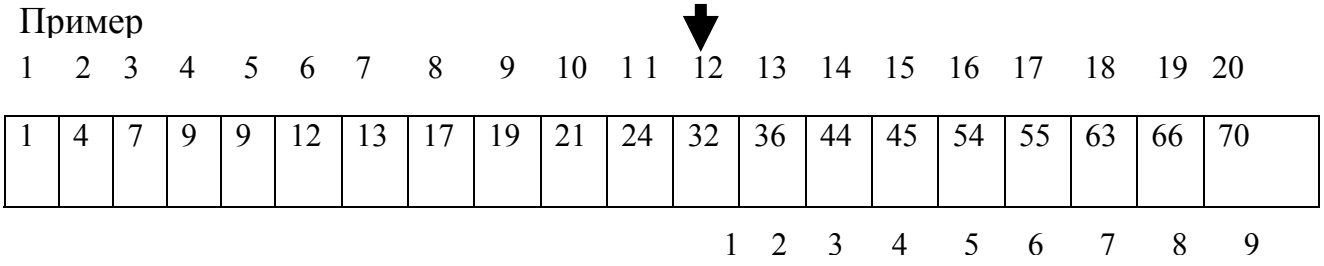
Связь со стороны Преподаватель – один ко многим, связь со стороны Студент – многие к одному, связь между сущностями Студент и Дисципина – многие ко многим.

Обязательность и необязателность связи на диаграмме может обозначаться по разному. Например, необязательная связь – кружочком, обязательная – чертой (перпендикулярной линией, перечеркивающей связь).
Резюме. Семантические сети используются в качестве структуры, пригодной для представления информации общего вида в терминах естественного язака общения. При этом узлы представляют некоторые понятия, а связи отношения между ними..

21.Методы ускорения поиска (доступа) к данным

Алгоритм поиска, использующий интерполяцию, ускоряет процесс поиска элемента в списке, путём предсказания неизвестных значений на основе имеющихся. При использовании индексов известных значений в списке, можно определить индеккс искомого значения

Пример



Интерполяционный поиск делит список так, чтобы найти ближайший к искомому элементу в списке. При этом точка разбиения определяется следующим образом

d=Round(min+(иск-L(min))*(max-min)/(L(max)-L(min))

Пример

d=Round(1+(44-1)*(20-1)/(70-1)=Round(1+43)(19)/(69)=12

d^{+}=Round(1+(44-1)*(9-1)/(70-32)=Round(1+12)(8)/38=2.73

Образуем сумму 12+2,73=14 (с округлением).

Сравнивая искомые значения со значениями в известных позициях, алгоритм может определить позицию, где должен быть искомый элемент. В сущности, создается функция, которая вводит соответственно между искомым значением и индексом позиции, где он должен находиться. Если первое предположение не верно, алгоритм снова использует эту функцию, чтобы сделать новое предположение и так далее до тех пор, пока находит искомый элемент.

В алгоритме хеширования используется подобный принцип для преобразования элементов в хеш-таблицу. При помощи функции алгоритм хеширования определяет положение элемента в таблице на основе значения искомого элемента

Схема хеширования устанавливается соответственно между записями и позиции в таблице

Литература

1.Мишенин А.И. Теория экономических информационных систем.- М.: Финансы и статистика ,2007.- 237с.

2. Мишенин А.И, Салмин С.П. Теория экономических информационных систем. Практикум – М.:Финансы и статистика 2005.- 190 с.