УПРАВЛЕНИЕ ДАННЫМИ

Доронина Юлия Валентиновна

Лекция № 1

**Основные понятия:**

Информационные системы – системы, которые позволяют пользователю хранить информацию, обрабатывать и обмениваться ей. Информационные системы делятся на 2 класса:

* Информационное обеспечение
* Системы с самостоятельным целевым назначением

К 1 классу относятся САПР (система автоматизированного проектирования) АСУП (автоматизированная система управления предприятием).

Ко 2-й группе систем относятся системы информационно-справочные (ИСС), информационно-поисковые (ИПС), информационно-управляющие системы (ИУС). В ИСС и ИПС информация представляется в определенном формате, который удовлетворяет пользователя и выдается по его требованию.Классификация ИС:

1. По типу данных:
   1. Фактографические
   2. Документальные
2. По степени автоматизации:
   1. Автоматические
   2. Автоматизированные
   3. Вручную
3. По сфере применения
4. По характеру обработки данных:
   1. Интеллектуальная
   2. Статическая
5. По характеристике управления:
   1. Стратегические
   2. Оперативные
   3. Функциональные

ИС – техническое или аппаратное обеспечение.

Банк данных – записанныйе в памяти системы по одним критериям информационные данные одной предметной области с отражением взаимосвязей и организацией прямого доступа.

Данные – информация, фиксированная в определенной форме для последующей передачи, обработки и хранения.

2 Аспекта данных:

* Инфологический (связан со смысловым содержанием без учета представления их в памяти машины)
* Даталогический (связан со смысловым содержанием с представлением в памяти машины)

БД – совокупность взаимосвязанных хранящихся вместе данных при наличии такой минимальной их избыточности, которая допускает их оптимальное использование для одного или нескольких приложений, при этом данные организуются не зависимо от программных средств, а добавление или модификация осуществляется общим управляющим способом.

СУБД – программное обеспечение для использования или модификации БД одним или несколькими лицами.

Функции СУБД:

* **Обеспечить пользователя инструментарием, позволяющим оперировать данными в абстрактной форме не связанной со способом хранения.**
* Обеспечение секретности
* Обеспечение целостности
* Защита от отказов
* Обеспечение синхронизации

ЯОД – язык высокого уровня, предназначенный для задания схемы базы данных. То есть описывающий данные, структуру. Декларативный язык.

CREATE TABLE EMP

(EMP NUM INT NOT\_NULL, L\_ NAME CHAR (10))

ЯМД – язык манипулирования данными – язык запросов, система команд для управления данными. ЯМД различают СУБД с включенным языком и СУБД с базовым языком.

ЯМД

SELECT \* FROM EMP

Словарь данных – специальная система, предназначенная для хранения централизованной информации обо всех ресурсах БД.

Администратор БД – человек, реализующий управление БД.

**Этапы развития управления данными:**

1. 60-е - логические файлы данных связывались с физическими файлами и системой ввода-вывода. Особенности: данные располагаются последовательно, не реальный режим времени.
2. Конец 60-х - методы доступа: 1 логический файл и несколько физических файлов данных. Особенности: не зависимость прикладных программ от аппаратных средств; зависимость между логической и физической структурой данных; прямые, последовательные, индексно-последовательные методы доступа.
3. 70-е – управление данными: логические файлы прикладных программистов связываются с физической БД – первые СУБД. Особенности: увеличение объема информации, элементы данных – это поля записи, независимость логической и физической структур данных, независимость прикладных программных систем и данных.
4. 80-90-2000-е – глобальное логическое представление данных: независимость данных. Особенности: 2 уровня независимости данных (логический и физический), возможность работы в реальном режиме времени, рост объема данных при онлайн обработке.
5. 2000-е – наши дни – гетерогенные, объектно-ориентированные, пост реляционные БД.

Лекция №2

**Требования к организации управления данными:**

* Пользовательские требование:
  + **Многократное использование данных**
  + Низкая стоимость
  + Скорость обмена данными
  + Простота внесения изменений
  + Секретность
  + Защита от уничтожения
  + Производительность
* Функциональные требования:
  + **Физическая и логическая независимость данных**
  + **Контролируемая избыточность**
  + Обеспечение требуемой скорости поиска
  + Возможность настройки и привлечение вспомогательных средств

**Проектирование структур данных**

Цель: создать точную и защищенную БД, на основе которой можно гарантировать эффективное построение прикладных программ.

Этап 1: Анализ предметной области (АПрО), информационные потребности пользователя (ИПП) (требуются исходные данные) – Инфологическое проектирование.

Этап 2:Логическое проектирование. Внешняя модель переходит при построении в концептуальную модель. Обеспечиваются 3 типами данных (сетевая, древовидная, реляционная) даталогическое проектирование.

Этап 3: Внутренняя модель. Физическое проектирование. Структура манипулирования.

Все это готовая схема БД.

При проектировании проекта БД рассматривают 3 уровня абстракции

* Физический
* Концептуальный
* Уровень представлений

**Терминология**

**Объект** – это элемент информации, который сохраняется и им можно управлять.

**Атрибут** – свойство объекта.

**Домен** – множество атомарных значений одного атрибута.

**Атомарное значение** – единое значение.

**Отношение** – это таблица, обладающая следующим свойствами:

* Каждый элемент таблицы представляет собой один элемент данных (свойство отсутствия повторяющихся групп)
* Все столбцы однородны
* Столбцам однозначно присвоены номера
* В таблице нет одинаковых строк
* В операциях с такой таблицей все строки и столбцы могут рассматриваться в любом порядке

**Ключ** – это атрибут или совокупность атрибутов, используемых для идентификации кортежа (строки).

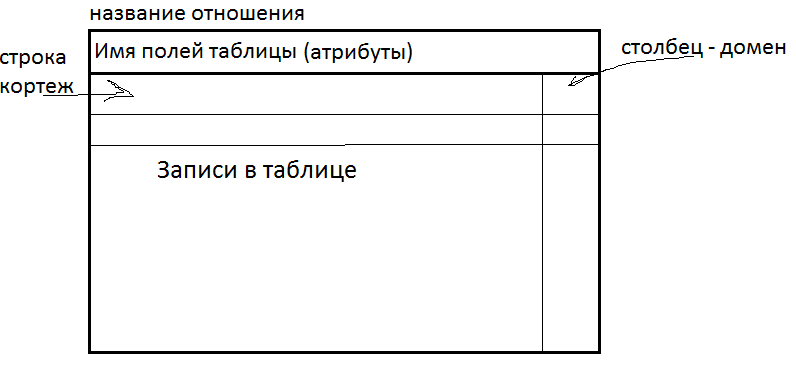
**Основной ключ** (первичный) – ключ используется для однозначной уникальной идентификации кортежа.

**Дополнительный ключ** – уникальную запись не идентифицирует, но указывает множество записей, обладающих общим свойством.

**Внешний ключ** – атрибут ил комбинация атрибутов одного отношения, которые содержатся в другом отношении и обладают свойствами:

* Являются первичными ключами в 1 и 2 отношении
* Являются в первом первичные, а во втором обычные атрибуты

**Экземпляр записи** – конкретное значение в текущей строке.



Существуют 2 способа представление информации в отношении

* Первый способ – прямой (каждый кортеж содержит все атрибуты)
* Второй способ – инверсный (необходим для идентификатора объекта связанного со вторым объектом)

**Описание данных для проектирования**

* + 1. *ER – диаграммы*

*Принятые обозначения:*

1. прямоугольник обозначает объект;
2. овал - атрибут;
3. ромб - связь;
4. объекты и атрибуты связаны ребрами, связи между объектами обозначаются дугами;
5. ключевой атрибут подчеркивается.

ПОСТАВЩИК

поставляет

поставки

Рис. 12.

* + 1. *Диаграммы Бахмана*

*Принятые обозначения:*

1. прямоугольник - объект, внутри прямоугольника пишется имя объекта и в круглых скобках могут быть перечислены атрибуты;
2. объекты связаны между собой направленными ребрами, связи подписываются;
3. диаграммы предназначены для древовидных, сетевых структур, поэтому всегда существует исходный объект и подчиненный объект.

|  |
| --- |
| Организация (наименование, адрес) |

работает

|  |
| --- |
| сотрудник (ФИО) |

Рис. 13.

* + 1. *Овал – диаграммы*

*Принятые обозначения:*

1. атрибуты представляются в овалах, ключевой атрибут подчеркивается;
2. связи представляются в виде направленных ребер между овалами.

Число членов экипажа Количество мест в самолете Тип самолета

№ самолета

№ рейса

Количество свободных мест Пункт отправления Пункт назначения

**Схемы и подсхемы**

Схемой называется описание логической БД. Представляет собой наброски будущих таблиц без включения данных

Правила построения схемы

* Отображать различия между именами элементов и атрибутов
* Отмечать идентификаторы связи
* Отличать связи от перекрестных ссылок
* Изображать объединение элементов данных в агрегаты

В процессе построения схемы БД возможно разделение заданий для проектировщиков, которые отображаются в форме подсхем.

**Типы связей**

Бинарные и множественные

Бинарные связи:

* Один к одному
* Один ко многим
* Многие ко многим

Множественные связи:

* Многие ко многим

Если между объектами существуют несколько типов отношений, то связь должна быть отмечена специальным образом.

Виды связей:

* Несвязанные
* Взаимозаменяющиеся
* Связанные

Существует развязка множественных отношений.

В процессе формирования проекта БД необходимо организовать связь между двумя подсистемами. Т. е. решить задачу развязки отношения многие ко многим и неопределенности, связанной с похожей документации спецификаций. Обычно это производится введением таблицы ключевых атрибутов ключевых отношений.

Типы структур БД:

* Сетевая
* Иерархическая
* Реляционная

**Иерархическая структура**. В основе лежит схема дерева, которое представляет собой иерархию элементов, называемых узлами. При этом дерево определяется как иерархия узлов с двоичными связями такими, что самый верхний единственный и называется корнем. Все узлы кроме корня связаны только с 1 узлом на более высоком уровне. Файл, организованный по этому принципу называется иерархическим. Бывают 2-х типов: однородные и неоднородные.

Однородные – один и тот же тип связи.

Неоднородные

Отдел

Помещения

Сотрудники

Работа

Дерево может быть переменной или фиксированной глубины.

Обычно СУБД работают с неоднородными деревьями фиксированной глубины.

**Сетевая структура**. Если порожденные элементы имеют более одного исходного, то такая структура описывается в форме сети. Структура называется сетевой, если есть 3 признака:

* Если у порожденного элемента есть более 1 родителя
* Есть связь «многие ко многим»
* Наличие петель
* Наличие циклов

Петля – это цикл из одного типа записи. Простая сетевая структура: 1,3,4. Сложная сетевая структура:1-4.

Преобразование сложной сетевой структуры в простую введением дополнительной избыточности. Добавляется отношение с 2 ключами. Это называется развязкой отношений «многие ко многим».

Преобразование простой сетевой структуры в древовидную осуществляется посредствам введения избыточности, то есть дублирование отношения входящего в ПСС связь.

Преобразование петель и циклов.

Вводим дополнительное отношение из заданных.

Существует классификация категорий схем.

1. Двухуровневые схемы:
   1. Без ветвления
   2. С ветвлением
2. Многоуровневые схемы
   1. Без ветвления
   2. С ветвлением
   3. С циклом

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Двухуровневые***  ***Схемы*** | | ***многоуровневые***  ***схемы*** | | ***схема с***  ***циклом*** |
| ***без ветвления*** | ***с ветвлением*** | ***без ветвления*** | ***с ветвлением*** |
| древо-видные  струк-туры |  |  |  |  |  |
| простые сетевые |  |  |  |  |  |
| сложные сетевые |  |  |  |  |  |
| петли (одно-уровне-вые циклы) |  |  |  |  |  |

**Реляционная модель.** Понятие РБД строится на следующих основных принципах:

* Целостность отношений (связь по внешним ключам)
* Нормализация отношений – оптимизация

Основой РБД являются плоские таблицы. Процесс приведение древовидной структуры к реляционной – оптимизация (Одд). Общая задача нормализация – это введение и устранение избыточности.

РБД – совокупность отношений с минимальной избыточностью. Т. к. отношение – это множество кортежей и n-столбцов, то такое отношение называется отношение степени n.Бинарные, тринарные, n-арные отношения.

Получение 1-ой нормальной формы из древовидной структуры.

Чтобы отношение соответствовала 1НФ необходимо привести в соответствии 3 условиям:

* Определить первичные ключи
* Устранить повторяющиеся группы
* Привести поля к атомарности

Для 1НФ применяется правило распространения ключей древовидной структуры – ключ из вышестоящего отношения уходит к нижнему отношению.

Свойства, которыми должен обладать ключ:

* Однозначная идентификация кортежа
* Отсутствие избыточности (никакой атрибут нельзя удалить из ключа не нарушая свойства идентификации кортежа)

Для одного отношения может существовать несколько наборов атрибутов, обладающих этими свойствами. Эти наборы называют множеством возможных ключей. Из них выбирают первичный или основной.

**Анализ полей на атомарность**

Атомарность – каждое поле содержит только одно значение данных.

Существует 2 вида не атомарности:

* Горизонтальная (ФИО - 1 поле)
* Вертикальная (в одном поле содержится много составляющих)

Составной ключ не является не атомарностью

**№ рейса дата время** место

Избавление от не атомарности заключается в разбиении поля на отдельные элементы или дублировании части кортежа соответствующему первому элементу.

Вертикальная не атомарность может иметь место для одного или нескольких полей кортежа

**Сравнение моделей на базе сетевой, иерархической и табличной структуры**

БД классифицируются:

1. по объему сравнивая количества записей:
   1. малые около 1000
   2. средние около 10000
   3. большие около 100000
   4. крупные свыше 100000
2. по способу манипулирования
   1. в табличных моделях существует реляционная алгебра
   2. в сетевых и иерархических поиск навигационного типа

Для сравнения моделей существует 2 критерия

* Применение – связан с издержками в малых БД которые приходятся на написание прикладных программ и формирование запросов
* Эффективность – издержки приходятся на стоимость памяти и машинного времени для больших БД

Для больших и крупных БД используют сетевую или иерархическую структуру

Для малых и средних – реляционную.

Критерием принятия решения о выборе структуры БД является вид запросов. В случае сложных запросов используют реляционную.

**Теория проектирование Реляционных БД**

Реляционная БД – это набор экземпляров конечных отношений.

Схема реляционной БД может быть представлена в виде:

R1(PK1,A11,A12,…,A1m)

R2(PK2,A21,A22,…,A2r)

…

Rn(PKn,An1,An2,…,Ant)

Главная задача проектирования РБД – понизить избыточность и повысить надежность.

Под надежность понимается целостность данных – способность БД по любому запросу выдать данные, которые требуются.

Существуют 2 вида ограничений на зависимости между атрибутами отношений:

* Зависит от семантики элементов домена (когда значение элемента должно находиться в диапазоне либо зависимость между атрибутами)
* Равенство значений атрибутов различных отношений (внешние ключи)

Таким ограничениям относят функциональные, многозначные зависимости и зависимости соединений.

Функциональные зависимости - F-зависимости

A, B, C – атрибуты

Z, X, Y – множества

a, b, c, z, x, y – значения атрибутов

RSVW – таблицы

XY, XZ, YZ – объединения

U – таблица, состоящая из всех атрибутов всех доменов.

Пусть X, Y атрибуты отношения R. Атрибут Y функционально зависит от атрибута X, если в каждый момент времени каждому элементу множества X соответствует элемент множества Y.

X→Y

X#Y – связи нет

X→Y и Y→X существует взаимоопределяющее соответствие X↔Y

Определение полной функциональной зависимости. Пусть существует ZcX; X,YcR. Атрибут Y отношения R называется полностью зависимым от атрибута X если он зависит от всех атрибутов множества X но не зависит от подмножества Z. При наличии подмножества X определяющего Y можно говорить о нарушении правила минимальности ключа.

Аксиомы функциональных зависимостей.

№1

Пусть для R задано множество F функциональных зависимостей F1, F2,…,Fn. Замыкание множества F+ множество всех зависимостей, которые выводятся из множества F. При решении этой задачи необходимо иметь правила вывода. Множество таких правил должно обладать свойством полноты (на заданном множестве F правила позволяют вывести все зависимости принадлежащие F+) и надежности(Используя эти правила не выведется зависимость не принадлежащую F+).

**Аксиома 1**(*рефлексивность*):

если X ⊆ Y ⊆ U то X →Y, такие зависимости называют тривиальными (то есть правая часть содержится в левой: А → А, АС → А).

**Аксиома 2** (*пополнение*):

если X →Y и Z ⊂ U, то ZX → ZY

**Аксиома 3** (*транзитивность*):

если X→Y и Y → Z. то X → Z

Кроме этих аксиом, существуют правила:

**Правило 1** (*объединение*):

если X → Y и X → Z , то X → YZ.

**Правило 2** (*псевдотранзитивность*):

если X → Y и WY → Z , то WX → Z

**Правило 3** (*декомпозиция*):

если X → Y и Z ⊆ Y , то X → Z.

Лемма №1 аксиомы Армстронга являются надежными

Лемма №2 функциональная зависимость X→Y следует из аксиом Армстронга, только если Y является подмножеством замыкания X, при этом X+ множество атрибутов А таких, что зависимость X→А может быть выведена по аксиомам Армстронга.

**Вычисление замыканий функциональных зависимостей**

Чтобы получить множество F+ надо сформировать множество зависимостей вида X->Y F={x1->b1, x2->b2 … }. Алгоритм построения замыкания

X+ (замыкание множества атрибутов)

Дано:

F, U, X

Найти:

X(0), X(1), …

Алгоритм:

1)X(0)=X, i=0;

2)x(i+1)=>X(i) < F последовательное присоединение атрибутов из правой части F, в левой части которых стоят элементы подмножества X(i);

3)i++;

4)переход на п2 пока X(i+1)#X(i) # - невозможность присоединения;

5)X+=X(i+1);

Таким образом, исходное сочетание атрибутов БД определяет исходную комбинацию из X+ а следовательно X+ построена, а значит может считаться построенным F+.

Покрытие множеств F.

Пусть F и G множества зависимостей универсального отношения. F и G эквивалентны если совпадают множества F+ и G+. Проверка заключается в определении принадлежит ли G+некоторая зависимость из F.

(X->Y) c F, Є G+

(X+) c G+ если Y Є X+

(X->Y) Є G+

(F) c G+ если F Є G+, то G перекрывает F

Если в ходе проверки выяснилось, что F перекрывает G, то F=G(эквивалентны)

Условие минимальности множества зависимостей

* Все зависимости имеют вид X->A(в правой части стоит только 1 элемент)
* Ни для какой зависимости X->A ее удаление не приведет к получению эквивалентного множества.
* Ни для какой зависимости X->A не выполняется условие эквивалентности (F-{X->A}) U (Z->A), где Z Є X

Декомпозиция схемы отношений.

Существуют 2 основные операции над БД:

Выделение столбцов

Соединение столбцов в горизонтальное сцепление.

Понятие декомпозиции схемы и ее свойства.

Декомпозиция схемы отношения R содержащей атрибуты представляется множеством отношений ƥ={R1,R2,…,Rm} таких, что их соединение равно исходному R, причем Ri необязательно непересекающиеся.

* Свойство соединения без потерь информации
* Свойство сохранения зависимостей

Не каждая декомпозиция обладает этими 2 свойствами. Говорят, что декомпозиция обладает свойством соединения без потерь информации, если естественное соединение дает исходное отношение. Говорят, что декомпозиция обладает свойством сохранения зависимостей, если множество зависимостей выводимо из схемы отношений.  
Проверка свойства соединения без потерь

* Создание матрицы из m-столбцов и n-строк.
* Столбец j соответствует атрибуту Aj. Строке I соответствует схема отношений Ri. На пересечении образуется элемент aj если атрибут Aj Є Ri иначе ставится bj.
* Анализируем зависимость Х определяем У ищем строки для которой совпадают значения по всем столбцам соответствующего Х если найдены то отождествляем символ в строках соответствующего У, при этом если оба символа b маленькие, то они приравниваются одному из символов bik bil.
* Если какая либо строка состоит только из символов a то эта декомпозиция обладает искомым свойством.

Теорема используется для декомпозиции на 2 отношения

R1

Проекция F на множество атрибутов Z называется множество зависимостей Х->У , таких что соединение

* Сохранение зависимостей необходимо, т. к. при нарушении этого свойства могут обнаружиться такие текущие значения схемы отношений, которые не удовлетворяют множеству функциональных зависимостей.
* Каждое обновление схемы отношений потребовало бы осуществление соединения для проверки выполнения ограничения целостности.

Пример: дано R={G, A, I};G - город; A - адрес; I - индекс;

F = {GA→I, I→G}

ρ = {R1, R2}, R1 = {A, I}, R2 = {I, G}

R1∩R2 = I R2-R1 = G I→G

Декомпозиция ρ обладает свойством сохранения зависимостей, если из объединения всех зависимостей, принадлежащих проекции F на Ri (i=1...n), логически следуют все зависимости F.

Декомпозиция обладает свойством соединения без потерь, но не обладает свойством сохранения зависимостей, поскольку, не может быть получена зависимость GA→I.

**Многозначные зависимости (MV - зависимости)**

*Определение многозначной зависимости*

Пусть дано отношение R и множество атрибутов X, Y, которые являются подмножеством R

X, Y ⊆ R.

Говорят, что X мультиопределяет Y

f : X →→ Y,

если заданному значению X соответствует множество, состоящее из ∅ или более значений Y, причем Y значения никаким образом не связаны с множеством атрибутов R - X - Y.

Если имеется многозначная зависимость, то для нее должны выполняться следующие условия:

Пусть существует два кортежа s и t в отношении R и X→→Y, и пусть s [X] =t [X] , тогда должно существовать два других кортежа v и u таких, что:

1. u [X] = v [X] = s[ X] = t [X];
2. u [Y] = s [Y] u [R-X-Y] = t [R-X-Y];
3. v [Y] = t[Y] v [R-X-Y] = s [R-X-Y].

То есть, если поменять в кортежах s и t местами значения атрибутов множества Y, то получим два кортежа, которые также должны принадлежать отношению R.

**Пример:**

Пусть дана схема отношений R={C, T, H, S, G, A}, где C - курс; T - учитель; H - день недели и время; S - студент; G - оценка; A - аудитория, где проводятся занятия.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C | T | H | S | G | A |
| S | 1 | Иванов | понедельник - 900 | Петров | 5 | 420 |
| U | 1 | Иванов | понедельник - 900 | Сидоров | 4 | 420 |
| T | 1 | Иванов | среда - 1200 | Петров | 5 | 421 |
| V | 1 | Иванов | среда - 1200 | Сидоров | 4 | 421 |

Рис. 29.

Здесь существует многозначная зависимость C→→HA.

Многозначные зависимости определяются только семантикой отношения.

* + 1. *Аксиомы многозначной зависимости*

**Аксиома 4**: (дополнения для многозначных зависимостей)

если X→→Y, то X→→U-X-Y

**Аксиома 5**: (пополнение для многозначных зависимостей)

если X→→Y V⊆W, то XW→→VY .

**Аксиома 6**: (транзитивность для многозначных зависимостей)

если X→→Y Y→→Z , то X→→Z-Y

Существуют две аксиомы, связывающие многозначные зависимости с функциональными зависимостями.

**Аксиома 7**:

если X → Y, то X→→Y

то есть функциональная зависимость является частным случаем многозначной зависимости.

**Аксиома 8**:

если X→→Y , Z⊆Y, W не пересекается с Y, W→Z

то X→Z .

***Многозначные зависимости*** - это зависимости между ключевыми атрибутами.

***Теорема*** *:*

аксиомы 1 - 8 являются надежными и полными для функциональных и многозначных зависимостей .

Пусть D - множество функциональных и многозначных зависимостей, рассматриваемых на множестве атрибутов.

D+ - множество функциональных и многозначных зависимостей, выводимых из D, то есть каждое отношение на V, которое удовлетворяет D, удовлетворяет также зависимостям из D+, тогда D+ в точности является множеством зависимостей, которые выводятся из D по аксиомам 1 - 8.

***Правила для многозначных зависимостей:***

1. объединения

если X →→Y, X→→Z ,то X→→YZ;

1. псевдотранзитивности

если X→→Y , WY→→Z ,то WX→→Z - WY;

1. смешанное правило транзитивности

если X→→Y, XY→Z ,то X→Z - Y;

1. правило декомпозиции

если X→→Y, X→→Z ,то X→→ Y ∩ Z, X →→ Y - Z, X →→ Z - Y;

***Теорема*** :

Пусть U - множество всех атрибутов, тогда можно построить разбиения множества U - X на подмножества Y1, Y2, ..., Yk, такие что для Z, являющимся Z ⊆ U - X, имеем X→→Z если и только если Z являестся объединением некоторых Yi.

Множества Y1, Y2, ..., Yk называют базисом зависимостей для X относительно D.

Если имеем функциональные зависимости X→Z , то Z∈{A1, A2,..., An} и базис Yi∈{A1, A2,..., An}.

Для функциональных зависимостей множество Yi состоит из одного атрибута, а для многозначных зависимостей множество Yi состоит из нескольких атрибутов.

* + 1. *Замыкание множества функциональных и многозначных зависимостей*

Нахождение множеств D+ , логически выводимого из D, производится при помощи аксиом 1 - 8. Аксиомы применяются ко множеству D до тех пор, пока возможен вывод новых зависимостей.

Задача состоит в следующем: определить следует ли из D зависимость X→→Y либо X→Y, этот процесс трудоемкий.

Для этого необходимо построить базис зависимостей для X. Если Y - X есть объединение каких-либо Yi, то данная зависимость принадлежит D+.

Для вычисления базиса зависимостей X относительно D достаточно найти базис относительно множества многозначных зависимостей M, которое состоит:

1. из всех многозначных зависимостей D;
2. из приведенных функциональных зависимостей X→Z к виду X→→Y1, X→→Y2, ..., X→→Yn, где Yi ∈ { A1, A2,..., An }

Для вычисления базиса зависимостей существует алгоритм Бири.

* + 1. *Соединение без потерь*

Данный алгоритм является обобщенным алгоритмом, учитывающим существование как функциональных, так и многозначных зависимостей.

1. Строим таблицу из величин a, b как в алгоритме проверки для функциональных зависимостей.
2. Строим совокупность функциональных таблиц, считая некоторую таблицу Т уже построенной и либо

а) отождествляем два символа вследствие функциональной зависимости как в аналогичном алгоритме, или

б) выбираем многозначную зависимость X→→Y и две строки t1 и t2 таких, что t1[X] = t2[X], строим новую строку u такую, что

u [X] = t1[X] = t2 [X]

u [Y] = t1 [Y]

u [R - X - Y] = t2 [R - X - Y]

и добавляем ее в таблицу, если ее там еще нет.

1. так как мы не добавляем никаких новых a и b, то множество таких таблиц будет конечно. Если какя-либо из таблиц будет иметь строку, содержащую только а, то рассмотренная декомпозиция обладает свойством соединения без потерь.

***Теорема* :**

Пусть R - схема отношения, ρ - декомпозиция на два отношения R1 и R2 , а D - множество функциональных и многозначных зависимостей, то говорят, что декомпозиция ρ обладает свойстом соединения без потерь, если :

(R1  ∩ R2 ) →→ (R1 - R2)

(R1  ∩ R2 ) →→ (R2 - R1)

* + 1. *Встроенные многозначные зависимости (ЕМV зависимости)*

Многозначные зависимости, справедливые для проекции любого вероятностного отношения r для R на подмножестве X, но не имеющего места в самом r называют ***встроенными многозначными зависимостями*** (EMV).

X →→ Y ( Z ), т.е. X мультиопределяет Y в контексте Z.

**Пример:**

R(C, BC, S, Y), где C - курс. BC - вспомогательный курс, S - студент, Y - год.

|  |  |  |  |
| --- | --- | --- | --- |
| **C** | **BC** | **S** | **Y** |
| 1 | 301 | Петров | 1976 |
| 1 | 302 | Петров | 1978 |
| 1 | 301 | Сидоров | 1979 |
| 1 | 302 | Сидоров | 1976 |

Рис. 30.

C →→ B C (Y) или C →→ S (Y)

* 1. *Зависимости соединения*

Многозначные зависимости являются попыткой выделить декомпозицию, обладающую свойством соединения без потерь. В этом случае такая декомпозиция может обладать этим свойством для всех отношений декомпозиций, но не обладать им для некоторого подмножества отношений декомпозиции. Поэтому необходимо определить декомпозиции, которые обладают свойством соединения без потерь, то есть найти так называемые J - зависимости. Нахождение такой декомпозиции приводит к получению 5НФ (PJ - формы).

Многозначные зависимости являются частным случаем J - зависимости; J - зависимости являются частным случаем EJ - зависимости, встроенной зависимости соединения.

Отношение R имеет EJ - зависимость, если существует несколько проекций для этого отношения, обладающих J - зависимостями, но в целом отношение не удовлетворяет J - зависимости.

Для поиска зависимостей соединения :(J и EJ - зависимостей) нет полной автоматизации. Выводимость для этого класса не разрешима.

* 1. *Нормализация отношений* 
     1. *Назначение нормализации и этапы нормализации*

Нормализация необходима для решения проблемы рационального выбора вариантов схем отношений из возможного множества альтернативных вариантов.

*Схема отношений должна удовлетворять следующим требованиям:*

1. выбранные для отношения первичные ключи должны быть минимальными;
2. выбранный состав отношений БД должен быть минимален, то есть отличаться минимальной избыточностью атрибутов;
3. при выполнении операций модификации, удаления и включения не должно быть трудностей;
4. перестройка набора отношений, приведение новых типов данных должна быть минимальной;
5. разброс времени ответа на различные запросы к БД должен быть небольшим.

Методы нормализации базируются на понятиях функциональных зависимостей, многозначных зависимостей и зависимостей соединения.

*Выделяют 7 этапов нормализации, соответственно 7 НФ:*

-устранение повторяющихся групп

|  |
| --- |
| 1НФ |

-устранение неполных функциональных зависимостей

|  |
| --- |
| 2НФ |

-устранение транзитивных функциональных зависимостей

|  |
| --- |
| 3НФ |

-устранение многозначных зависимостей

|  |
| --- |
| 4НФ |

-устранение встроенных многозначных зависимостей

|  |
| --- |
| 5НФ |

-устранение J - зависимостей

|  |
| --- |
| 6НФ |

-устранение EJ – зависимостей

|  |
| --- |
| 7НФ  Рис. 31. Этапы нормализации. |

* + 1. *Универсальные отношения*

Предположим, что проектирование БД начинается с выявления атрибутов и подбора данных, что выражается в построении словесного описания или нескольких разнородных таблиц.

##### БД «Питание»

###### Разнородные таблицы

|  |  |
| --- | --- |
| ***№*** | ***Продукты*** |
| 1 | Картофель |
| 2 | Капуста |

|  |  |
| --- | --- |
| ***№*** | ***Блюда*** |
| 1 | ….. |
| 2 | ….. |

|  |  |
| --- | --- |
| ***Поставщик*** | ***Дата поставки*** |
| Тихоок. порт | 25.02.02 |
| …. | …. |

###### Универсальная таблица «Питание»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Блюдо*** | ***Вид*** | ***Рецепт*** | ***Порций*** | ***Дата р.*** | ***Прод.*** | ***Калор*** | ***Вес (г)*** | ***Постав*** | ***Город*** | ***Страна*** | ***Вес (кг)*** | ***Цена*** | ***Дата по-ки*** |
| Лобио | Закус. | Лом. | 158 | 1/2002 | Фасоль  лук  зелень | …. | … | … | … | … | … | … | … |
|  | | | | |  | | | | | | | |

Этот вариант таблицы не является отношением и не находится ни в какой нормальной форме, т.к. большинство ее строк не атомарны. Атомарным являются значения полей: блюда, вид, рецепт (хотя рецепт представляется как большое текстовое поле)… до 5 колонки. Остальные поля таблицы множественные.

Для придания таким данным формы отношения необходимо реконстр. таблицу. Наиболее простой вариант – процесс вставки (заполнения пустых колонок исх. таблицы).

В таком виде исх. таблица «Питание» предст. экз. корректного отношения, которое и называется универсальным отношением проектируемой БД.

В одно универсальное отношение включаются все представляющие интерес атрибуты, а также все данные, которые планируется разместить в БД.

Для малых БД (до 15 атрибутов) универсальное отношение может использоваться в качестве исходной точки при проектировании БД.

Достоинства схемы объектов:

Уже есть все возможные отношения

Недостатки схемы объектов:

Необходимо переформирование

Достоинства универсального отношения:

Обобщенная структура позволяет сразу начать процесс декомпозиции

Недостатки универсального отношения:

При большем числе атрибутов возможно разночтение, а так же сложность с выделением составного ключа

**Доказать необходимость наличия составного ключа в УО. УО содержит все объекты области, следовательно, УО содержит несколько первичных ключей – составной ключ.**

Задачи выделения ключевых атрибутов в универсальном отношении.

Для анализа данных отношений используют метод попарных сравнений для выделения зависимостей.

**Почему проект БД может быть плохим?**

* 1. При использовании универсального отношения возникает ряд проблем:
     + - избыточность – данные практически всех столбцов многократно дублируются, повторяются также наборы данных (блюдо – вид – рецепт, продукт – калорийность – поставщик – город – страна). Нежелательно повторение значений, содержащих большие текстовые поля (рецепт), повторяются все данные о блюде вкл. в меню.
  2. Потенциальная противоречивость. Вследствие избыточности можно обновить адрес поставщика в одной строке не изменяя его в других, то есть при обновлении необход. просмотреть всю таблицу.
  3. Аномалия включения. В БД (∪) нельзя добавить, например, поставщика, продукты, которые не использ. ни в каком блюде.
  4. Аномалия удаления. Обратно 3.
     1. *Нормализация*

Нормализация – это разбиение отношения на два или более обладающими лучшими свойствами при включении модификации или удаление данных.

Окончательная цель нормализации – это получение такого проекта БД в которой каждый факт появляется лишь в одном месте, то есть max исключена избыточность информации (избыточность больше всего влияет не на объем памяти, а на удалении противоречивости данных).

Нормализованная таблица автоматически считается приведенной в 1 НФ. Таким образом, понятие «нормализованная» и находящаяся в 1НФ эквивалентны.

На практике термин нормализованная используется в смысле полной нормализации, то есть приведение к другим более высоким формам.

Каждая нормальная форма является более ограниченной, но и более желательной, чем предшествующая. Это связано с тем, что (N + 1) НФ не обладает положительными свойствами, характерными для N-ой формы. Общий смысл дополнительного условия, налагаемого на (N + 1) НФ по отношению к N состоит в исключении некоторых непривлекательных особенностей.

Теория нормализации основывается на:

* + - * наличии зависимостей между полями, которые подразделяются на многозначные и функциональные. Например: в таблице Блюда поля *Блюдо* и *Вид* функционально зависят от ключа *№*.

|  |  |  |
| --- | --- | --- |
| № | блюдо | вид |
|  |  |  |

поставщик

|  |  |  |
| --- | --- | --- |
| Поставщик | город | Страна |
|  |  |  |

В таблице *Поставщик* поле *Страна* функционально зависит от сост. ключа *Страна* и *Город*, но эта зависимость не является функционально полной, так как страна функционально зависит лишь от части ключа – *Город*.

Полные функциональные зависимости присутствуют только тогда, когда нет зависимости от части ключа.

* + 1. *Нормальные формы (НФ)*

Отношение находится в **1НФ** тогда и только тогда, когда ни одна из его строк не содержит в любом своем поле более одного значения и не одно из ключевых полей не пусто.

Таблица находится во **2НФ**, если она удовлетворяет определению первой и все ее поля не входящие в первичный ключ связаны полной функциональной зависимостью с первичным ключом.

**2НФ**

**5НФ**

**…**

Ко **2НФ** приведение заключается в необходимости произвести разбиение на несколько таблиц. В примере с поставщиками необходимо разбить на две: *Поставщики* и *Города*.

Поставщики

|  |  |  |
| --- | --- | --- |
| № | поставщик | город |
|  |  |  |

Города

|  |  |
| --- | --- |
| город | Страна |
|  |  |

Зависимость от части первичного ключа приводит к:

1. включение данных (пока не появится поставщик из Вильнюса нельзя зафиксировать, что это город Литвы)
2. удаление данных (исключение конкретного поставщика приводит к потере информации о местонахождении города)
3. обновление данных (при изменении названия страны приходится просматривать множество строк, поскольку много различных поставщиков живут в этой стране)

Для упрощения нормализации подобных таблиц целесообразно использовать рекомендацию: для проведения нормализации таблиц, в которых введены цифровые заменители составных или текстовых первичных ключей (внешних ключей) следует на время нормализации изменять их на исходные, а после окончания нормализации восстанавливать.

Таблица находится в **3НФ**, если она удовлетворяет **2НФ** и не одно из ее не ключевых полей не зависит функционально от другого не ключевого поля.

Для разбиения таблицы «Поставщики» все части этого проекта удовлетворяют определению **2НФ**, а так как в них нет не ключевых полей, зависящих друг от друга, то все они находятся в  **3НФ.** Это не относится к таблице «Блюда», так как после ввода первичного ключа № блюда появились не ключевые поля с функциональными связями: Блюдо → Вид, Продукт → Калорийность.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Блюдо | Вид | Продукт | Калорийность |
|  |  |  |  |  |

В этом случае приведение к **3НФ** будет заключаться в разбиении на 4 таблицы.

Блюда (№, Блюдо);

Вид (№, Вид);

Продукт (№ прод., Продукт);

Калорийность (№ прод., калорийность).

Эта проблема, которая может возникнуть не только из-за введения кодированных первичных ключей, но и по другим причинам, решается посредством более строгого определения **3НФ**.

Таблица находится в **НФ Бойса-Кодда (НФБК)** тогда и только тогда, когда любая функциональная зависимость между полями сводится к полной функциональной зависимости от возможного ключа.

В соответствии с этим таблицы Блюда и Продукты, имеющие по паре возможных ключей (№ блюда, блюдо, № продукта, продукт) находятся в **НФБК** или в **3НФ**.

1НФ

2НФ

3НФ

НФБК

4НФ

**3НФ** имеет определение с помощью отсутствия транзитивных зависимостей.

|  |  |  |  |
| --- | --- | --- | --- |
| № зач. | ФИО | № темы реферата | Тема |
|  |  |  |  |

X → Y, Y → Z ⇒ X→ Z

Решение этой проблемы заключается в разбиении этой таблицы на две - № зач., Фамилия; № темы, тема.

В следующих **НФ** (4 и 5) учитываются не только функциональные многозначные связи между полями таблицы. Таблицы нвходятся в 5НФ тогда и только тогда, когда в каждой её полной декомпозиции все проекции содержат возможный ключ. Полная декомпозиция таблицы – это совокупность любого числа ее проекций, соединение которых полностью совпадает с содержимым исходной таблицы.

Таблица не имеющая ни одной полной декомпозиции также находится в **5НФ**.

**4НФ** является частным случаем 5НФ. Когда полная декомпозиция должна быть соединением двух проекций. Очень важно подобрать реальную таблицу, которая находится в 4НФ, но не была бы в 5НФ.

**3НФ**: 2НФ + неключевой атрибут не транзитивно зависит от первичного ключа

**2НФ**: 1НФ + неключевой атрибут функционально полно зависит от первичного ключа

**1НФ**: каждый атрибут принимает атомарное значение

Тривиальной называется такая многозначная зависимость, для которой Х ⊆ У и объединение Х и У является рассматриваемым отношением. Тривиальная многозначная зависимость не нарушает 4НФ.

Отношение находится в 4НФ если оно находится в 3НФ и в нем отсутствуют нетривиальные многозначные зависимости.

|  |  |  |
| --- | --- | --- |
| ID | Autor | Еditor |
| 200 200 | Бочков Домус | Садчиков Садчиков |
| 100 | Джихани |  |
| 300 | Крон | Баранов |
| 876 876 | Гик Гик | Кикоин Капица |
| 440 |  | Витенберг |

Атрибуты автор и редактор образуют 2 многозначные зависимости от первичного ключа, при этом значения атрибутов не зависят друг от друга

|  |  |
| --- | --- |
| ID | Autor |
| 200 200 100 300 876 | Бочков Домус Джихани Крон Гик |

|  |  |
| --- | --- |
| ID | Editor |
| 200 300 876 876 440 | Садчиков Баранов Кикоин Капица Витенберг |

Нормализация сокращает дублирование данных, устраняет неопределенные поля, но усложняет проектирование с поддержкой логической целостности.

**3НФ**:

R → 3НФ, если не существует:

* Ключа Х для R;
* Множества атрибутов Y ⊆ R;
* Не первичного атрибута A из R

A ∉ X, Y, чтобы:

1. X → Y справедливо в R
2. Y → A справедливо в R, но
3. Y → X не имеет смысла в R.

**НФБК**

R с зависимостью F находится в НФБК, если всякий раз, когда в R имеет место X → A, A∉X, X включает ключ R.

То есть допускаются зависимости, в которых ключевые функции определяют один или более других атрибутов.

Анализ процесса реализации. На практике считается, что приведение к НФБК гарантирует нахождение таблицы в 5НФ.

Алгоритмически процедура нормализации основывается на том, что единственным типом зависимостей должны быть зависимости вида К->НП. Это следует из определения первичного ключа и F зависимости, при этом зависимость может быть распространена на зависимости вида F, J, MV, EJ,EMV. Цель нормализации состоит в том, чтобы избавится от зависимостей отличных от данной.

* + 1. *Процедура нормализации*

Нормализация – это последовательный процесс замены таблиц ее полными декомпозициями до тех пор, пока они не будут находиться в **5НФ**.

На практике считают, что приведение к **НФБК** гарантирует нахождение таблицы в **5НФ**.

Процедура нормализации основывается на том, что единственными функциональными зависимостями в любой таблице должны быть зависимости вида K → F (1), где К – первичный ключ, F – некоторое поле. Это следует из определения первичного ключа в соответствии с которым функциональная зависимость (1) всегда имеет место для любого поля данной таблицы. Это следует из определения первичного ключа и F зависимости, при этом зависимость может быть распространена на зависимости вида F, J, MV, EJ,EMV.

Цель нормализации состоит в том, чтобы избавиться от функциональных зависимостей отличных от (1).

Рассмотрим два случая:

1. Таблица имеет составной первичный ключ вида (К1, К2) и поле F, которое зависит от К2. в этом случае рекомендуется сформировать таблицу, содержащую К2 и F (К2 – первичный ключ). Заменить Т(К1, К2, F), первичный ключ (К1, К2), функциональную зависимость К2 → F на Т1 (К1, К2) первичный ключ (К1, К2) и Т2 (К2, F) первичный ключ (К2). 2НФ
2. Таблица имеет первичный (возможный) ключ К, не являющееся возможным ключом поле F1 (F1 функционально зависит от К), другое не ключевое поле F2, которое функционально зависит от F1. в этом случае формируется таблица, содержащая F1 и F2 с первичным ключом F1. Заменить Т(К, F1, F2), первичный ключ К, функциональную зависимость F1 → F2 на Т1(К, F1) – первичный ключ К и Т2(F1, F2) – первичный ключ F1.

Для любой заданной таблицы применение двух указанных правил почти во всех практических ситуациях дает возможность получить структуру (множество таблиц), которые находятся в «окончательной нормальной форме».

Для выполнения этих операций в качестве входных данных нужно иметь «большие таблицы» (или универсальные отношения).

* + 1. *Процедура проектирования*

Проектирование ИС начинается с построения инфологической модели данных, то есть идентификации сущностей. Затем выполняются следующие шаги процедуры проектирования даталогической модели:

1. Представить каждую независимую сущность базовой таблицей и специфицировать первичный ключ этой базовой таблицы.
2. Выбрать ассоциации (связь вида М:М) и представить каждую ассоциацию как базовую таблицу. Использовать внешние ключи для идентификации участков ассоциации и ввести ограничения, связанные с каждым из внешних ключей.
3. Представить каждую характеристику как базовую таблицу с внешним ключом. Специфицировать ограничения на внешний ключ и ее первичный ключ (комбинация этого внешнего ключа и свойства, гарантирующего уникальность).
4. Представить каждое обозначение, которое не рассмотрено в п.3 как базовую таблицу с внешним ключом, специфицировать ограничения.
5. Представить каждое свойство как поле в базовой таблице.
6. Если было произведено изменение таблиц, то следует модифицировать инфологическую схему и повторить шаги даталогии.
7. Указать ограничение целостности проектируемой БД и дать краткое описание полученных таблиц и их полей.

Для наиболее распространенных РБД используется инфологический язык моделирования, который называется *таблица – связь*. В нем сущности обозначаются одностолбцовыми таблицами с заголовками, состоящими из имени и типа сущности. Ключевые атрибуты обводятся рамкой. Связи между сущностями изображаются стрелками от первичных ключей или их составляющих.

Стержневые

Ассоциативные

Характеристические

Продукт

Калории

Цена, Вес (кг)

Поставщик (обознач.)

Поставки (ассоциатив.)

Поставщик Дата\_п

Поставщик

Город отправки

Город

Страна

Города (стержень)

Рецепт

Блюдо

Рецепты (характерист.)

Вес (г)

Блюдо

Продукт

Вид

Блюдо

Блюдо

Дата\_р

Кол-во

Расход (характерист.)

Блюда (стержень)

Состав (ассоциат.)

Продукты (стержень)

Приложением к любой модели является описательная часть, которая оформляется таблицей, где указывается кол-во полей, название атрибутов различных типов, тип сущности, возможные типы полей.

При нарушении ограничений требуется вывод сообщений. Значение поля Вид принадлежит набору.

1. ***Базовые средства манипулирования реляционными данными.***

Конкретный язык манипулирования БД называется реляционо полным, если любой запрос сформулированный с помощью одного выражения РА или одной формулой реляционного исчисления (РИ) этого языка. Механизмы РА и РИ эквивалентны. Ра и РИ различаются уровнем процедурности. РА имеет прямую процедурную ориентацию и строится на основе алгебраических операций. Запрос на языке РА вычисляется с помощью старшинства операций и скобок. Формула РИ не может быть однозначно трактована - она ставит условие, которому должны удовлетворять кортежи.

Существует 3 составляющих реляционной модели:

1. структурная
2. целостная
3. манипуляционная

Манипуляционная составляющая имеет два механизма манипулирования реляционными данными: реляционная алгебра (основная на теории множеств) и реляционное исчисление (базируется на математической логике).

Реляционное исчисление бывает двух видов: *исчисление доменов* и *исчисление предикатов*. Эти механизмы замкнуты относительно понятия отношения, то есть выражение реляционной алгебры и формулы реляционного исчисления определяется над отношением РБД и результатом является также отношение.

Любое выражение или формула могут быть представлены как отношения, что позволяет использовать их в других отношениях или формулах.

Алгебра и исчисление обладают большой выразительной мощностью: очень сложные запросы БД могут быть выражены с помощью одной формулы РИ или одного выражение РА.

* 1. *Реляционная алгебра*

Конкретный язык манипулирования РБД называется реляционно-полным, если любой запрос, выражаемый с помощью одного выражения РА или одной формулы РИ может быть представлен с помощью одного оператора этого языка.

Механизмы РА и РИ эквивалентны, то есть для любого допустимого выражения РА можно построить производящую такое же результат формулу РИ и наоборот.

Различаются оба этих механизма уровнем процедурности. Выражение РА строится на базе алгебраических операций (высокого уровня) и имеют процедурную интерпретацию, то есть запрос на языке реляционной алгебры может быть вычислен на основе алгебраич. операций с учетом старшинства и скобок.

Формула РИ не может быть однозначно трактована, она устанавливает условие, в котором должны удовлетворять кортежи обр. языки РИ декларированы и менее процедурны.

Редко алгебра или исчисление принимаются в качестве полной основы языка БД (SQL).

Расширенный начальный вариант алгебры представлен Коддом.

Набор алгебраических операций состоит из 8, которые делятся на 2 класса:

* + - * теоретико-множественные (1)
      * специальные (2)

В состав (1) входит:

* + - * объединение отношений
      * пересечение отношений
      * взятие разности отношений
      * декартовое произведение отношений

В состав (2) входит:

* + - * селекция отношений
      * проекция отношений
      * соединение отношений
      * деление отношений

В состав алгебры включается операция присваивания (позволяется сохранить в БД результаты вычисления), операция переименования атрибутов (дает возможность корректно сформировать заголовок или схему результата).

Общая интерпретация реляционных операций:

При объединении 2-х отношений производится отношение, включающее в себя кортежи входящие хотя бы в одно отношение из операндов.

При пересечении производится отношение, включающее все кортежи, входящие в оба отношения операнда.

Отношение разности включает все кортежи, входящие в отношение 1-ого операнда, такие что ни один из них не входит в отношение 2-ого операнда.

При прямом произведении производится отношение, кортежи которого являются полной конкатенацией кортежей 1-ого и 2-ого операнда.

При селекции отношения по некоторому условию является отношение включающее кортежи, удовлетворяющие данному условию.

При проекции производится отношение, домены которого формируются путем взятия соответствующих значений из отношения операнда.

Соединение отношений по некоторому условию образуют отношение, кортежи которого являются условной конкатенацией кортежей 1-ого и 2-ого операндов.

Операция реляционного деления имеет 2 операнда (бинарный и унарный). Результат состоит из условно унарных\* кортежей, включающих значения из первого операнда совпадающих со значением 2-ого сцепленных с кортежами 2-ого операнда.

Переименование производит отношение тело, которого совпадает с телом операндов, но атрибуты изменены.

Условно унарное означает, что вся группа доменов входящее во 2-ой операнд считается единым унарным или условным атрибутом. 1-ый операнд состоит из 2-ух унарных и представляет собой бинарный.

* 1. *Особенности теоретико-множественных операций.*

***Особенности теоретико-множественных операций РА (∩, , -):***

Смысл операции в целом теоретико-множественный, но в теории множеств операция осложнена для двух множеств операндов, а в случае РА результатом операции является отношение.

При допущении в РА возможности объединения отношений разными схемами, результатом операции может быть множество разнотипных кортежей (не отношения), а по определению замкнутости РА такая операция объединения невозможна. Следовательно для операций объединения, пересечения, разности должно быть соблюдено условие совместимости.

Совместимость отношений по объединению – отношения, имеющие одинаковые заголовки.

Если два отношения совместимы по объединению, то операции пересечения, объединения выполняются корректно.

*Взятие прямого произведения*

В теории множеств прямое произведение применимо для любых двух множеств; результаты – пара, так как отношения являются множителями, то для любых отношений эта операция возможна. Результаты – пара кортежей.

Поэтому в РА используется специализированная форма взятия прямого произведения, следовательно, понятие совместимости по взятию расширенного прямого произведения (если множества имен атрибутов отношений не пересекаются).

*Специальные реляционные операции*

Ограничение требует наличия двух операндов ограничиваемого отношения и простого условия ограничения.

Простое условие ограничения может иметь вид:

a

comp\_op

b

Имена атрибутов

Операции сравнения

a comp\_op const

Результат отношения: заголовок = заголовку отношения операнда, а в тело входят те кортежи, для которых значения условия является «истина».

Для обозначения операции ограничения используют конкатенацию.

A WHERE comp;

А – ограничиваемое отношение

Comp – условие сравнения

Comp1, comp2 – 2 простых условия ограничения, следовательно:

|  |
| --- |
| A WHERE comp1 AND comp2 |
| ↕ |
| (A WHERE comp1) INTERSECT |
| A WHERE comp1 OR comp2 |
| ↕ |
| (A WHERE comp1) UNION (A WHERE comp2) |
|  |
| A WHERE NOT comp1 |
| ↕ |
| A MINUS (A WHERE comp1) |

*Операция взятия проекции*

Требует наличия двух операндов проецируемого отношения А и списка имен атрибутов, входящих в заголовок отношения А.

Результатом проекции отношения А по списку атрибутов (а1, а2,.., аn) (1) является отношение с заголовком, определяемым множеством атрибутов. (1) является отношением с заголовком и телом, состоящее из кортежей вида <a1:v1; a2:v2, …, an:vn> таких, что в отношении А имеется кортеж, атрибут a1 которого имеет значение v1, а1…v2 и т.д., то есть выделяется «вертикальная» вырезка отношения операнда с уничтожением кортежей дубликатов.

*Операция соединения*

Требует двух операндов соединяемых отношений третьего операнда простого условия. Применяются условия ограничения.

Пусть соединяются отношения А и В. Как и в случае операции ограничения условия имеют вид: а comp\_op (b const), где а, b – имена атрибутов отношений; const – литерально-заданная константа.

Результатом операции сравнения является отношение получаемое путем выполнения операции ограничения по условия comp\_op прямого произведения А и В.

В общем случае применение условия соединения уменьшает мощность результата промежуточного произведения. Поэтому на практике считают реальными операциями соединения именно те, которые основаны на условии соединения приведенного вида.

Результатом является отношение с операцией ограничения на прямом произведении а и b. Частный случай соединения называется эквисоединением, а так же расширение соединения называется естественным соединением. Эквисоединение работает при условии равенства выбранных атрибутов.

Операция *естественного соединения* применяется к паре отношений, обладающие общим атрибутам, то есть атрибутом с одним и тем же именем и определенным на одном и том же домене.

Пусть а, b объединения заголовков отношений А и В, тогда естественное соединение А, В – это спроектированный на a, b результат эквисоединения А и В.

*Операция деления отношений*

Задаются 2 отношения с известными заголовками:

R1 с заголовками (a1, a2, …, an, b1, b2, …, bm)

R2 с заголовками (b1, b2, …, bm)

Будем считать, что атрибут bi отношения R1 и атрибут bi отношения R2 является общим. Назовем множество атрибутов {aj} составным атрибутом а, {bj} составным атрибутом b.

Будем говорить о реляционном делении бинарного отношения R1 (а, b) на унарное отношение R2 (b).

Первым этапом реляционного деления считается подготовительный этап анализа вхождения делителя в делимое.

Результатом такого деления является унарное отношение С, состоящее из кортежей V таких, что в отношении R1 имеются кортежи <V, W> такие, что {W} включает множество атрибутов в отношение R2.

*Замкнутость РА и операция переименования*

Она реализуется при конфликте имен атрибутов в отношениях.

|  |  |
| --- | --- |
| З1 | С1 |

|  |  |
| --- | --- |
| З1’ | С1’ |

………..

Заголовок

отношения

Множество таких пар

З1 – имя атрибута

С1 – имя домена

Домены атрибутов определяются доменами отношений-операндов. Сложность состоит в том, что отношения операндов прямого произведения имеют атрибуты и домены, при этом они одноименные.

Отношения

операндов

A1 : D1

A2 : D2

An : Dn

Ai = Aj

Di = Dj

Так как это множество, то одинаковых элементов быть не может, иначе невозможно выполнить операцию прямого произведения. Аналогичные проблемы возникают в других операциях. Решение – применение переименования. Эту операцию используют при конфликте имен атрибутов в отношениях.

*Дополнительные операторы Реляционной алгебры*

* Полу соединение SemiJoin – результат последовательного выполнения двух операций – соединение А и В и проекции результата на отношение А.
* Полу вычитание SemiMinus – результат выполнения двух операций – полу соединения и вычитания.
* Расширение Extend – на практике бывает необходимо запросить результат арифметического выражения или сослаться на результат такого выражения в условии выборки. Результатом операции расширения отношения А является отношение С (заголовок А + В1) которое получается из кортежей А и некоторого атрибута В указанного в операции расширения.
* Обобщение Summarize
* Транзитивное замыкание Tclose

***Реляционное исчисление***

В основе исчисления лежит понятия переменной и определенной для неё ОДЗ. Формулы в РИ опираются на переменные, предикаты и кванторы. В зависимости от ОДЗ существует исчисление кортежей и исчисление доменов. В РИ указываются характеристики результата без указания способов формирования. РИ основан на построении формул. Формулы РИ имеют вид:

Q=

Атомы:

1. R(t)
2. S[i] Ɵ U[j], где Ɵ - =,>,<,<=,>=,<>
3. S[i] Ɵ a, где a – const

Кванторы играют роль декларации и представляют информацию, относящуюся к объекту

***Правильно построенные формулы WFF***

Служит для выражения условий, накладываемых на кортежные переменные. Для определения кортежной переменной необходимо указать её область определения, то есть определенное отношения, на котором задана переменная.

RANGEVAR <имя переменной>

RANGES <список реляционных выражений>

Для определения WFF используют следующие утверждения:

* Каждый атом есть формула. Все вхождения переменной кортежа в атом являются свободными.
* Если существуют 2 формулы, то формулами так же являются их пересечение, объединение и отрицание. Если в эти формулы входят переменные с кванторами, то экземпляры кортежей считаются свободными или связанными в соответствии с этими кванторами.
* Если λ – формула то (ƎS)λ тоже формула. Существует такая некоторая S, при подстановке которой в места всех ее вхождений в формулу, λ становится истиной.

Если принять во внимание что форма If then – импликация, то импликация принимает значение ложь только в 1 случае, когда X – истина, а Y - ложь

РИ с переменными на доменах.

Формулы строятся на тех же операторах. Различия в том, что:

1. Вместо переменных кортежа в качестве переменной выступает домен.
2. Атомы имеют следующий вид: R(х1, х2,…, хк), где хi – const или переменная на домене. Значения тех хi, которые являются переменными должны быть выбраны таким образом, чтобы эта совокупность х была кортежем отношения.
3. Формулы в РИ с переменными на доменах используют связки , кванторы, сущ. понятия свободных и связанных переменных.

Существуют 3 теоремы попарно связывающих все виды представлений способов манипуляций

**Теорема 1:** если G есть формула РА, то существует эквивалентное ей выражение с переменными кортежами.

**Теорема 2:** для каждого выражения реляционного счисления с переменными кортежами существует эквивалентное ему выражение РИ с переменными на доменах.

**Теорема 3:** для всякого выражения РИ с переменными на доменах существует выражение РА.

Дополнительные возможности ЯМД:

* Команды включения и замены;
* Арифметические операции;
* Агрегатные функции;
* Построение многошаговых запросов (реализация запросов с промежуточными результатами и их сохранение);
* Возможные команды присвоения и печати (допускается печать отношения которая сконструирована с помощью РА или РИ).

Полным называется язык, в котором можно моделировать отношения с переменными на кортежах или доменах с помощью РА.

Сравнение языков исчисления и алгебраических языков.

Непроцедурные языки РИ являются языками более высокого уровня, чем РА.

Преимущества относительны, так как в РА есть блок оптимизации запросов.

Временные затраты наиболее заметны на операциях соединения и декартового произведения.

**Общая стратегия оптимизации**:

1. Выполнять операцию селекции по возможности раньше. Это приводит к сокращению промежуточных результатов и как следствие – повышение быстродействия.
2. Целесообразно обрабатывать файлы перед выполнением операции декартова произведения или соединения. Предварительная обработка заключается в сортировке файла и определении индексов с целью ассоциации одинаковых элементом в каждом из файлов.
3. Поиск общих подвыражений выражения. Если результат выполнения какого-либо выражения является небольшим, то можно запомнить, а в дальнейшем используется для вычисления других выражений (не всегда применяется).
4. Сбор в каскады, селекция и проекция. Так как данные операции для 1-го операнда могут выполняться за один просмотр.
5. Комбинация проекции с предшествующими и последующими двухместными операциями.
6. Комбинация селекции с предшествующим декартовым произведениями и выполнение этой операции вместо соединения.

Компоненты SQL СУБД определяют, как осуществлять навигацию по физическим структурам данных называются оптимизатор запросов. Навигационная логика, то есть вариант алгоритма называется путем или методом. Последовательность выполняемых действий для обеспечения выбранных путей называется планом. Процесс использования оптимизатором называется оптимизацией. Во время процесса оптимизации рассматриваются все команды SQL DML.

Синтаксическая оптимизация:

Оптимизация запросов определяется как сумма приемов для повышения эффективности запросов. Синтаксическая оптимизация представляет собой переформулирование запроса. Определить стоимость процесса обработки в терминах операции ввода-вывода. Для улучшения показателей используют эквивалентные алгебраические преобразования.

Оптимизация, основанная на правилах:

Использование техники индексов. Если таблица имеет индексы по 3 колонкам, то любой из индексов может быть для доступа. Используются различные виды соединений: 1)соединение с помощью вложенного цикла, когда вводится понятие внешней и внутренней таблицы; 2)соединение посредствам объединения. Таблицы отсортированы так, что строки читаются в порядке значений колонки, по которой они соединяются. Этот подход позволят производить соединение пока оно необходимо. Соединение осуществляется за один проход по каждой таблице. Соединение обладает коммутативным законом.

Кроме того все ситуации называются комбинированной сложностью выполняющих соединение.

Характеристики для анализа и возможности применения эвристики

* Число строк в таблицах
* Интервал и распределение значений колонок
* Длины строк и количество строк на физической странице диска
* Параметры индекса
* Количество терминальных страниц в индекса

Эти характеристики данных значительно влияют на эффективность обработки запросов

Оптимизация на основе стоимости

Сама оптимизация аналогична предыдущей – вычисляет стоимость плана, используя статистику. Формулы стоимости учитывают количество операций ввода-вывода и процессорное время. Использование индекса будет замедлять обработку по сравнению с обычным просмотром. Оптимизатор основанный на стоимости не обнаружил бы что индекс не дает преимуществ при обычном просмотре.

1. ***Алгебраическое манипулирование***
   1. *Эквивалентность выражения и законы преобразования выражений*

Два выражения в РА являются **эквивалентными** (или ), если представляют одно и тоже отображение, то есть при подстановке конкретных значений в выражение E1, E2 получается один и тот же результат.

Законы эквивалентных преобразований:

* 1. закон коммутативности для соединения и произведения



F – условное соединение.

* 1. закон ассоциативности







* 1. каскад проекций



* 1. каскад селекций



* 1. перестановка селекции и проекции



* 1. перестановка селекции и декартова произведения



**Следствия:**

1. если F представляется как формула , то можно записать 
2. 
   1. Перестановка селекции и объединения



**Примечание**: объединение E1 и Е2 должно иметь одинаковые атрибуты Ai = Aj или указано соответствие атрибутов

* 1. Перестановка селекции и разности



* 1. Перестановка проекции и декартова произведения



* 1. Перестановка проекции и объединения



В 10 примечания такие же как и в 7, 8.

* 1. *Алгоритм оптимизации регулярных выражений*

Основное применение: дерево, представленное выражение РА.

Получение результата: программы для вычисления выражений.

**Метод:**

1. Представляем любую селекцию вида  в каскад селекций 

2.Перемещаем селекцию насколько это возможно вниз по дереву соответственно (4) – (8)

1. Перемещаем проекцию вниз по дереву (3), (5), (9), (10). Правило (3) приводит к исчезновению некоторых проекций. Обобщение правила (5) разбивает проекцию на две, одну из которых можно перевести вниз. Проекция исключается, если она проецирует значения на все атрибуты.
2. Комбинируем каскад в одиночную проекцию (или селекцию) с последующей проекцией (3) – (5).
3. Разбиваем внутренние узлы полученного дерева на группы. Узел, представляет двухместный оператор произведения, пересечения, разности принадлежит группе вместе с непосредственными предками.
4. Создаем программу для вычисления группы так, чтобы ни одна группа на вычислялась раньше группы-потомка.

**Пример** организации запроса.

**Даны отношения:**

КНИГИ (№, название, автор, издательство)

ИЗДАТЕЛЬ (название\_изд-ва, место, адрес)

ЧИТАТЕЛЬ (№карты, ФИО, адрес, город)

ВЫДАЧА (№карты, №книги, дата)

**Запрос:** определить книги, которые были выданы до 1 февраля 2002 читателю Иванову.





Сравнение нужно делать через декартово произведение.

**Строим каскад:**











книга



книга

читатели

Спускаемся вниз по дереву селекции:

Объединение проекций П5 и Читатель:









книги







читатель

выдача

Строим каскад проекции и селекции:







книги







читатель

выдача

1)







Пкнигиюназв, книги.наим., выдача.наимен.











Пвыдача.наим\_книги

выдача

читатель

Пкнигиюназ, номеркниги

2)

Пвыдача.наим\_книги



Пчитатюномер\_карты, выдача.№карты, выдача.№книги







Пкниги.назв.,нимен.№книги

книги

Пвыд.№книги





Пчитатель.№карты

Пвыдача.№карты, выдача.№книги



выдача



читатели

Эта проекция перечисляет все атрибуты по отношению к выдаче, поэтому каскад (3) мы удаляем.

3)

Пвыдачи.№карты, выдача.наимен\_книги



Пвыдача.дата, выдача.книга, выдача.карта

I часть

Пназв.



Пвыдача.№книги

Пчитатель.назвение, книги.№выдачи





Пвыдача.№карты, выдача.№книги

Пчитатель.№карты



Пчитатель.№карты, читатель.ФИО

читатели



выдача

II часть

Сначала выполняется II, а потом I

4)

Пчмтатель.№карты



Пчмтатель.ФИО, читатель.№карты

Для оптимизации необходимо реализовать порядок проекции и выборки.

Преобразование логики предикатов

Когда есть несколько предикатов в операции выборки важна эвристика состоит в том, что бы преобразовать их в эквивалентные выражения способом перегруппировки.

Конъюнктивная форма более предпочтительна, так как может быть оценена с помощью метода укороченной цепочки. Набор предикатов в конъюнктивной форм будет истина тогда и только тогда, когда каждая компонента будет истина. Так как число операций сравнения, которые выполняются в запросе, прямо влияют на время, то использование метода укороченной цепочки напрямую сокращает время выполнения запроса.

Алгоритм оптимизации выражений РА.

Входными данными является дерево запроса. Выход – программа для вычисления выражения.

1 этап – представление селекции в форме каскадов.

2 этап - перемещаем селекцию вниз по дереву, насколько это возможно в соответствии законов 4-8.

3 этап – перемещаем проекцию вниз по дереву согласно правилам 3,5,9,10. Правило 3 приводит к исчезновению некоторых проекций. Правило 5 может разбить проекцию на 2, одну и которых переносим вниз по дереву.

4 этап – комбинируем каскад в одиночную селекцию или проекцию с последующей проекцией.

5 этап – разбиваем внутренние узлы дерева на группы. Узел представляет собой 2-местный оператор пересечения или разности и принадлежит группе вместе с предками.

6 этап – ставим ограничения на вычисление групп предков позже, чем групп потомков.

* 1. *Декомпозиция запросов*



S - множество атрибутов, F – некоторая формула.

Необходимо определить порядок декартова произведения.

Имеем три отношения:

А) R1 (A, B)

Б) R2 (B, C)

В) R3 (C, D)

Необходимо выполнить естественные соединения:



Чтобы представить запросы, включающие декартовы произведения, и затем его оптимизировать, используем граф связи.

* + 1. *Граф связи*

Допустим, имеем запись , к ней применяем селекцию, причем . тогда вершина графа – это отношение, а ребра – условие Fj. Fj не должно включать операции логического умножения и должно соответствовать ребрам некоторого связывающего отношения, атрибуты которого используются в Fj.

**Пример**: 



Все отношения тесно связаны между собой:

F1

F2

F3

F4

* + 1. *Декомпозиция графа связи*

Исполнение запроса можно рассмотреть как последовательность операций над графом. Любой операции будут соответствовать новое отношение и некоторое изменение графа. В результате выполнения всех операций граф должен исчезнуть, и будет получено новое отношение. Const изображаются в виде , если имеем ребро, которому соответствует условие А = В, то такое условие (ребро) называется **простым**.

При моделировании графа (изменение модификации) используется 2 вида построения:

* 1. инстансация
  2. разбиение

1. Если есть 2 вершины n и m, соединенные простым ребром A = V, если вершине m соответствует единичная const V, n соответствует отношение r, то эти два узла заменяем одним, которому будет соответствовать отношение.



V

r

m

n

A = V ⇒

r’

1. Если есть некоторый узел N, имеющий m инцендентных ребер, в которых сопоставлено условие fj, узлу сопоставлено отношение r, в результате разбиение вместо одного узла получаем m узлов, которые будут соотв. узлы const, а в условиях происходят изменения атрибутов.

n

fm

fj

1

j

m

⇒

ti [A1]

ti [Am]

**…**

**1**

**m**

**Алгоритм выполнения запроса путем декомпозиции.**

1. Входные данные: граф связи G0.
2. Выходные данные: отношение, заданное на графе G0 R(G0)

**Метод:**

* 1. индекс i:=1
  2. Выполняем операцию инстансакции пока это возможно.
  3. Производится выбор узла для разбиения. Принимаются следующие приоритеты:
  4. узел-const с простыми ребрами
  5. узел с простыми ребрами
  6. узел, при использовании которого, граф становится несвязным
  7. произвольный узел

3. Разбиение выбранного узла и получение графа Gi.

Отношения: R (G0)



Если существуют связанные узлы, то пункт 1, если граф несвязан, то пункт

4. Вычисляем



Вj – несвязанные узлы графа.

**(\*)**

G0 = книги.№книги=выд.№книги

кн.назв.изд = изд.назв

Чит.№карты = выд.№карты

изд.место = чит.город

Пример:

**1**

* 1. Const здесь нет ⇒ операции выполняться не будут.
  2. Так как все ребра простые, выбираем любой узел, например, книги. Операция разбиения: есть два инцидентных узла, узел запоминаем, получаем G0.

Заменяем **выдачи.№книги (\*)** на t1 [Nкниги]

Выд.№книги = t1 [Nкниги]

Изд.назв = t1 [наимен. изд.]

Строим G’0 = R (G0) = 

кн.№кн = t1 [№кн]

кн.назв\_изд = t1 [назв\_изд]

2. Заменяем два узла t1 [№книги] и выдачи на следующие соотношения



Выбор узла для разбиения: берем В1 и заменяем const.

Чит.№карты = t2[№кар]

Изд.место = чит.город

G’1

Изд.место = чит.город

Чит.№карты=выд.№карты

1. Выполняем для единой const следующие действия



изд.место=чит.город

Выполняем разбиение по В3. Заменяем const.

изд.место= t3[гор]



4. 



G’3

* 1. *Точная оптимизация для подмножества реляционных запросов.*

Алгебраические способы декомпозиции, рассмотренные ранее для преобразования запросов дают запросы, которые могут быть обработаны более эффективно. Указанные способы не определяют показатель эффективности обработки запросов.

Проблема нахождения оценки критериев оценки для большинства запросов неразрешима.

Существует класс запросов: **конъюнктивные запросы** для которых есть методы, обеспечивающие точную оптимизацию.

Критерием является количество соединений и произведений.

* + 1. *Конъюнктивные запросы*

Вычисления с переменными на доменах имеют вид:



а – переменные или const

b – переменные

c – const

 имеет вид: 



Пример: найти читателей, которые берут книги, опубликованные в городе, в котором они живут.

m=9

b1,…,b9 – атрибуты во всех отношениях

a1 – ФИО.

Пример 2: найти названия всех книг, выданных по карте №13 и дату их выдачи.



a1, a2 – название книги и дата выдачи.

Конъюнктивные запросы эквивалентны тогда, когда существует двойное отношение **«содержится».**



Отношение «содержится», когда имеется свертка.



Свертка имеет место в том случае, если есть отображение f-символов запросов Q1 в символы запроса Q2. Для f существует зависимости:

1. 

2. 

3. 

4. 

 можно представить в виде совокупности d1, d2,…, dk тогда имеет место формула:



**Теорема 9.** Теорема о свертках.



* + 1. *Минимизация конъюнктивных запросов*

**Теорема 10.** Для любого КЗ Q существует КЗ Q0 с наименьшим возможным числом входящих элементов. Любой другой запрос Q0 эквивалентен Q. Имеет min число составляющих (термов) и содержит какие-либо квантируемые переменные наз. **взаимоодназным переименованием символов Q0.**

*Следствие:*

Для любого заданного КЗ Q можно найти эквивалент с min числом термов путем удаления из него термов и всех кванторов существования таких, что bi не используется ни в одном из термов.

Пример: дан запрос:



Минимизация запроса:

2, 3, 4 удалять нельзя, так как в них присутствуют константы и переменные а1, а2. Возможно удалить 1, так как b1 и b3 нигде больше не используются, то есть не встречаются в других термах, следовательно нужно удалять их кванторы.



*Сетевая модель данных. Язык CODASYL.*

Язык Codasyl (первая версия 1969 г.) был предназначен для описания сетевых структур данных. Находился в разработке до 80-х годов.

На его базе были созданы ЯОД и ЯМД.

Рассчитан на пользователей, программистов и рассматривается как набор средств и методов для хранения данных. Поэтому для него не существует строгой математической модели описания данных. Большинство больших БД разрабатывалось на этом языке.

Основная идея языка: использование наборов. Он отражается связи 1:М. В наборе выделяется **владелец** и **член набора**. Реализация наборов осуществляется с использование связанных списков, рассматриваются функции связи членов с владельцами и наоборот.

*Записи CODASYL*

Тип представляет собой множество записей, обладающих структурой и другими свойствами, которые специфицированных в данном типе записей.

Со структурной точки зрения представляет собой иерархию, образованную из простейших элементов данных, простых групп таких элементов и повторяющихся групп.

Для повторяющихся групп допускается переменное число повторений и вложенность. В записи могут содержаться произвольные элементы данных, а также производные, которые зависят от значений других переменных (элементы той же записи, элементы записей владельца наборы и т.д.).

А

В

С

D

А

С

*Денормализация*

* Полная нормализация обозначает наличие большого количества базовых переменных.
* Большое количество отношений означает большое количество файлов.
* Большое количество файлов означает большое количество операций ввода-вывода

Денормализация в большинстве случаев выполняется на уровне хранимых файлов а не отношений.

Если нормализация переменной отношения R означает замену ее множеством проекций Ri для всех возможных значений этой переменной, что результатом обратного соединения значений переменной R будет исходной значение. Конечной целью денормализации является сокращение избыточности за счет приведения Ri к максимальному уровню нормализации (5НФ)

Денормализация отношений Ri это замена соединением R таких, что для всех значений r операция проекции отношения R по атрибутам снова приводит к созданию значений r. Конечная цель денормализации – сокращение количества соединений Ri которые требуются при работе приложения за счет предварительного создания этих соединений на одном из этапов проектирования.

Проблема денормализации:

* Проблема окончания. В случае нормализации существуют логические причины ее продолжения пока не будет достигнута самая высшая НФ. При денормализации обычно не решают задачу достижения самой низкой НФ и окончания денормализации связано с решением практической задачи.
* Избыточность хранения данных и аномалии обновления.
* Извлечение данных. Усложнение запросов.
* Проблема эффективности для разных приложений.

Проблема нормализованных отношений заключается в реализации множественных соединений. Повышение эффективности достигается путем предварительного анализа и пошаговой реализации соединений.

Ортогональный принцип – не какие 2 переменные отношения в БД не должны иметь перекрывающихся смысловых отношений.

Безусловно некорректная структура требует объединения.

Ортогональное проектирование представляет возможность дублирование информации с целью ее кластеризации или принадлежность к определенному смыслу.

*Многомерные БД*

Узко специализированные СУБД для интерактивной аналитической обработки данных. Основные понятия СУБД:

* Агрегируемость – рассмотрение информации на различных уровнях обобщения.
* Историчность – обеспечение высокого уровня неизменности относительно данных и взаимосвязей и привязка их ко времени.
* Прогнозируемость – подразумевает наличие функций интерполяции на различных временных интервалах.

Мы имеем многомерное логическое представление

По сравнению с РМ многомерная обладает наглядностью и информативностью.

Основные понятия многомерных моделей это измерение и ячейка

Измерение – это множество однотипных данных, образующих грань гиперкуба.

Ячейка – поле с фиксированным набором значений.

Гиперкубическая схема.

---

Полекубическая схема.

---

Специальные операции:

Вращение – изменение осей самого куба;

Формирование среза – подмножество куба полученное путем фиксации одного или нескольких измерений;

Агрегация;

Детализация.

Громоздкость для простых задач, сложное поддержание целостности, обновление ссылок.

*Иерархические системы*

Типичный представитель IMS. Манипулирование данными – стандарт. Принципы для иерархических систем:

* 1. Найти указанное дерево.
  2. Перейти от одного дерева к другому.
  3. Перейти от одной записи к другой в порядке обхода иерархии и т.п.

Ограничение целостности: поддерживает целостность между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

*Сетевые системы*

Основной представитель IDMS. Структура основана на известных принципах построения структур и является расширением иерархической модели (потомки могут иметь несколько предков).

Манипулирование данными:

* 1. Найти конкретную запись в наборе однотипних записей.
  2. Перейти от предка к первому потомку по определенной связи и т.п.

Ограничение целостности: их поддержание не требуется, кроме ограничений по ссылкам.

1. *Сетевая модель данных. Язык CODASYL.*

Язык Codasyl (первая версия 1969 г.) был предназначен для описания сетевых структур данных. Находился в разработке до 80-х годов.

На его базе были созданы ЯОД и ЯМД.

Рассчитан на пользователей, программистов и рассматривается как набор средств и методов для хранения данных. Поэтому для него не существует строгой математической модели описания данных. Большинство больших БД разрабатывалось на этом языке.

Основная идея языка: использование наборов. Он отражается связи 1:М. В наборе выделяется **владелец** и **член набора**. Реализация наборов осуществляется с использование связанных списков, рассматриваются функции связи членов с владельцами и наоборот.

* 1. *Записи CODASYL*

Тип представляет собой множество записей, обладающих структурой и другими свойствами, которые специфицированных в данном типе записей.

Со структурной точки зрения представляет собой иерархию, образованную из простейших элементов данных, простых групп таких элементов и повторяющихся групп.

Для повторяющихся групп допускается переменное число повторений и вложенность. В записи могут содержаться произвольные элементы данных, а также производные, которые зависят от значений других переменных (элементы той же записи, элементы записей владельца наборы и т.д.).

А

В

С

D

А

С

* 1. *Наборы CODASYL*

Если существует отображение m:1 записи типа R2, R1, то запись m(S)=r говорит о том, что кортеж S отношения записи R2 является членом набора, владельцем которого является запись r типа R1. Множество всех записей членов набора S, владельцем которого является r и саму r наз. **экземпляром набора.**

В CODASYL существуют ограничения:

1. Тип записей. Запись не может быть одновременно владельцем и членом одного и того же набора. Чтобы обойти это ограничение вводятся фиктивные наборы.

Поставщик

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ФИО | Адрес | Дата рождения | | |
|  |  | ч | м | г |

Товар

Пост\_цена

|  |
| --- |
| Назв\_товара |

Цена

|  |  |  |
| --- | --- | --- |
| ФИО | Назв\_товара | цена  товар\_цена |

Record Поставщик

1 ФИО char (20)

1 Адрес char (20)

1 Дата\_рожд

2 число int (20)

2 м int (20)

2 г int (8)

record Товар

1 назв char (20)

record Цена

ФИО virtual source is Поставщик

ФИО of owner of Поставщик\_цена

1 назв\_товара virtual source is товар

назв\_товара of owner of товар\_цена

1 цена int(10)

Наборы SET

Set Пост\_цена

owner is Поставщик

member is Цена

set Товар\_цена

owner is Товар

member is цена

Для данного типа записи можно определить понятие **сингулярного запроса**, который обладает двумя свойствами:

* + - 1. типом записи владельца такого набора является специальным тип записи SYSTEM
      2. имеется в точности один экземпляр набора, а все записи данного типа являются членами этого набора. Эти записи автоматически становятся членами набора без специальных указателей.

|  |  |
| --- | --- |
| SYSTEM  Товар  Цена | Поставщик |
| Все поставщики | |

Set все поставщики owner is SYSTEM member поставщик

* 1. *Древовидные структуры*

Тип набора может содержать несколько типов записей. В этом случае набор представляет собой дерево.

клиент

счет

доход

Любое n-уровневое дерево может представлять n-1 набор. В этом дереве все, кроме 1-ой и последней записи иные записи могут быть членами и владельцами различных наборов.

Три уровня и два набора.

клиент

счет

доход

расход

налог

деньги

Банк\_счет

* 1. *Простые файлы*

Для любого типа записей можно указать регулярный набор. Если используется регулярный набор SYSTEM, то мы имеем **простой файл**.

* 1. *Сетевая структура*

SYSTEM

клиент

счет

доход

расход

налог

деньги

Банк\_счет

CODASYL не допускает описания сложной сетевой структуры.

Сложная сетевая структура:

Покуп

Клиент

Покупатель

Товар

Покупка тов.

А

В

* 1. *Петли*

В CODASYL понятие петли не описывается, то есть запись не может быть одновременно и владельцем и членом одного и того же набора, но легко описываются циклы как деревья.

Способы размещения записей:

1. CALC
2. VIA
3. KEY
4. DIRECT
5. Вычисление размещается с помощью хэш-функции (вычисление адреса на диске по ключу)
6. Через (посредством). Размещение вблизи владельца, при этом владельцы размещаются на значительном расстоянии друг от друга, а члены – рядом с владельцами используя хэш-функцию.
7. Размещение по порядку. Оставляется место для вставок.
8. непосредственное размещение по алгоритму пользователя. Обычно используется для размещения уникальных записей владельца сингулярного набора.
   1. *Операции манипулирования данными в CODASYL*
9. Операции поиска FIND делят на 2 группы:
   1. Простой поиск («поиск на ровном месте»)
      1. FIND ANY < имя записи> [USING <имя ключа>] – поиск по ключу
      2. FIND FIRST <имя записи> WITHIN <имя области> - для нахождения записи в первой области
      3. Если ключ был установлен, то

FIND CURRENT <имя> [WITHIN <имя набора><имя области>]

* + 1. FIND FIRST <имя записи> USING <имя ключа> - поиск первой записи использующей ключ
  1. Навигационный поиск
     1. FIND FIRST | LAST <имя записи> WITHIN <имя набора> - найти первую, последнюю запись в наборе, при этом запись владелец – текущая.
     2. FIND NEXT | PRIOR <имя запроса> WITHIN <имя набора> - н6айти следующую или предыдущую запись, при этом запись-владелец – текущая.
     3. FIND <имя записи> WITHIN <имя набора> CURRENT USING <ключ> - найти в наборе текущую запись, использующую поисковый ключ для записи-члена.
     4. FIND NEXT <имя записи> USING <ключ> - найти следующую запись
     5. *Другие команды манипулирования данными*

1. Открытие одной или нескольких областей памяти

READY <имя области> | all mode retrieved | update [protect | exclusive]

1. Окончание работы с БД, закрытие всех областей

FINISH ALL <имя области>

1. Копирование текущей записи в текущую область

GET <имя записи>

1. Заполнение индикатора текущей записи для поиска по ключу

ACCEPT <ключ> from [<имя записи> | <имя набора> | <имя области>] currency

1. Модификация записи

Modify <имя записи> [using <список наборов>]

1. Стереть запись

ERASE [ALL] <имя записи>

Если есть потомки у записи, то они удаляются.

* 1. *Достоинства и недостатки*

Достоинства:

* + - * эффективное обновление и доступ к большим БД
      * обеспечение разумной скорости при выполнении операций соединения на большом количестве записей

Недостатки:

* + - * отсутствие полного языка запросов высокого уровня
      * нет математического обеспечения аналогичного реляционной алгебре

1. *Иерархическая модель данных. Язык DL1.*

В отличие от языка CODASYL основной структурой языка DL1 является **дерево сегментов.**

DL1 был разработан лоя работы с большими БД. Этот язык используется для логического описания БД и для физического.

Физическая БД размещается на носителе в виде дерева. Доступ к данным осуществляется одним из 4-х способов.

1. Иерархично-последовательностный
2. индексно-последовательностный
3. прямой
4. индексно-прямой

Деревья, в которых используются два первых метода доступа располагаются последовательно, для последних методов используется хэширование (хэш-функции).

На первом уровне могут располагаться несколько типов сегментов, а БД может состоять из нескольких физических деревьев.

* 1. *Физические и логические БД*

Логическая БД в DL1 имеет следующую структуру:

**Запись** в логической БД определяется как иерархически связанная совокупность сегментов. В целом лог. БД – это совокупность лог. записей. Запись в DL1 соответствует набору в CODASYL.

Указатель на сегм. служащие

описание

профессия

Тариф\_ставки

курсы

служащие

служ

Плат\_ведомость

образование

Трудовая деят.

Кред. рганизация

Сведения о банке

звание

опыт

Изм. оклада

Указатель на сегм. курсы

курсы

служ

образование

звание

Курсы обучения

Логическая БД

Логическая запись

сегмент

поле

**Сегмент** – это поименованная единица данных, которая содержит одно или несколько полей.

**Поле** – это мин. поименованный элемент данных.

Существуют лог. БД, которая сконструирована из одной или нескольких физических БД.

Схема физической БД – это дерево типов логических записей.

Описание физической БД включает:

1. имена сегментов
2. имена их полей
3. информация об образуемой сегментами иерархической структуре
4. спецификация физической организации доступа (метод доступа)

В DL1 физической БД соответствует КМД в РБД, логическая БД соответствует понятиям внешней модели. Поэтому на основе физической структуру можно построить несколько логических БД целиком. Для этих целей используются указатели на соответствующие сегменты.

Пример: есть физическая БД, которая включает две древовидные структуры.

Каталог работ

* 1. *Представление сетевых структур в DL1*

Существует возможность представить простую и сложную сетевую структуру.

Сложная сетевая структура:

А

В

Введение указателя преобразует сложную сетевую структуры в древовидную

А

В

Указатель на В

Указатель на А

А

В

С

Указатель на С

* 1. *Описание БД в DL1*

1. Определение БД

DB NAME = <имя БД> {ACCESS=метод доступа> | LOGICAL}

1. Описание сегмента

SEGM MANE=<имя сегмента> PARENT = <имя исходного сегмента>

1. Операторы

DB DGEN

FINISH

END

Пример недвижимостей

районы

конторы

Агенты (имя, сд)

Опись (адр, цена)

Клиент (имя, адр)

DBD NAME=НЕДВИЖ

SEGM NAME=РАЙОНЫ

FIELD NAME=(НАЗВ\_РАЙОНА, SEQ, И)

SEGM NAME=КОНТОРЫ, PARENT=РАЙОНЫ

FIELD NAME=(ГОРОД, SEQ, И)

FIELD NAME= АДРЕС

….

* 1. *Манипулирование данными в DL1*

1. GET UNIQUE – выбрать уникальный
2. GET NEXT – выбрать следующий
3. GET NEXT PARENT – выбрать следующий в пределах исходного. Под исходным сегментом определен п.1 или п.2
4. GET HOLD UNIQUE – выбрать уникальный и сохранить
5. GET HOLD NEXT – выбрать следующий и сохранить
6. GET HOLD NEXT WITHIN PARENT – выбрать следующий в пределах исходного
7. DELETE – удалить
8. REPLACE – изменить
9. INSERT – вставить