**Лекция 1**

**Управление данными** – деятельность, направленная на определение, создание, хранение, поддержку данных, а также на обеспечение доступа к данным и процессам манипулирования ими в одной или более информационной системе.

**Данные** – предоставленная информация в формальном виде, пригодном для передачи, интерпретации или обработки людьми и компьютерами.

**Информационная система** – система, организующая обработку информации о предметной области и ее хранение.

**База данных (БД)** – совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы, отображающих состояние объектов и их взаимосвязей в предметной области.

**База данных (ГОСТ 33707)** – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

**Система управления базами данных (СУБД)** – совокупность программных средств для создания, ведения и совместного использования баз данных многими пользователями.

**Система управления базами данных (ГОСТ 33707)** – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

**Основные функции СУБД:**

* **управление данными во внешней памяти**: обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях. В развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, и если использует, то как организованы файлы. В частности, СУБД поддерживает собственную систему именования объектов БД.
* **управление буферами оперативной памяти**: СУБД обычно работают с БД значительного размера; по крайней мере этот размер обычно существенно больше доступного объема оперативной памяти. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти.

Существует отдельное направление СУБД, которое ориентировано на постоянное присутствие в оперативной памяти всей БД. Это направление основывается на предположении, что в будущем объем оперативной памяти компьютеров будет настолько велик, что позволит не беспокоиться о буферизации. Пока эти работы находятся в стадии исследований.

* **управление транзакциями: транзакция** - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.
* **ведение журнала изменений**: одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя.

**Журнал** - это особая БД или часть основной БД, недоступная пользователю и используемая для записи информации обо всех изменениях БД.

Ведение журнала изменений (журнализация изменений) выполняется СУБД для обеспечения надежности хранения данных при наличии аппаратных сбоев и отказов, а также ошибок в программном обеспечении.

**Язык базы данных (ГОСТ 33707)** – язык с использованием формального синтаксиса, предназначенный для определения, создания, организации доступа и поддержки базы данных.

SQL (Structured Query Language).

* **поддержка языков БД**: для работы с базами данных используются специальные языки, в целом называемые языками баз данных. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка - язык определения схемы БД (SDL - Scheme Definition Language) и язык манипулирования данными (DML - Data Manipulation Language). SDL служил главным образом для определения логической структуры БД, т.е. той структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

**Типовая организация СУБД:**

* ядро СУБД:
* управление данными во внешней памяти (менеджер данных);
* управление буферами оперативной памяти (менеджер буферов);
* управление транзакциями (менеджер транзакций);
* журнализация (менеджер журнала).
* компилятор языка БД:
* преобразует операторы языка БД в некоторую выполняемую программу.
* подсистема поддержки времени выполнения.
* набор утилит:
* в отдельные утилиты БД обычно выделяют такие процедуры, которые невыгодно выполнять с использованием языка БД: загрузка и выгрузка БД, сбор статистики, проверка целостности БД и т.д. программируются с использованием интерфейса ядра СУБД.

**Классификация баз данных**

На основе хранения и доступа к данным:

* настольные (индивидуального доступа, desktop);
* с единым хранением данных;
* распределенные;

СУБД: ACCESS, Paradox, dBase, FoxPro.

Отсутствие средств для обеспечения надежности функционирования и средств защиты данных, необходимых при создании многопользовательских БД.

**БД с единым хранением**

Обеспечивают единое хранение данных и работу с этими данными многих пользователей.

Два похода к построению: **файл-серверные** и **клиент-серверные системы.**

**Файл-серверные системы**



Плюсы:

* легкость реализация

Минусы:

* большой объем передаваемых по сети данных (сетевой трафик);
* низкая надежность (данные передаются на компьютеры пользователей и там могут быть искажены);
* для повышения производительности обработки следует повысить производительность компьютеров пользователей, а не одного сервера;
* пока с данными работает пользователь, другие вынуждены ждать.

**Клиент-серверные системы**



Сервер служит для хранения и для обработки данных.

Хранимые процедуры и триггеры



**Трехуровневые (трехзвенные) БД**



**Сервер данных** – для хранения данных.

**Сервер приложений** – для обработки данных.

**Распределенные БД**

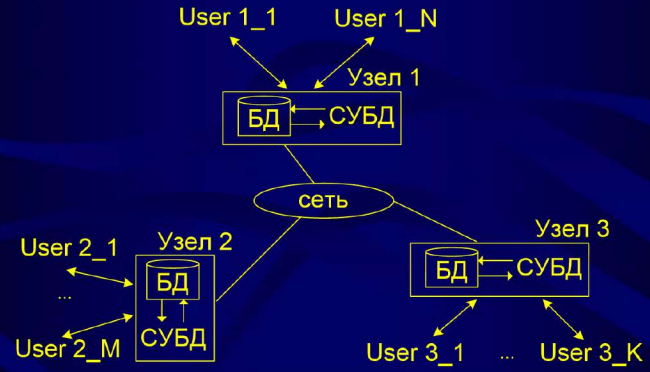
**Распределенная обработка данных (ГОСТ 33707)** – обработка данных, при которой выполнение операций распределено по узлам вычислительной сети.

**Распределенная БД (РБД)** – совокупность логически взаимосвязанных разделяемых данных, физически распределенных в компьютерной сети.

**Распределенная СУБД (РСУБД)** – программный комплекс, предназначенный для управления РБД и обеспечивающий прозрачный доступ пользователей к распределенной информации.

Прозрачный доступ заключается в том, что с точки зрения конечного пользователя РБД должна вести себя как нераспределенная.

**Распределенные БД**



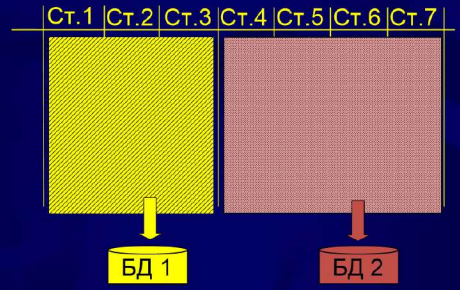
**Подходы к разработке распределенных БД**

Полная фрагментация

**Горизонтальные фрагменты** – подмножества строк таблицы.



**Вертикальные фрагменты** – подмножества столбцов таблицы.



**Подходы к разработке распределенных БД**

При полной репликации (тиражировании) на всех серверах хранятся одни и те же данные (полное дублирование)

Плюсы:

* повышение доступности и надежности данных;
* повышение рабочих характеристик системы.

Минусы:

* необходимость в дополнительном объеме памяти;
* поддержание согласованности данных в разных копиях.

Основные проблемы создания распределенной БД:

* фрагментация данных и распределение по компьютерам;
* составление глобального каталога, содержащего информацию о каждом фрагменте БД и его местоположении в сети;
* организация обработки запросов:
* синхронизация нескольких запросов к одним и тем же данным;
* исключение аномалий удаления и обновления одних и тех же данных;
* оптимизация последовательности шагов при обработке запроса и т.д.

**Лекция 2**

**Классификация баз данных по характеру организации данных**

Базы данных:

* неструктурированные – БД, информация в которых представлена в виде семантических сетей;
* частично структурированные – БД содержит информацию в виде текста или гипертекста;
* структурированные – БД с предварительно описанной структурой данных.

**Классификация баз данных по типу модели**

Структурированные базы данных:

* теоретико-графовые:
* иерархические;
* сетевые.
* теоретико-множественные:
* реляционные;
* постреляционные;
* многомерные.
* объектно-ориентированные.

**Иерархическая модель БД**

Эта модель отражает множественные отношения между объектами реального мира, когда один объект выступает в роли главного, с которым связано большое количество подчиненных объектов.

**Система IMS (Information Management System)** – результат работы фирм IBM и Rockwell по созданию СУБД для поддержки лунного проекта Аполлон.

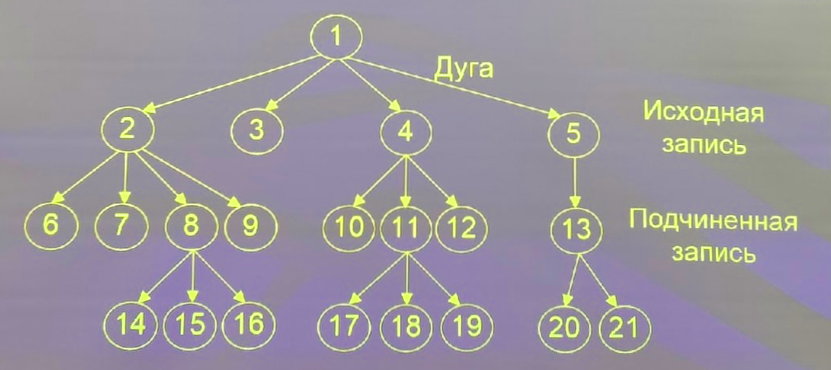
Информационные единицы:

* **поле** – минимально именованная единица данных, доступная пользователю с помощью СУБД;
* **запись** – основная единица обработки БД:
* **тип записи** – именованная совокупность полей данных с указанием их типов; **простой тип** – состоит из полей одного и того же типа; **составной** – разные типы данных;
* **экземпляр записи** – набор конкретных значения в последовательности, соответствующей типу этой записи.
* **групповое отношение –** иерархическое отношение между записями двух типов;
* **база данных**.

СТОЛ (ЦВЕТ символьный, ДЛИНА числовой, ШИРИНА числовой, ВЫСОТА числовой, МАТЕРИАЛ символьный)

СТОЛ (Коричневый, 1.30, 0.9, 0.75, дерево)

**Иерархическая модель БД**



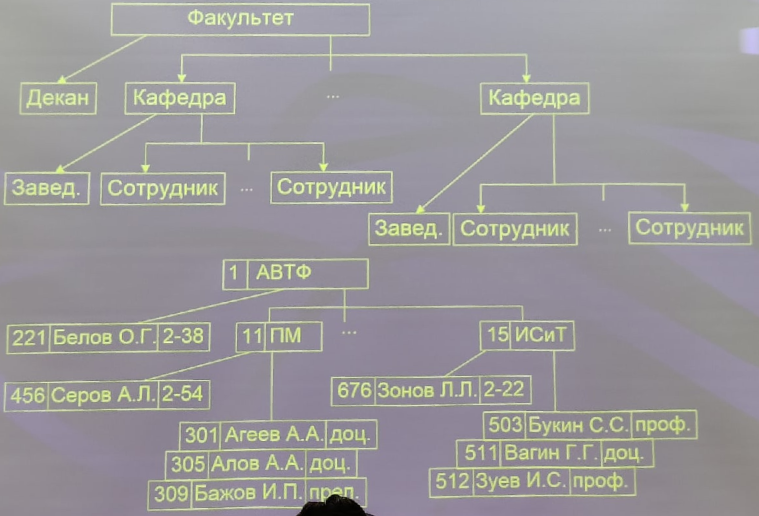
**Каждая вершина** – тип записи, **дуги** – связи между записями двух типов. **Экземпляр групповых отношений** – задается исходной записью и множеством подчинений записей

Экземпляр-потомки одного типа, связанных с одной и той же записью предка называются **близнецами**.

**Дерево** – совокупность корневой записи и множества подчиненных записей.



**Иерархическая модель БД** – упорядоченная совокупность экземпляров деревьев, каждое из которых содержит экземпляры записей.



**Ограничения** иерархической модели БД:

* существует только одна корневая запись, которая не связана ни с какой исходной записью;
* каждая логически исходная запись может быть связана с произвольным числом логически подчиненных записей;
* каждая логически подчиненная запись может быть связана только с одной логически исходной записью.

Обход всех элементов – сверху вниз и слева направо.

Операторы **манипулирования** иерархически организованными данными:

найти указанное дерево БД;

* перейти от одного дерева к другому;
* перейти внутри дерева от одной записи к другой;
* вставить новую запись в указанную позицию;
* удалить текущую запись.

**Достоинства:**

* эффективное использование памяти и неплохие показатели времени выполнения основных операций.

**Недостатки:**

* громоздкость для обработки данных со сложными связями и сложность понимания.

**Сетевая модель БД**

**Сетевая модель БД** – совокупность взаимосвязанных наборов.

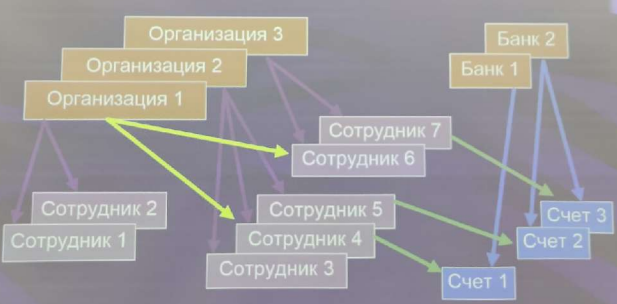
Позволяет встроить структуру данных более общего вида, чем иерархическая модель.

Вместо понятия групповых отношений вводятся понятие набора.

CODASYL (Conference on Data Systems Language), 1971.

Набор:

* один и тот же тип записи может участвовать в нескольких наборах;
* для любых двух типов записей может быть задано любое количество наборов, которые их связывают.



**Каждый экземпляр набора** – список записей-членов набора, поставленных в соответствие записи-владельцу набора.

Записи-члены набора могут быть упорядочены:

* произвольным образом;
* хронологическим в порядке их поступления;
* обратнохронологическим;
* упорядоченным по некоторому ключу.

**Достоинства:**

* эффективное использование памяти и неплохие показатели времени выполнения основных операций; по сравнению с иерархической моделью предоставляется больше возможностей по созданию связей.

**Недостатки:**

* громоздкость для обработки данных со сложными связями и сложность понимания.

**Операторы манипулирования**

* найти конкретную запись;
* перейти от записи-владельца к записи члену в некотором экземпляре набора;
* перейти от записи-члена набора к его владельцу в некотором наборе;
* перейти к следующей записи в некотором экземпляре набора;
* создать новую запись;
* удалить текущую запись;
* модифицировать запись;
* включить запись в набор;
* исключить запись из текущего экземпляра набора;
* переставить в другой экземпляр набора.

**Реляционная модель БД**

**Реляционная модель БД –** все данные представлены в виде прямоугольных заполненных таблиц и все операции над БД сводятся к манипуляции таблиц.

Объекты реального мира – сущность; свойства характеристик сущности – атрибуты.

Эдгар Франк Кодд, 1969-70 гг.

**Реляционная модель данных (РМД)** разработана на основе математической теории отношений и опирается на систему базовых понятий.

|  |  |
| --- | --- |
| **Элемент РМД** | **Форма представления** |
| Отношение | Таблица |
| Схема отношения | Заголовок таблицы |
| Кортеж | Строка таблицы |
| Атрибут | Заголовок столбца таблицы |
| Значение атрибута | Значение поля таблицы |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Деталь** | **Шифр** | **Название** | **Вес** | **Материал** |
|  | Д1 | Втулка | 0.8 | Сталь |
|  | Д2 | Педаль | 1.0 | Сталь |
|  | Д3 | Крыло | 0.8 | Пластмасса |
|  | Д4 | Крыло | 0.9 | Пластмасса |

**Домен** – множество всех возможных значений атрибута объекта.

Домен НАЗВАНИЕ:

базовый тип – строка символов

логическое выражение: строки начинаются с букв, за исключением букв ь, ъ, ы.

Ai – атрибут, Di – его домен.

Ai – материал, Di = dom (Ai) = {«Сталь», «Олово», «Цинк», …}

**Схема отношения** – именованное множество пар {имя атрибута, имя домена}.

ДЕТАЛЬ (Шифр, Название, Вес, Материал).

**Схема БД** – набор именованных схем отношений.

Для схемы отношения R (A1, A2, …, An) k-ый кортеж: <a1k, a2k, …, ank>, aik – значение i-ого атрибута в k-ом кортеже.

Отношение r со схемой отношения R (A1, A2, …, An).

**Степень отношений** – число атрибутов.

**Сущность отношения** – количество кортежей.

**Отношения** – множество кортежей, соответствующие одной и той же схеме.

**Лекция 3**

Свойства отношений:

1. Отсутствие кортежей-дубликатов (возникает из-за определения отношения как множество кортежей)

**Первичный ключ** – набор атрибутов, значения которых однозначно определяют каждый кортеж отношения.

Отношения R(A1, A2, …, An).

K=(Ai, Aj, …, Am) – возможный ключ (в произвольный момент времени никакие два разных кортежа не имеют одного и того же значения для Ai, Aj, …, Am)

А) уникальность;

Б) минимальность;

**Реляционная модель данных**

|  |  |  |  |
| --- | --- | --- | --- |
| **r** | **A** | **B** | **C** |
|  | a1 | b1 | c1 |
|  | a1 | b2 | c2 |
|  | a1 | b3 | c3 |
|  | a2 | b1 | c1 |
|  | a2 | b2 | c2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **r** | **A** | **B** | **C** |
|  | a1 |  | c1 |
|  | a1 | c2 |
|  | a1 | c3 |
|  | a2 | c1 |
|  | a2 | c2 |

А – нет

В – нет

С – нет

АВ – да

ВС – нет

АС – да

АВС – да

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Деталь** | **Шифр** | **Название** | **Вес** | **Материал** |
|  | Д1 | Втулка | 0.8 | Сталь |
|  | Д2 | Педаль | 1.0 | Сталь |
|  | Д3 | Крыло | 0.8 | Пластмасса |
|  | Д4 | Крыло | 0.9 | Пластмасса |

1. отсутствие упорядоченности кортежей (свойство также является следствием определения отношения как множество кортежей)
2. отсутствие упорядоченности атрибутов (к каждому атрибуту обращаемся по имени, следует из того, что схема отношений – множество пар, включающие атрибуты и его данные)

**Связывание таблиц**

В теоретико-графовых моделях объекты и связи представляются по-разному, а в реляционной модели связи и объекты представляются в виде отношений.

**Внешний ключ** – атрибут (или множество атрибутов) отношения, являющийся первичным ключом другого отношения.

|  |  |  |  |
| --- | --- | --- | --- |
| **Деталь** | **Шифр** | **Название** | **Материал** |
|  | Д1 | Втулка | Сталь |
|  | Д2 | Педаль | Сталь |
|  | Д3 | Крыло | Пластмасса |

|  |  |  |
| --- | --- | --- |
| **Поставщик** | **Имя** | **Телефон** |
|  | П1 | 12345 |
|  | П2 | 45678 |

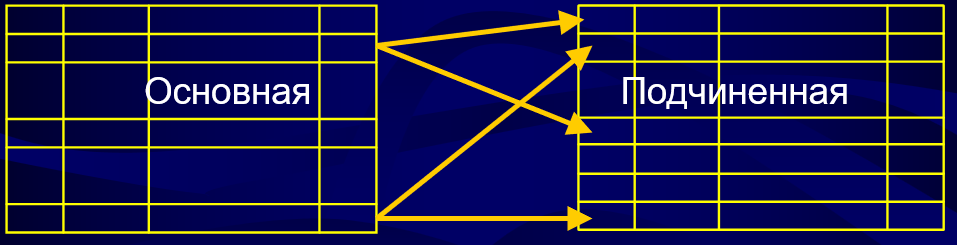
|  |  |  |  |
| --- | --- | --- | --- |
| **Поставка** | **Шифрдетали** | **ИмяПоставщика** | **Цена** |
|  | Д1 | П1 | 100 |
|  | Д2 | П2 | 120 |
|  | Д3 | П3 | 100 |

Шифр детали и имя поставщика – внешние ключи.

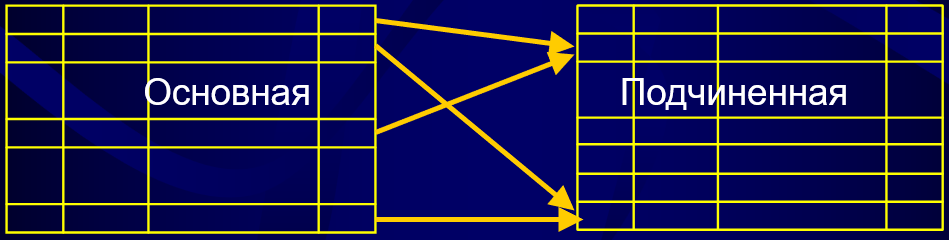
При связывании двух таблиц выделяют основную таблицу и подчиненную, суть связывания состоит в установлении соответствия полей основной таблицы и подчиненной.

Виды отношений:

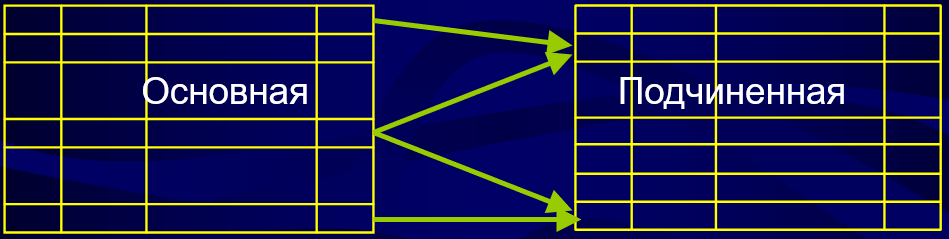
1. **один к одному (1:1)** – к каждой записи одной таблицы соответствует только одна запись другой таблицы, таблицы считаются равноправными и их не разделяют. Образуется в том случае, если поля связи таблиц являются ключевыми. На практике при наличии связи 1:1 таблицы объединяются в единое целое (экономия места в памяти), однако возможны случаи, когда это делать не нужно (1 - необходимо ускорить обработку информации, 2 - повысить удобство работу пользователя с общей информацией, 3 - обеспечить защищенность информации);
2. **один ко многим (1:М)** – одной записи одной таблицы соответствует несколько записей подчиненной таблицы;



1. **многие к одному (М:1)** – нескольким записям одной таблицы соответствует одна запись подчиненной таблицы;



1. **многие ко многим (М:N)** – нескольким записям одной таблицы соответствует несколько записей подчиненной таблицы.



|  |  |  |
| --- | --- | --- |
| **Лектор** | **Курс** | **Фамилия** |
|  | Физика | Иванов |
|  | Физика | Петров |
|  | Химия | Алексеев |
|  | Алгебра | Лыков |

|  |  |  |
| --- | --- | --- |
| **Учебник** | **Курс** | **Название** |
|  | Физика | Динамика |
|  | Физика | Оптика |
|  | Химия | Учебник1 |
|  | Химия | Учебник2 |

Реляционная модель данных:

* структурная часть (единственной структурой данных в реляционной модели являются отношения);
* манипуляционная часть;
* целостная часть.

Механизмы манипулирования реляционными БД:

* реляционная алгебра;
* реляционное вычисление.

Требования целостности:

* требование целостности сущностей (любой кортеж отношения должен быть отличим от любого другого кортежа этого отношения – не должно быть двух совпадающих кортежей);
* требование целостности по ссылкам (3 требования):
* каждой записи основной таблицы соответствует нуль или более записей подчиненной таблицы;
* в подчиненной таблице нет записей, которые не имеют родительских записей в основной таблице;
* каждая запись подчиненной таблицы имеет только одну родительскую запись основной таблицы.

1. Добавление новые записей (данные сначала добавляются в основную таблицу, а затем в подчиненную, заполнение поля связи в подчиненной таблице контролируется на совпадение со значениями поля связи в основной таблице, если значение не совпадает, то ввод такой строки должен быть заблокирован)
2. Модификация записей (при модификации поля связи в подчиненной таблице см. предыдущее требование, при модифицировании поля связи в основной таблице требуется подчинить одному из правил):

* редактировать записи, у которых нет подчиненных записей, иначе блокировать модификацию полей связи;
* либо изменения в полях связи основной записи мгновенно передавать всем дочерним записям.

1. Удаление записей (в подчиненной таблице происходит бесконтрольно)

* удалять можно запись, которая не имеет подчиненных записей (иначе запретить удаление записи);
* либо удалять запись вместе со всеми подчиненными записями (каскадное удаление).

**Реляционная алгебра**

Поскольку отношения являются множествами, то операции манипулирования ими могут основываться традиционными операциями над множествами.

Теоретико-множественные операции:

* объединение;
* пересечение;
* разность;
* произведение.

Специальные реляционные операции:

* ограничение (селекция, выборка);
* проекция;
* соединение;
* деление.

Операции могут выполняться над одним отношением или над двумя отношениями. В бинарных операциях оба отношения должны быть совместимы по структуре. Структура результирующего отношения по определенным правилам наследует структуру…

**Теоретико-множественные операции**

**Объединение R1∪R2** одинаковой размерности - является отношение R, содержащие все кортежи исходных, за исключением повторений.

**Пересечение R1⋂R2** одинаковой размерности - является отношение R, содержащее кортежи, одновременно принадлежащие двум отношениям.

**Вычитание R1 - R2** одинаковой размерности - является отношение R, содержащие кортежи, принадлежащие R1, но не принадлежащие кортежи R2.

**Произведение R1 х R2** отношения R1 размерности n1 и отношения R2 размерности n2 – отношение R степени (n1+n2), содержащее кортежи, в которых первые n1 элементов принадлежат множеству R1, а последние n2 элементов – множеству R2.

**Лекция 4**

**Реляционная алгебра**

**Специальные** реляционные операции:

1. **Выборка (R)** отношения R по формуле f представляет собой отношение, состоящее из тех кортежей отношения R, которые удовлетворяют истинности формулы f. Формула f - логическое выражение, которое содержит имена атрибутов, константы, логические операции, операции сравнения и скобки.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Деталь** | **Шифр** | **Название** | **Вес** | **Материал** |
|  | Д1 | Втулка | 0.8 | Сталь |
|  | Д2 | Педаль | 1.0 | Сталь |
|  | Д3 | Крыло | 0.8 | Пластмасса |
|  | Д4 | Крыло | 0.9 | Пластмасса |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **(Деталь)** | **Шифр** | **Название** | **Вес** | **Материал** |
|  | Д2 | Педаль | 1.0 | Сталь |
|  | Д4 | Крыло | 0.9 | Пластмасса |

1. **Проекция**  (R) - отношение R на атрибуты с именами А1, А2, …, Аn - подмножества всех атрибутов отношения R, является новым отношением с заголовком А1, А2, …, Аn и телом, содержащим кортежи исходного отношения за исключением повторяющихся.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Деталь** | **Шифр** | **Название** | **Вес** | **Материал** |
|  | Д1 | Втулка | 0.8 | Сталь |
|  | Д2 | Педаль | 1.0 | Сталь |
|  | Д3 | Крыло | 0.8 | Пластмасса |
|  | Д4 | Крыло | 0.9 | Пластмасса |

|  |  |  |
| --- | --- | --- |
| **П Название, Материал (Деталь)** | **Название** | **Материал** |
|  | Втулка | Сталь |
|  | Педаль | Сталь |
|  | Крыло | Пластмасса |

1. Результат **деления R1/R2** отношения R1(А, В) на отношение R2(В) – отношение с заголовком А и телом, состоящим из кортежей r таких, что в отношении R1 имеются кортежи (r,s), причем множество значений включает все значения атрибута В отношения R2.

|  |  |  |
| --- | --- | --- |
| **R1** | **Фамилия** | **Курс** |
|  | Иванов | Алгебра |
|  | Иванов | Физика |
|  | Петров | Физика |
|  | Петров | Черчение |
|  | Петров | Химия |
|  | Сидоров | Физика |
|  | Сидоров | Химия |
|  | Алексеев | Физика |

|  |  |
| --- | --- |
| **R2** | **Курс** |
|  | Физика |
|  | Химия |

|  |  |
| --- | --- |
| **R1/ R2** | **Фамилия** |
|  | Петров |
|  | Сидоров |

1. **Соединение R1 f R2** представляет собой новое отношение, содержащее кортежи декартового произведения R1 и R2 и удовлетворяющее формуле f.

* (тета) соединение;
* эквисоединение (формула f содержит только оператор равенства);
* естественное соединение R1 R2 - эквисоединение отношения R1 и R2, выполненное по всем общим атрибутам из результатов которого исключается один экземпляр общего атрибута. Шифр детали зачеркнули и написали шифр.

|  |  |  |
| --- | --- | --- |
| **Деталь** | **Шифр** | **Цвет** |
|  |  | Красный |
|  |  | Красный |
|  |  | Синий |

|  |  |  |
| --- | --- | --- |
| **Поставщик** | **Имя** | **ШДетали** |
|  | П1 | Д1 |
|  | П2 | Д2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Поставщик Деталь** | **Имя** | **Шифр** | **Цвет** |
|  | П1 | Д1 | Красный |
|  | П2 | Д2 | Красный |
|  | П2 | Д2 | Синий |

**Нормализация баз данных**

Одним из преимуществ реляционной модели является наличие формального механизма оценки качества логической структуры базы данных методами нормализации.

Избыточность данных приводит к:

* увеличению объему увеличению объему хранимой информации;
* может вызвать нарушение целостности данных;

|  |  |  |  |
| --- | --- | --- | --- |
| **№сотрудника** | **Фамилия** | **Должность** | **Проект** |
| 1111 | Алексеев А.А. | Программист I | Альфа |
| 1111 | Алексеев А.А. | Программист I | Омега |
| 2222 | Иванов И.И. | Вед. программист | Альфа |
| 2222 | Иванов И.И. | Вед. программист | Омега |
| 2222 | Иванов И.И. | Вед. программист | Гамма |
| 2222 | Иванов И.И. | Вед. программист | Сигма |
| 3333 | Сидоров С.С. | Программист II | Альфа |

Аномалии модификации:

* **аномалия удаления** – проявляется в том, что удаляя факты, относящиеся к одной сущности мы непроизвольно удаляем факты, относящиеся к другой сущности;
* **аномалия добавления** – нельзя поместить в БД факты об одной сущности, не указав дополнительно факты о другой сущности;
* **аномалия обновления** – для внесения изменения об одном факте необходимо выполнить множественное изменение во всей БД.

**Нормализация** - разбиение исходного отношения на два и более, обладающих лучшими свойствами при обновлении, удалении и добавлении данных. Цель сводится к получению такой БД, в которой каждый факт появляется только в одном месте (исключена избыточность).

Процесс проектирования БД включается в процесс нормализации схем отношений, где каждая следующая нормальная форма обладает свойствами лучшими, чем предыдущие. Каждой нормальной форме соответствует определенный набор ограничений.

Отношение находится в **первой нормальной форме (1НФ)**, если все его атрибуты являются простыми.

|  |  |  |  |
| --- | --- | --- | --- |
| **№сотрудника** | **Фамилия** | **Должность** | **Проект** |
| 1111 | Алексеев А.А. | Программист I | Альфа, Омега |
| 2222 | Иванов И.И. | Вед. программист | Альфа, Омега, Гамма, Сигма |
| 3333 | Сидоров С.С. | Программист II | Альфа |

**Функциональные зависимости**

АВ - атрибут В функционально зависит от атрибута А, если каждому значению А соответствует только одно значение В (А и В могут быть составными).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Фамилия** | **Должность** | **Зарплата** | **Проект** | **Задание** |
| Алексеев А.А. | Программист I | 100 | Альфа | Задание1 |
| Алексеев А.А. | Программист I | 100 | Омега | Задание2 |
| Иванов И.И. | Вед. программист | 150 | Альфа | Задание3 |
| Сидоров С.С. | Программист II | 150 | Альфа | Задание1 |

Функциональная зависимость является **полной**, если атрибут В не зависит от любого подмножества атрибута А. Если А – простой атрибут, то зависимость всегда полная.

ФамилияЗарплата

ФамилияДолжность

ДолжностьЗарплата

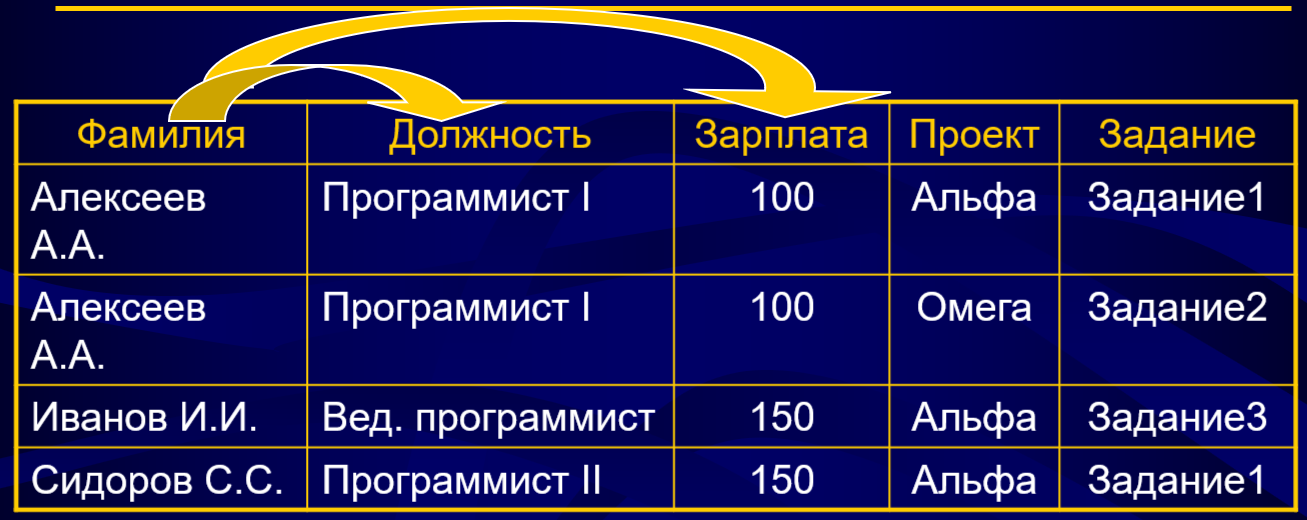
Фамилия, ПроектЗадание

Фамилия, ПроектДолжность

Фамилия, ПроектЗарплата

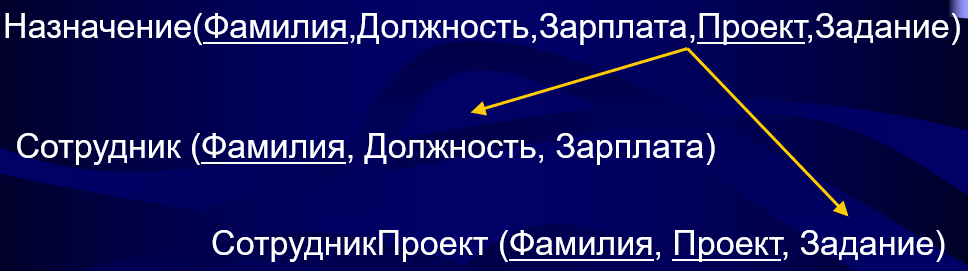
Фамилия, ПроектЗадание, Должность, Зарплата

Отношение находится во **второй нормальной форме (2НФ)**, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от первичного ключа. Отношение будет находится в 2НФ, если его первичный ключ не составной.



|  |  |  |
| --- | --- | --- |
| **Фамилия** | **Должность** | **Зарплата** |
| Алексеев А.А. | Программист I | 100 |
| Алексеев А.А. | Программист I | 100 |
| Иванов И.И. | Вед. программист | 150 |
| Сидоров С.С. | Программист II | 150 |

|  |  |  |
| --- | --- | --- |
| **Фамилия** | **Проект** | **Задание** |
| Алексеев А.А. | Альфа | Задание1 |
| Алексеев А.А. | Омега | Задание2 |
| Иванов И.И. | Альфа | Задание3 |
| Сидоров С.С. | Альфа | Задание1 |

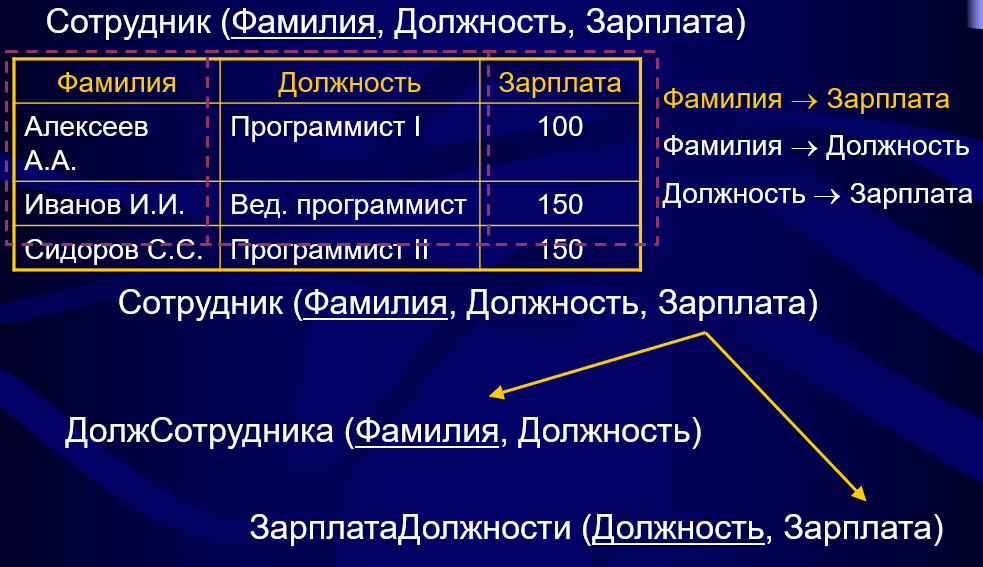


Первичным ключом является Фамилия и Проект, однако есть зависимость неключевых атрибутов.

Отношение находится в **третьей нормальной форме (3НФ)**, если оно находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

С : А С, С В и С А

В Сотрудник Проект нетранзитивная зависимость.



**Лекция 5**

**Проектирование структуры БД**

* **концептуальное** проектирование (целью является определение содержания базы данных: данные о каких объектах должны содержаться в БД и какими атрибутами они характеризуются)
* **логическое** проектирование (целью является определение состава и структуры таблиц БД)

*выполнение двух первых этапов не зависит от того, в какой СУБД предполагается реализация*

* **физическое** проектирование (целью является реализация спроектированной БД на конкретной СУБД и обеспечение выполнения всех требований к ней)

Концептуальное проектирование

Какие данные должны храниться в базе данных?

* анализ задач;
* анализ хранимых объектов.

Концептуальное проектирование заключается в формализованном описании предметной области, это описание должно быть таким, чтобы, с одной стороны, можно было проанализировать корректность схемы разрабатываемого проекта БД, с другой стороны, не должно быть привязано к конкретной БД.

Любая БД создается для решения определенных прикладных задач, в простейшем случае для хранения и выдачи данных, в более сложном варианте для решения задач, использующие данные из БД в качестве исходных данных.

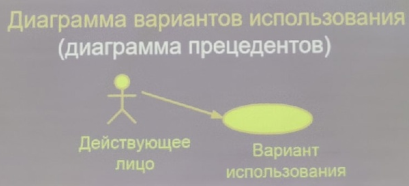
Какие данные должны храниться в базе данных?

Анализ задач (нужно ответить на вопросы «для решения каких задач используется БД?» и «какие данные нужны для решения этих задач?»)

Анализ хранимых объектов (нужно ответить на вопросы «данные о ком или о чем должны храниться в БД» и «какими атрибутами они характеризуются?»)

Анализ задач

**UML** – (Unified Modeling Language – унифицированный язык моделирования).



**Диаграмма вариантов использования** (диаграмма прецедентов). Главное назначение: формализация функциональных требований к системе с помощью действующих лиц и вариантов использования. Действующее лицо – потенциальный пользователь системы, в большинстве случаев – человек, но не исключен механизм. Вариант использования – задачи, с которыми действующие лица обращаются к БД.

Обычно предполагается, что действующих лиц не больше 20, а вариантов использования не больше 50. Надо фокусироваться на том, какие задачи будут решаться будут решаться в создаваемой БД, а не на том, как именно это будет реализовано.

Составление ДВИ позволяет:

1. **Экспертам** предметной области описать взгляд на систему извне с такой степенью детализации, что **разработчики** сумеют сконструировать ее внутреннее представление.
2. Варианты использования позволяют разработчикам понять **назначение** разрабатываемой БД
3. Варианты использования являются основой **тестирования** элементов БД на всем протяжении его жизненного цикла.

Последовательность составления ДВИ:

1. Выделить действующие лица (надо четко ответить на вопрос «чем одно действующее лицо отличается от другого?»)
2. Для каждого действующего лица выделить варианты использования. После этого составить общий список вариантов использования.
3. Связать между собой действующие лица и варианты использования
4. Проанализировать в первом приближении наличие общих подзадач у разных вариантов использования, при наличии выделить их и установить отношения.

**БД для автоматизации работы магазина**. По телефону потенциальные покупатели могут получить справки о наличии товара, о ценах и характеристиках и, при желании, могут забронировать товар. Забронированный товар должен быть куплен в течение дня (забронированный, но не купленный товар после закрытия магазина считается свободным для продажи).

Магазин торгует товарами малых размеров и массы, поэтому доставка товара покупателям не осуществляется.

Покупатель, пришедший в магазин, может получать у продавцов информацию о наличии товара, о ценах и характеристиках.

Кроме того, продавец может выписывать счет на оплату товара, копия счета передается на склад, кладовщик доставит товар на выдачу и сотрудник на выдаче, убедившись, что товар оплачен, передает его покупателю. Товар между выписыванием счета и доставкой кладовщиком на выдачу считается забронированным. Продавец может выписывать счет и на забронированный товар, а также забронировать товар. Забронированный у продавца товар обрабатывается аналогично забронированному по телефону. Выписанный, но не оплаченный в течение трех часов товар, возвращается на склад и считается свободным.

Автоматизация расчетов с покупателями и ведения бухгалтерского учета на данном этапе не рассматриваются. Менеджер магазина отслеживает расход товаров и состояние склада и принимает на этой основе решения о заказе новых партий у поставщиков или о продаже залежавшегося товара.

Действующие лица:

− консультант у справочного телефона магазина,

− продавец,

− кладовщик,

− сотрудник на выдаче,

− менеджер.

Рассмотрим некоторые варианты использования.

Консультант на телефоне:

1. Выдача информации о характеристиках и ценах товаров.

2. Выдача информации о наличии товара на складе.

3. Резервирование товара.

4. Аннулирование брони по желанию покупателя.

Продавец:

1. Выдача информации о характеристиках и ценах товаров.

2. Выдача информации о наличии товара на складе.

3. Резервирование товара.

4. Аннулирование брони по желанию покупателя или после закрытия магазина.

5. Выписывание счета.

6. Передача копии счета на склад.

Кладовщик:

1. Изменение количества товара на складе после отпуска.

2. Изменение количества товара на складе после возврата с выдачи.

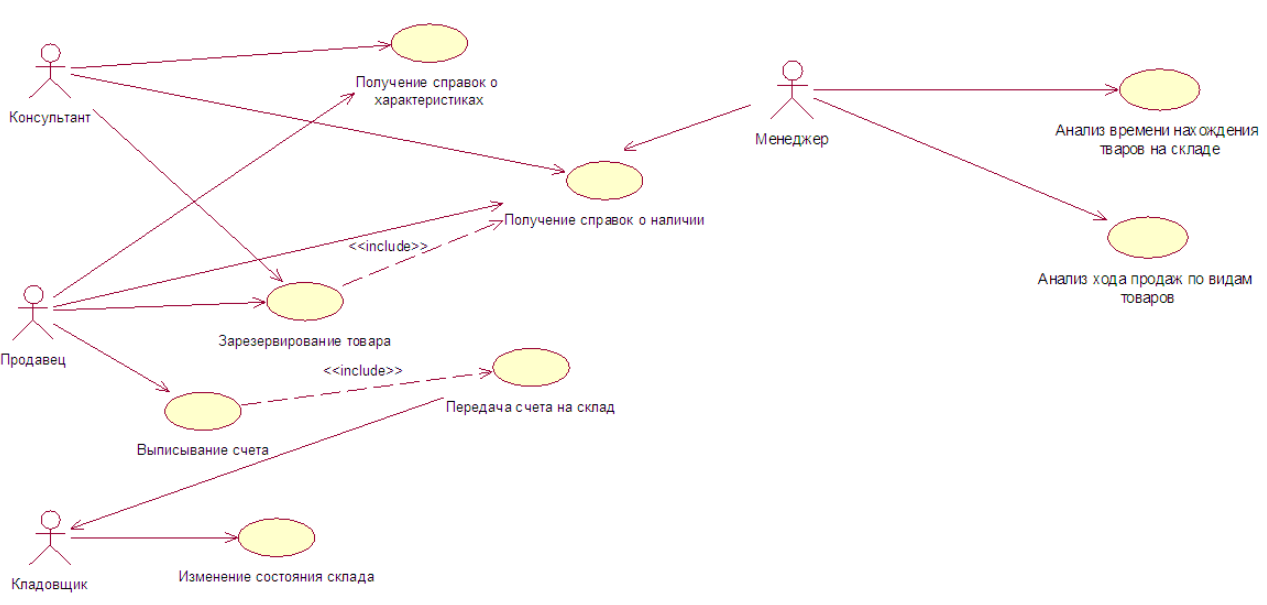
Менеджер:

1. Выдача информации о наличии товара на складе.

2. Анализ хода продаж отдельных видов товаров.

3. Анализ времени нахождения товара на складе.

Сотрудник на выдаче не осуществляет ввод данных и поэтому он не рассматривается в дальнейшем как действующее лицо. Рассмотрим составленные диаграммы вариантов использования.



По диаграмме вариантов использования мы определяем какие объекты надо хранить в БД для решения выделенных задач.

Анализ задач

1. Характеристики продаваемых товаров, при этом состав характеристик у разных товаров может быть различным.

2. Наличие продаваемых товаров на складе, регистрация их поступления.

3. Истории продаж.

4. Сведения о резервировании товара.

Необходимые данные можно классифицировать по частоте их изменения:

− условно-постоянные данные (например, характеристики продаваемых товаров);

− данные, которые оперативно обновляются при каждом решении задачи (какой товар, в каком количество продан, поступил, когда).

Для определения списка объектов необходимо проанализировать каждую задачу: какие исходные данные требуются для решения задачи (они есть в БД или носят разовый характер?); какие данные получаются в результате решения задач (надо ли их хранить в БД или они имеют временный характер?).

При проведении анализа предметной области возникают технические трудности: нужно добиться, чтобы одинаковые данные назывались одинаково, а разные данные – по-разному.

**Анализ выделенных задач**

Исходные данные, необходимые для их решения:

* какие из них требуются вводить каждый раз заново;
* какие имеют условно-постоянный характер и меняются относительно редко.

Данные, получающиеся в результате решения задачи:

* какие из них необходимо хранить в БД;
* какие имеют одноразовый характер и будут выданы пользователю.

**Анализ хранимых объектов**

Цель: **приближение** - определить, данные каких объектов будут храниться в БД и каким перечнем атрибутов они характеризуются.

Первоначальный перечень атрибутов можно получить в результате анализа задач, выделенных на первом этапе.

Можно создать задел на будущее - включить атрибуты, значения которых потребуются в будущем при расширении поставленных задач.

Рекомендации по **определению состава атрибутов**:

1. **Имя атрибута** должно быть существительным (желательно в единственном числе)
2. **Типы атрибутов** пока концептуальны и не должны быть связаны со средой реализации
3. Среди атрибутов не стоит иметь атрибуты, значения которых можно легко **вычислить**, зная значения других (пример: возраст)
4. Наличие **составных атрибутов** нецелесообразно, если составные части могут использовать независимо друг от друга.
5. При наличии атрибутов, принимающих значение из **фиксированного множества значений**, целесообразно предусмотреть **кодификаторы**.

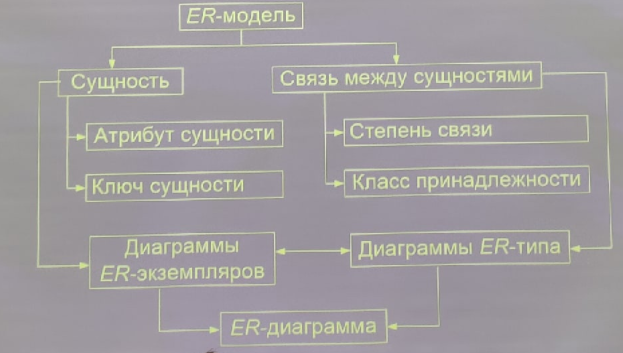
**Лекция 6**

**Семантическое моделирование данных**

Концептуальное моделирование данных направлено на отображение семантики (смысла) предметной области на модель БД.

ER-модели (Entity – сущность, Relation – связь).

Модель была предложена Ченом, 1976 г.

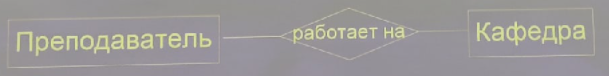


Представляет собой формальную конструкцию, которая сама по себе не имеет никаких средств для ее визуализации.

**ER-диаграммы**

**Сущность** – это объект, информация о котором должна храниться в БД. Название сущности – существительное в единственном числе.

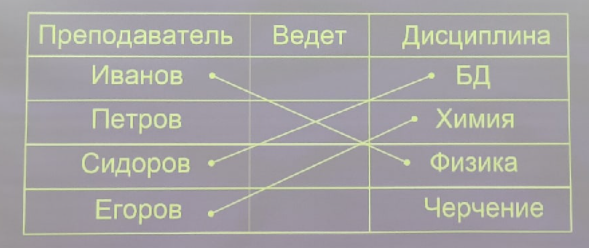
Нотации для построения ER-модели: нотация Питера Чена, нотация Мартина, нотация IDEF1X, нотация Баркера и др.



Название связи представляется глаголом.

* диаграммы ER-экземпляров
* диаграммы ER-типа (ER-диаграммы)

**Диаграмма ER-экземпляров**



**Диаграмма ER-типа**



**Степени связи** сущностей: 1:1, 1:М (М:1) или М:М.

В общем случае между двумя сущностями может быть установлено любое количество связи.

Класс принадлежности (КП) сущности может быть:

* **обязательный (О)** – класс принадлежности сущности обязательный, если каждый экземпляр сущности должен участвовать в связи

**(ФОТО)**

* **необязательный (Н)** – класс принадлежности сущности необязательный, если не каждый экземпляр сущности должен участвовать в связи.

**(ФОТО) попросить**

**Пример 1. Связь типа 1:1, Н-Н**

Диаграмма ER-экземпляров

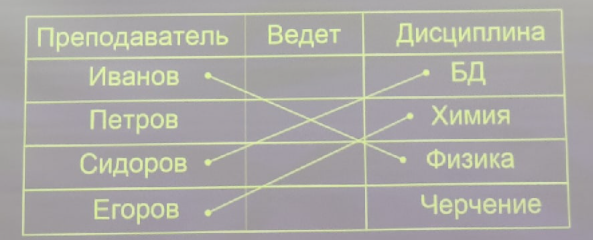


Диаграмма ER-типа



Каждый преподаватель ведет не более одной дисциплины (или ни одной) и наоборот: каждая дисциплина ведется не более, чем одним преподавателем (или никем).

**Пример 2. Связь типа 1:1, О-О**

Диаграмма ER-экземпляров

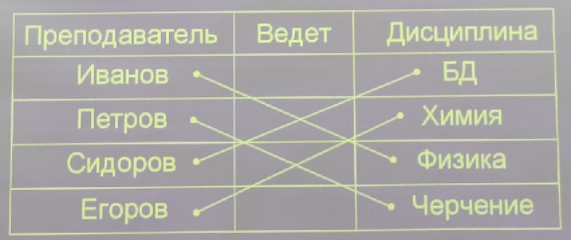
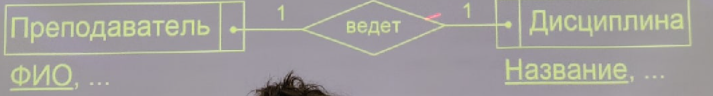


Диаграмма ER-типа



Каждый преподаватель обязательно ведет только одну дисциплину, каждая дисциплина считается только одним преподавателем.

**Пример 3. Связь типа 1:М, Н-О**

Каждый преподаватель может вести много дисциплин или ни одной, каждая дисциплина ведется только одним преподавателем.



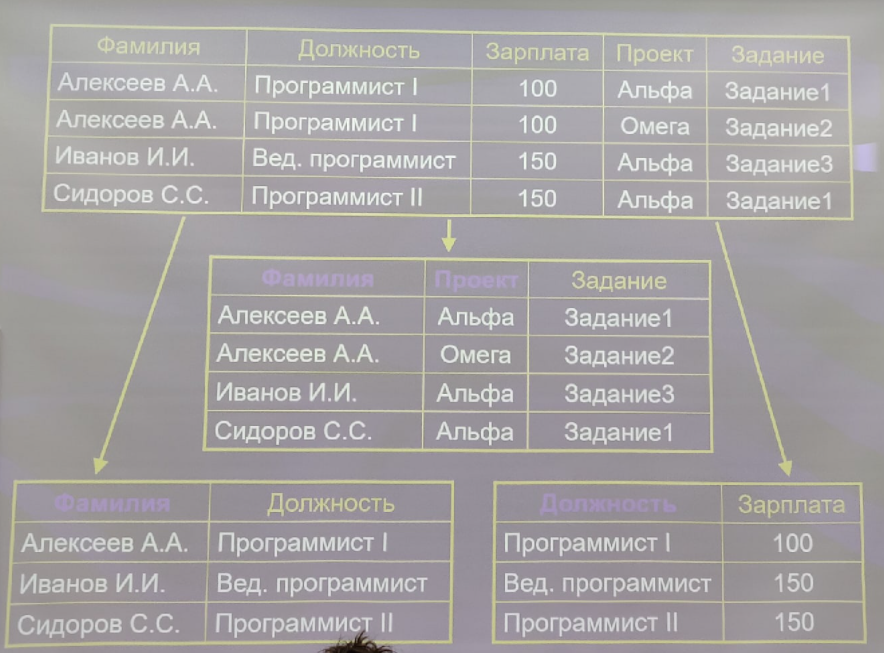
Каждый преподаватель ведет одну дисциплина (или ни одну), каждая дисциплина ведется многими преподавателями (или никем)

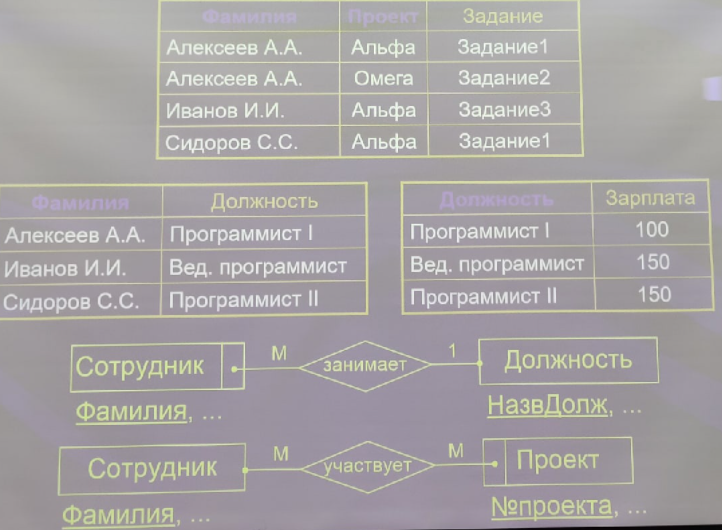


Каждый преподаватель обязательно ведет более одной дисциплины, а дисциплина читается многими преподавателями (или никем)



Каждый сотрудник занимает одну должность, сотрудник обязательно что-то занимает, каждую должность могут занимать несколько сотрудников, при это должность является необязательной.





Каждый сотрудник может участвовать в нескольких проектах, но есть сотрудники, которые не участвуют в проекте, проект предполагает множество работающих сотрудников, проекта без сотрудника не бывает.

**Правила формирования отношения**

Следующим этапом после построения ER-модели является формирование отношения. Процесс проектирования БД является итерационным, поскольку часто происходит возврат к предыдущим этапам для пересмотра ранее принятых решений.

Правила:

**Правило 1**. Если степень связи 1:1 и класс принадлежности обеих сущностей обязательный, то требуется всего одно отношение, объединяющее эти сущности. При этом *ключом* этого отношения может быть ключ любой сущности.



**Правило 2**. Если степень связи 1:1 и КП одной из сущностей обязательный, а другой − необязательный, то для каждой из сущностей формируется по одному отношению. Причем ключ сущности с необязательным КП добавляется в качестве атрибута в другое отношение.



**Правило 3**. Если степень связи 1:1 и КП обеих сущностей − необязательный, то требуются три отношения: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.



**Правило 4**. Если степень связи 1:M (M:1) и КП М-связной сущности − обязательный, то, независимо от КП другой сущности, достаточно сформировать два отношения: по одному для каждой сущности с соответствующими ключами. При этом ключ односвязной сущности добавляется в качестве атрибута в отношение для многосвязной сущности.



**Правило 5**. Если степень связи 1:M (M:1) и КП М-связной сущности − необязательный, то необходимо формирование трех отношений: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.



**Правило 6**. Если степень связи M:M, то необходимо формирование трех отношений: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.



**Пример формирования отношений на основе анализа ER-диаграммы**

А) удовлетворяет правилу 4, согласно которому должно появиться два отношения

Б) удовлетворяет правилу 6, согласно которому должно появиться три отношения



Поскольку отношение Проект кроме одного отношения ничего не имеет, то его можно исключить.

**Лекция 7**

**Логическое проектирование БД**

**Цель**: определение состава и структура таблиц БД на основе результатов концептуального проектирования и проверка полученной модели с помощью методов нормализации.

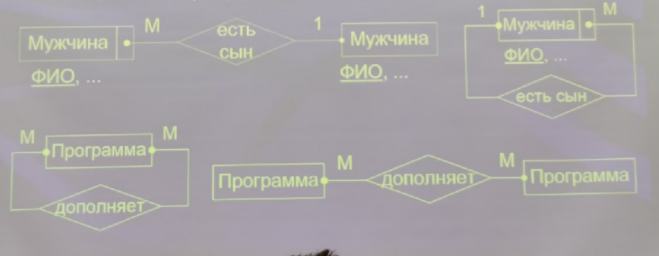
**Стадии** логического проектирования

1. исключения особенностей, несовместимых с реляционной моделью (необязательный этап);
2. формирование отношений на основе логической модели данных;
3. проверка отношений с использованием средств нормализации;
4. определение ограничений целостности.

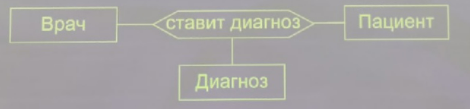
**Этап 1**. Исключения особенностей, несовместимых с реляционной моделью

**Рекурсивная связь** – вид связи, при котором сущность соединена сама с собой

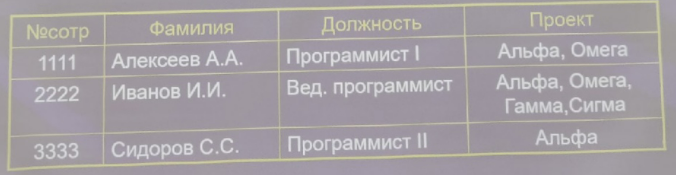
1. удаление рекурсивных связей



1. удаление сложных связей (связь, в которой участвуют 3 или более сущности)



1. удаление многозначных атрибутов

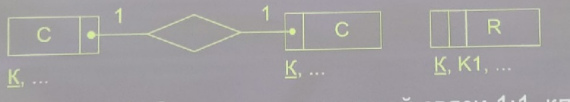


**Этап 2**. Формирование отношений на основе логической модели данных

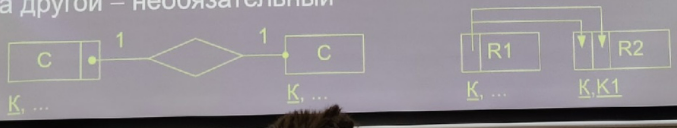
На данном этапе определяется набор отношений, необходимый для представления сущностей, связей и атрибутов. Связь между двумя сущностями отображается с помощью механизма первичный ключ и внешний ключ.

1. рекурсивные связи

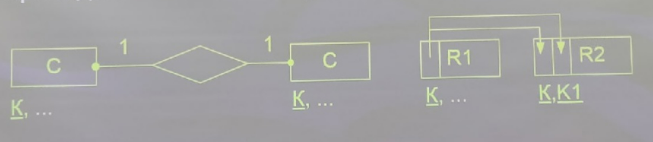
**Правило 1**. Степень рекурсивной связи 1:1, класс принадлежности обеих сущностей обязательный.



**Правило 2**. Степень рекурсивной связи 1:1, класс принадлежности одной из сущностей обязательный, а другой – необязательный.



**Правило 3**. Степень рекурсивной связи 1:1, класс принадлежности обеих сущностей – необязательный.

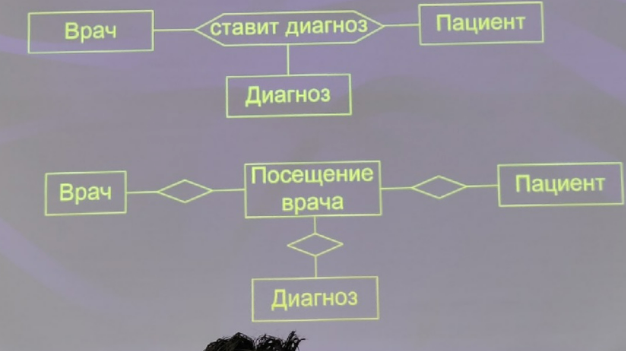


**Правило 4**. Степень рекурсивной связи 1:М (М:1) класс принадлежности М-связной сущности – необязательный.

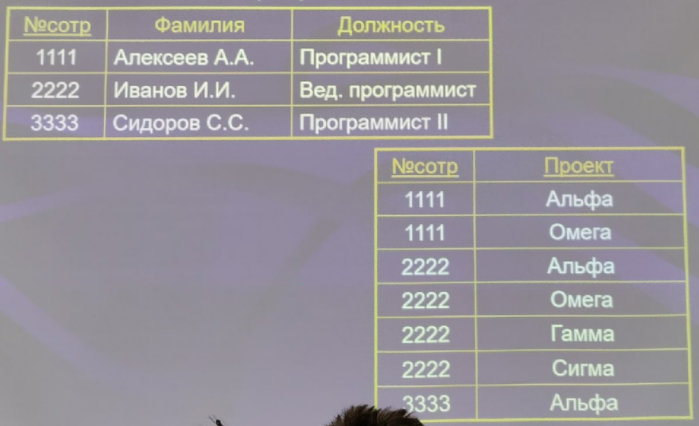
**Правило 5**. Степень рекурсивной связи М:М.

1. сложные типы связей

Сложную связь нужно декомпозировать путем выделения промежуточной сущности, промежуточную сущность связать со всеми исходными сущностями.



1. Многозначные атрибуты



Для каждого многозначного атрибута сущности создается отдельное отношение и это новое отношение представляет собой сочетание …

**Этап 3**. Проверка отношений с помощью правил нормализации.

Нормализация используется для улучшения модели, чтобы модель удовлетворяла разным ограничениям, позволяющим исключить нежелательное дублирование данных. Нормализация гарантирует, что полученная в результате модель будет наилучшим образом отображать особенность использования данных, не содержать противоречий, иметь минимальную избыточность.

Этапы процесса нормализации:

* приведение к 1НФ, позволяющее удалить из отношений повторяющиеся группы атрибутов;
* приведение ко 2НФ, позволяющее устранить частичную зависимость атрибутов от первичного ключа;
* приведение к 3НФ, позволяющее устранить транзитивную зависимость атрибутов от первичного ключа;
* приведение к нормальной форме Бойса-Кодда (НФБК), позволяющее удалить из функциональных зависимостей оставшиеся аномалии.

Целью выполнения всех 4 этапов – получение гарантии того, что каждое из отношений будет удовлетворять требованию НФБК. Если будут найдены отношения, неудовлетворяющие требованиям НФБК – это указывает на то, что при преобразовании логической модели произошла ошибка или часть логической модели неверна.

**Этап 4**. Определение требований поддержки целостности данных

Ограничение целостности данных вводится с целью предотвратить появление противоречивых данных. В современных СУБД функции контроля целостности могут присутствовать или отсутствовать.

5 видов ограничений целостности данных

* обязательные данные (некоторые атрибуты всегда должны содержать значения);
* ограничения для доменов атрибутов (например, для пола – М или Ж, стоимость – множество положительных значений);
* целостность сущность (первичный ключ не содержит пустого значения; если внешний ключ дочернего отношения содержит некоторое значение, то это же значение обязательно должно присутствовать в родительском отношении);
* ссылочная целостность ();

организация поддержки ссылочной целостности

Случай 1. Вставка новой строки в дочернее отношение (при добавлении новой строки необходимо убедиться, что значение внешнего ключа либо пусто, либо заполнено тем значением, что есть в первичном ключе родительского отношения);

Случай 2. Удаление строки из дочернего отношения;

Случай 3. Обновление внешнего ключа в строке дочернего отношения;

Случай 4. Вставка строки в родительское отношение;

Случай 5. Удаление строки из родительского отношения.

NO ACTION – если в дочернем отношении есть хотя бы одна ссылка на удаляемую строку, то удаление строки из родительского отношения запрещается.

CASCADE – удаляем родителей и детей.

SET NULL – удаляя строку из родительского отношения, во всех строках (дочернего отношения во внешний ключ), которые на нее ссылаются, записываем NULL.

NO CHECK – не предпринимается никаких действий.

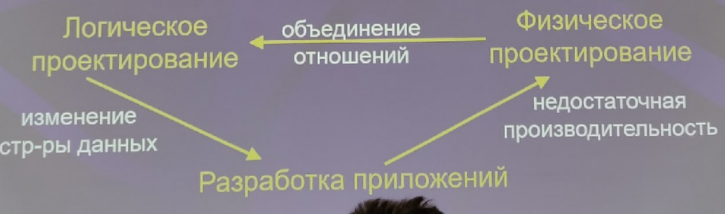
Случай 6. Обновление первичного ключа в строке родительского отношения.

* ограничения предметной области (анализируется бизнес-правило, потому что добавление или обновление сущностей должно проходить согласно ограничениям);

**Физическое проектирование баз данных**

Исходной информацией для этапа физического проектирования БД является логическая модель данных, состоящая из ER-диаграммы или диаграммы отношений, а также из реляционной схемы.

Обратная связь между физическим проектированием, логическим проектированием и разработкой приложений.



**Этап 1**. Проектирование основных отношений

Определение каждого отношения включает:

* имя отношения;
* список простых атрибутов;
* определение первичного ключа и альтернативных (АК) и внешних (FK) ключей;
* список производных атрибутов и списание способов их вычисления;
* определение требований ссылочной целостности для любых внешних ключей.

Для каждого атрибута следует указать:

* определение его домена, включающее тип данных, размерность внутреннего представления атрибута и любые требуемые ограничения на допустимые значения;
* значение атрибута, принимаемое по умолчанию;
* допустимость значения NULL

Конкретный способ реализации отношений зависит от выбранной СУБД.

**Этап 2**. Разработка способов получения производных данных

На этапе физического проектирования, необходимо определить должен ли производный атрибут храниться в БД или вычисляться каждый раз, когда есть необходимость.

Два варианта затрат:

* на хранение производных данных и поддержание их согласованности с реальными данными;
* на вычисление производных данных по мере необходимости.

**Этап 3**. Реализация ограничений предметной области

Способ реализации ограничений:

* с использованием возможностей стандарта языка SQL;
* с применением триггеров;
* непосредственно в самом приложении.

Показатели оценки эффективности хранения данных:

* производительность выполнения транзакций – количество транзакций, которые могут быть обработаны за определенный интервал времени;
* время ответа – время, необходимое для выполнения одной транзакции;
* дисковая память

Все эти показатели эффективности, которые находятся в распоряжении администратора БД, контролируются с помощью утилит, которые входят в состав СУБД.