Содержание

1 Основные понятия БД. Архитектура СБД **2**
1.1 База дынных (БД) 2
1.2 Система БД (СБД) 2
1.3 Основные составляющие СБД 2
1.4 Реляционные БД 2
1.5 Дореляционные и постреляционные БД 3
1.6 Три уровня архитектуры (внутренний, внешний, концептуальный) 5

2 Реляционная модель БД **7**
2.1 Таблица, столбец, строка, поле 7
2.2 Домены, отношения, атрибуты, кортежи 7

3 Целостность данных **11**
3.1 Потенциальные ключи, первичный ключ 11

1 Основные понятия БД. Архитектура СБД

1.1 База дынных (БД)

Определение:

База данных (БД) — набор постоянных данных, которые используются прикладными системами для какого-либо предприятия.

1.2 Система БД (СБД)

Определение:

Система баз данных (СБД) — это, по сути, не что иное, как компьютеризированная система хранения записей. Саму же базу данных можно рассматривать как подобие электронной картотеки, то есть хранилище для некоторого набора занесённых в компьютер файлов данных (где файл — абстрактный набор данных).

1.3 Основные составляющие СБД

1. данные;
2. аппаратное обоспечение;
3. программеное обеспечение;
4. пользователи.

или

1. СУБД;
2. приклодное программное обеспечение базы данных и операционной системы;
3. технические средства, обеспечивающие обслуживание пользователей.

1.4 Реляционные БД

Определение редяционной БД:

Реляционной базой данных называется набор отношений.

1.5 Дореляционные и постреляционные БД

Файлы и файловые системы:

Первые БД были созданы на основе файловых систем. Для каждой прикладной программы предоставлялся свой набор данных, оформленный в виде файла со своей структурой.

Такие системы имели ряд недостатков:

1. ФС не знает конкретной ситруктуры файла. Поэтому при малейших изменениях в структуре файлов приходилось менять программы, работающие с ними;

2. В файловой системе для каждого файла имеется информация о пользователе — владельце файла и определённые доступные для других пользователей. Из-за этого было очень трудно выстраивать ФС, так как часто файл созданные одной программой не доступен для другой;

3. Файл в файловой системе был недоступен для нескольких пользователей одновременно. Его можно было модифицировать одновременно только 1 человеку, что делало совместную работу невозможной.

Иерархические БД:

Иерархические БД

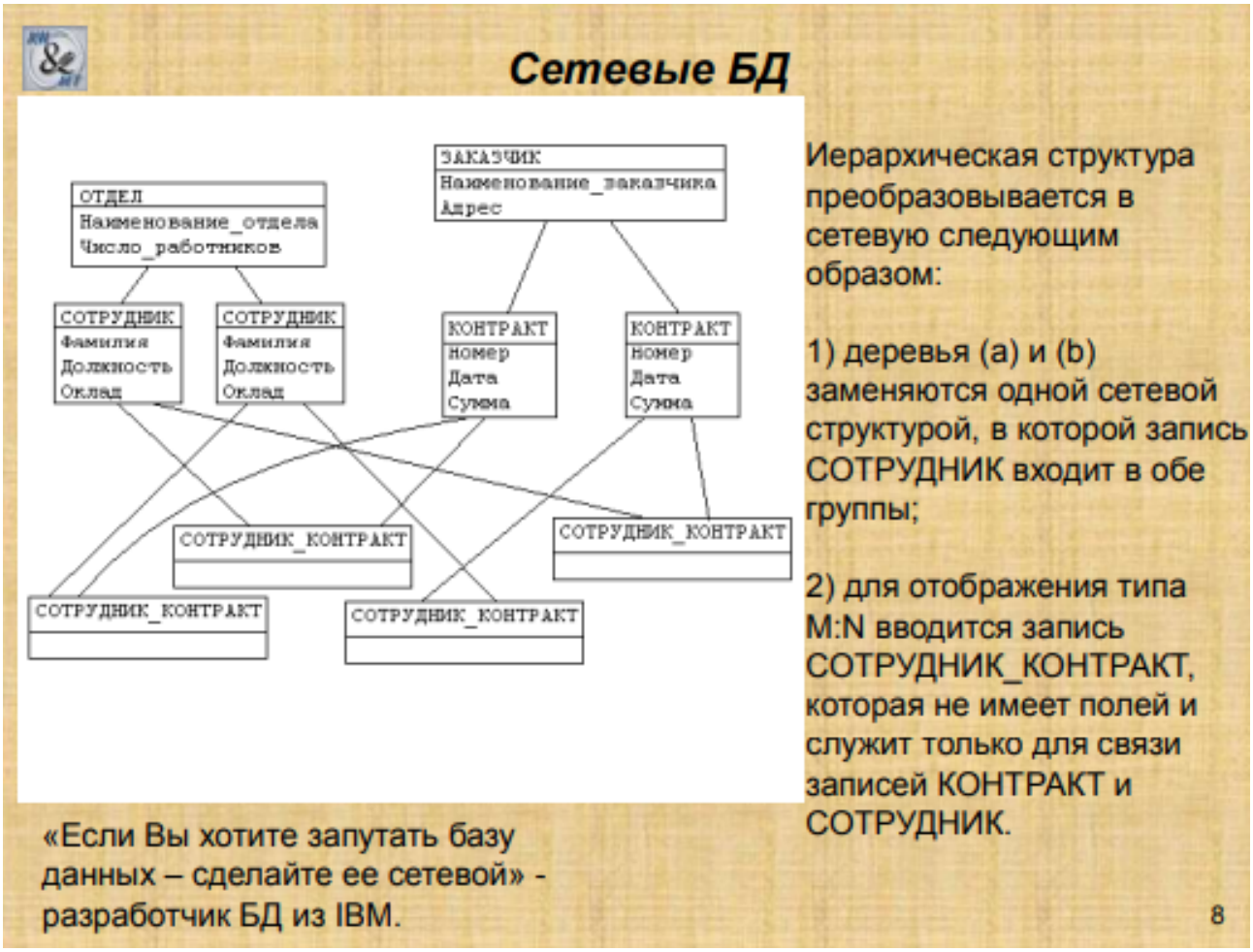
Рассмотрим следующую модель данных предприятия: предприятие состоит из отделов, в которых работают сотрудники. В каждом отделе может работать несколько сотрудников, но сотрудник не может работать более чем в одном отделе.

Поэтому, для информационной системы управления персоналом необходимо создать структуру, состоящую из родительской записи ОТДЕЛ (НАИМЕНОВАНИЕ_ОТДЕЛА, ЧИСЛО_РАБОТНИКОВ) и дочерней записи СОТРУДНИК (ФАМИЛИЯ, ДОЛЖНОСТЬ, ОКЛАД).

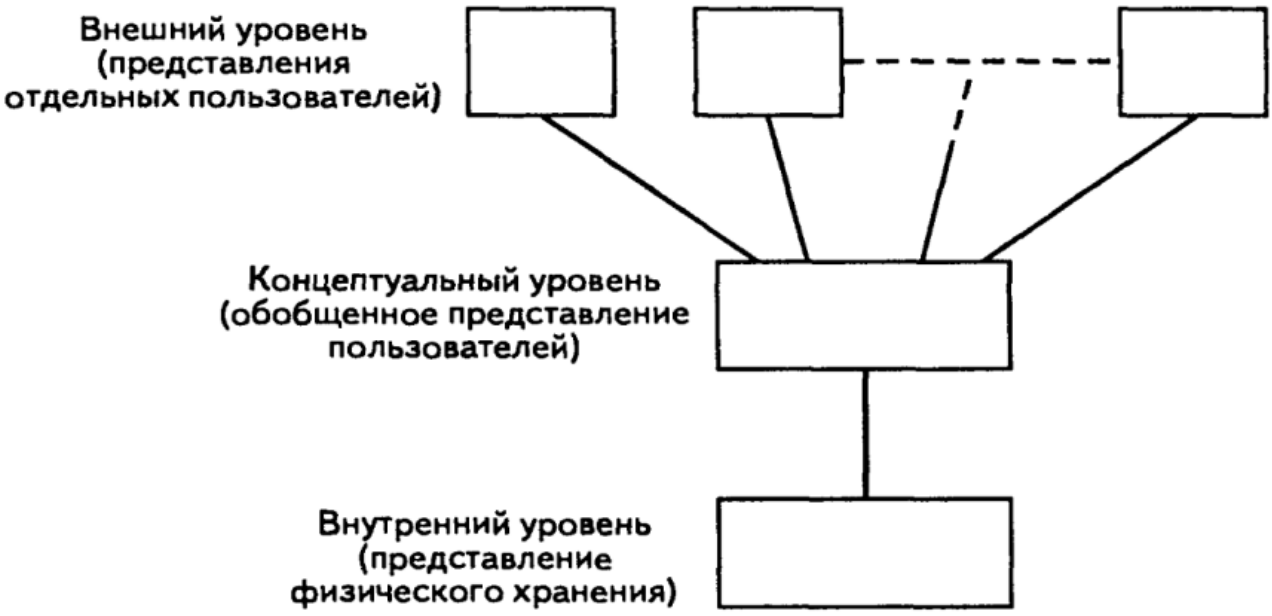
Для автоматизации учета контрактов с заказчиками необходимо создание еще одной иерархической структуры: заказчик - контракты с ним - сотрудники, задействованные в работе над контрактом. Это дерево будет включать записи ЗАКАЗЧИК(НАИМЕНОВАНИЕ_ЗАКАЗЧИКА, АДРЕС), КОНТРАКТ(НОМЕР, ДАТА,СУММА), ИСПОЛНИТЕЛЬ (ФАМИЛИЯ, ДОЛЖНОСТЬ, НАИМЕНОВАНИЕ_ОТДЕЛА).


ОТДЕЛ
Наименование_отдела
Число_работников
СОТРУДНИК
Фамилия
Должность
Оклад
СОТРУДНИК
Фамилия
Должность
Оклад
(a)
ЗАКАЗЧИК
Наименование_заказчика
Адрес
КОНТРАКТ
Номер
Дата
Сумма
КОНТРАКТ
Номер
Дата
Сумма
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
ИСПОЛНИТЕЛЬ
Фамилия
Должность
Наименование_отдела
КОНТРАКТ
Номер
Дата
Сумма
КОНТРАКТ
Номер
Дата
Сумма
(c)
(b)
Недостатки иерархических БД:
1. Частично дублируется информация между записями СОТРУДНИК и ИСПОЛНИТЕЛЬ, причем в иерархической модели данных не предусмотрена поддержка соответствия между парными записями.
2. Иерархическая модель реализует отношение между исходной и дочерней записью по схеме 1:N. Допустим, что исполнитель может принимать участие более чем в одном контракте (связь типа M:N). В этом случае в базу данных необходимо ввести еще одно групповое отношение, в котором ИСПОЛНИТЕЛЬ будет являться исходной записью, а КОНТРАКТ - дочерней (рис. (c)). Таким образом, опять дублируется информация.
7

Сетевые БД:



1.6 Три уровня архитектуры (внутренний, внешний, концептуальный)



Определение:

Внутренний уровень — уровень наиболее близкий к физическому хранению

данных, т.е. связанный с со способами сохранения информации на физических устройствах хранения. Внутренний уровень — собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации.

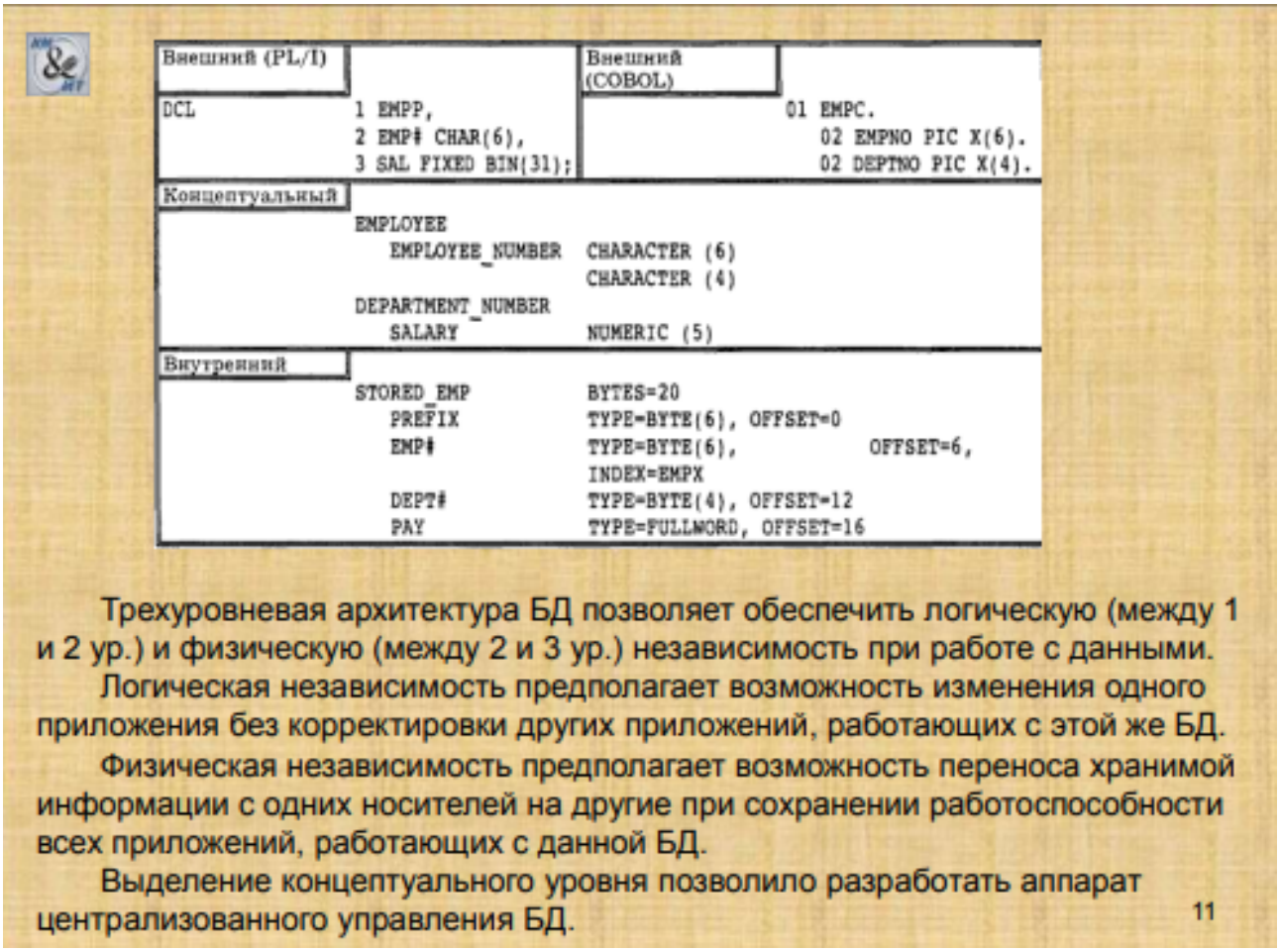
Определение:

Концептуальный уровень — уровень, на котором база данных представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной базой данных. Фактически концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой создавалась база данных. Как любая модель, концептуальная модель отражает только существенные, с точки зрения обработки, особенности объектов реального мира.

Определение:

Внешний уровень — самый верхний уровень, где каждое представление имеет свое "видение"данных. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, и наоборот, именно эти сведения используются в подсистеме отдела кадров.

Размышления:



2 Реляционная модель БД

2.1 Таблица, столбец, строка, поле

2.2 Домены, отношения, атрибуты, кортежи

Определение домена:

Домен — допустимое потенциальное множество значений для отдельно взятого типа.

Дополнительная информация:

1.2. Домен

Понятие домена более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, в число значений домена "Имена« могут входить только те строки, которые могут изображать имя.

Данные считаются сравнимыми только в том случае, когда они относятся к одному домену.

Домены более точно можно определить как именованное множество скалярных значений одного типа. Эти скалярные значения называют ***скалярами***. По сути, это наименьшая семантическая (смысловая) единица данных. У скаляров нет внутренней структуры, т.е. они не разложимы в данной реляционной модели.

Например, если имеется атрибут (свойство объекта) «ФИО», он предусматривает скаляры, содержащие фамилию, имя и отчество. Конечно, эти скаляры можно еще разбить на буквы, но тогда будет утрачен нужный смысл. То есть для данной модели наименьшими семантическими единицами данных будут именно фамилия, имя и отчество.

Из доменов, как уже говорилось, берутся значения атрибутов. *На практике* домены часто не описывают, а задают типом, форматом и другими свойствами данных. Каждый атрибут должен быть определен на ***единственном домене.***

Основное назначение доменов — ***ограничение сравнения*** различных по смыслу атрибутов.

Например: Если для атрибутов №ЗачетнойКнижки отношения Студенты и №Кабинета для отношения Кабинеты домены заданы следующим образом:

№ зачетной книжки = {100000, 100001, 100002, … 999999}

№ кабинета = {1, 2, 3, … 999},

то система выдаст ошибку на запрос типа: «Вывести всех студентов, № зачетной книжки которых совпадает с № кабинета». Если же домены не определены, а определен только целый тип данных для атрибутов №ЗачетнойКнижки и №Кабинета, то подобный запрос выполнится, хотя не будет иметь смысла.

Еще одно возможное применение доменов – использование их в специальных ***запросах***. Например, «Какие отношения в БД включают атрибуты, определенные на домене «№ зачетной книжки»?». В системе, поддерживающей домены, такой запрос будет иметь смысл и результатом его будет список отношений, где используется № зачетной книжки (это могут быть отношения Студенты,Занятия,Успеваемость, …). А в системе, где домены не определены, реализовать такого рода запрос гораздо сложнее – если через имена атрибутов, то они могут не совпадать (имена атрибутов, содержащих № зачетной книжки могут варьироваться: № зачетки, № зачетной книжки и т.п.), а если через тип – то получится много лишних отношений, т.к. немало атрибутов может иметь целый тип данных.

Определение отношения, атрибута и кортежа:

Еще одно возможное применение доменов – использование их в специальных ***запросах***. Например, «Какие отношения в БД включают атрибуты, определенные на домене «№ зачетной книжки»?». В системе, поддерживающей домены, такой запрос будет иметь смысл и результатом его будет список отношений, где используется № зачетной книжки (это могут быть отношения Студенты,Занятия,Успеваемость, ...). А в системе, где домены не определены, реализовать такого рода запрос гораздо сложнее – если через имена атрибутов, то они могут не совпадать (имена атрибутов, содержащих № зачетной книжки могут варьироваться: № зачетки, № зачетной книжки и т.п.), а если через тип – то получится много лишних отношений, т.к. немало атрибутов может иметь целый тип данных.

Что такое отношение в реляционной модели

У отношения нет физического представления. Реляционная база данных — это попытка отразить реляционную модель, а не ее точная копия.

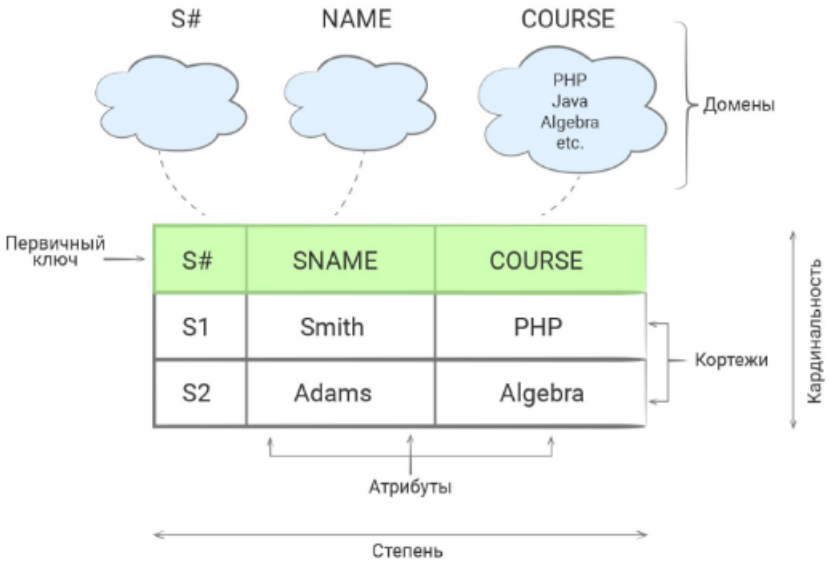
Реляционная модель опирается на раздел математики теорию множеств, которую нужно знать хотя бы немного. Множество — это совокупность произвольных элементов, которые объединены по некоторому признаку. Например, множество натуральных чисел или множество учеников одного класса. Изучить этот раздел подробнее вы можете в нашем курсе «Теория множеств».

Из любого множества можно выделить **подмножество** — множество элементов, все элементы которого входят в исходное множество. Подмножество — это часть множества. Например, множество натуральных чисел — это подмножество по отношению к множеству целых чисел, которое в свою очередь — подмножество рациональных чисел. Из этого следует, что натуральные числа — это подмножество рациональных чисел.

Еще одно важное понятие — **кортеж**. Это упорядоченный набор данных фиксированной длины. Элементами кортежа может быть все что угодно. Это математический способ представить некоторый набор связанных данных.

На основе этих понятий можно сформулировать более понятное определение отношения. Отношение — это множество кортежей, которые называются **телом отношения**. В нем каждый кортеж соответствует схеме.

Схема — это заголовок отношения. Она описывает общую структуру кортежей, количество элементов внутри них и их типы. Каждый такой элемент называется **атрибутом**:



Отношение визуально похоже на таблицу, но у него есть существенные отличия:

- Нет двух одинаковых элементов (кортежей)
- Порядок кортежей не определен
- Порядок атрибутов в заголовке не определен

По этой причине таблица непригодна для точного описания отношения. В любой таблице порядок столбцов (атрибуты схемы) и строк (кортеж) строго определен. При этом на практике мы оперируем таблицами и не можем игнорировать их.

По Батраевой:

1.3. Кортеж, отношение

Фундаментальным понятием реляционной модели данных является понятие ***отношения***. В определении понятия отношения будем следовать книге К. Дейта [11].

Определение 1. ***Атрибут отношения*** *есть пара вида* *<Имя_атрибута : Имя_домена>.*

Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

Определение 2. ***Отношение*** *R*, определенное на множестве доменов *D1, … Dn* (не обязательно различных), содержит две части: заголовок и тело.

Заголовок отношения содержит фиксированное количество атрибутов отношения:

(<A1:D1>,…, <An:Dn>)

Тело отношения содержит множество кортежей отношения. Каждый ***кортеж отношения*** представляет собой множество пар вида <Имя_атрибута : Значение_атрибута>:

(<A1:Val1>,…, <An:Valn>)

таких что значение *Vali* атрибута *Ai* принадлежит домену *Di*

Отношение обычно записывается в виде:

R(<A1:D1>,…, <An:Dn>) , или короче R*(A1,…,An)*, или просто *R*.

3 Целостность данных

3.1 Потенциальные ключи, первичный ключ