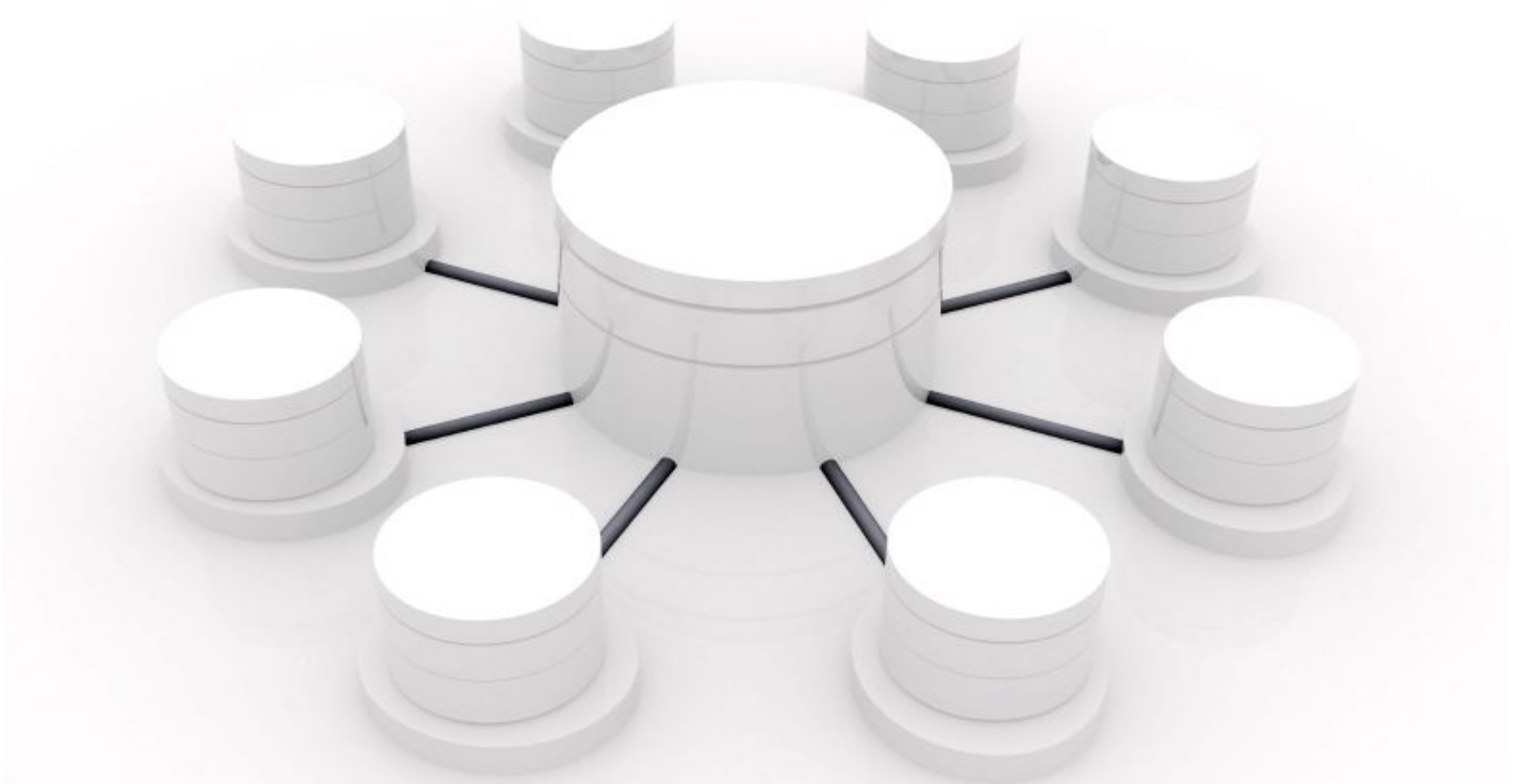
БАЗЫ ДАННЫХ



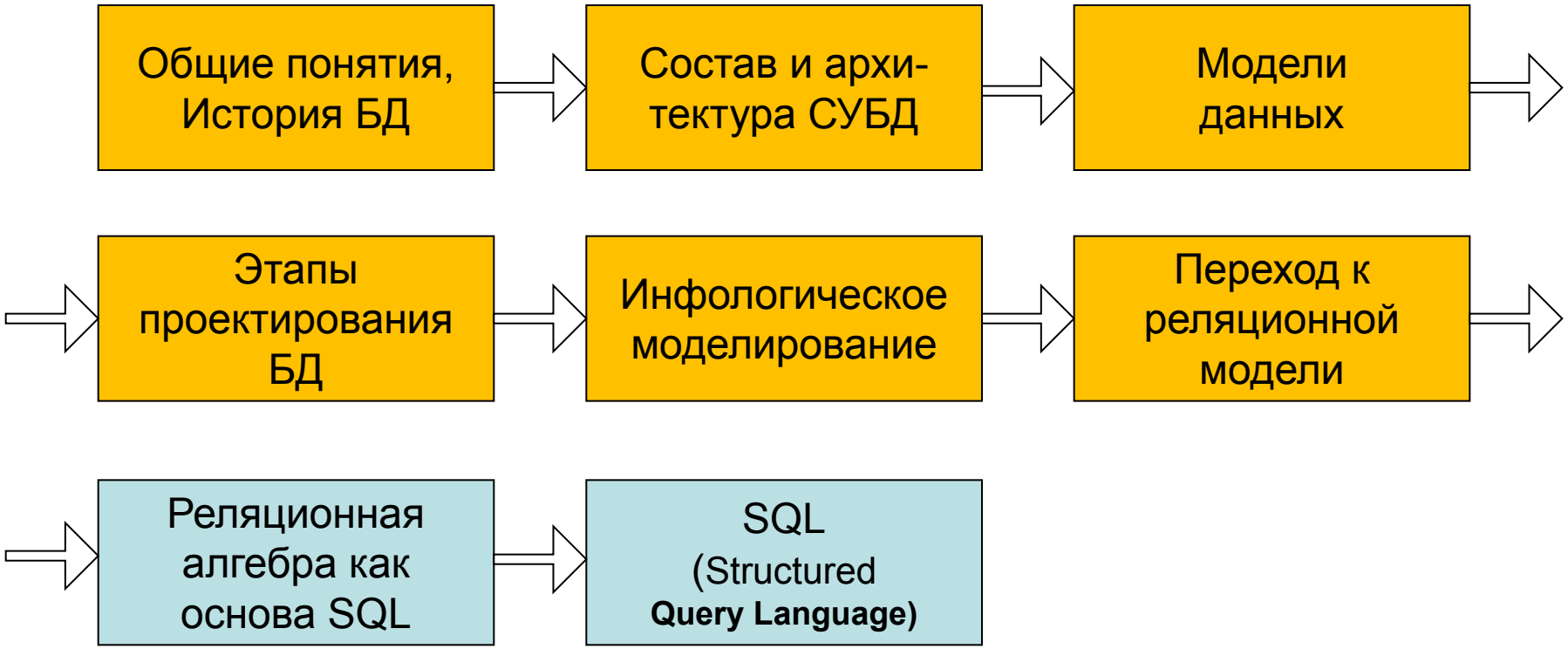
Лектор: **Кубил Виктор Николаевич**

E-mail: vksend@gmail.com

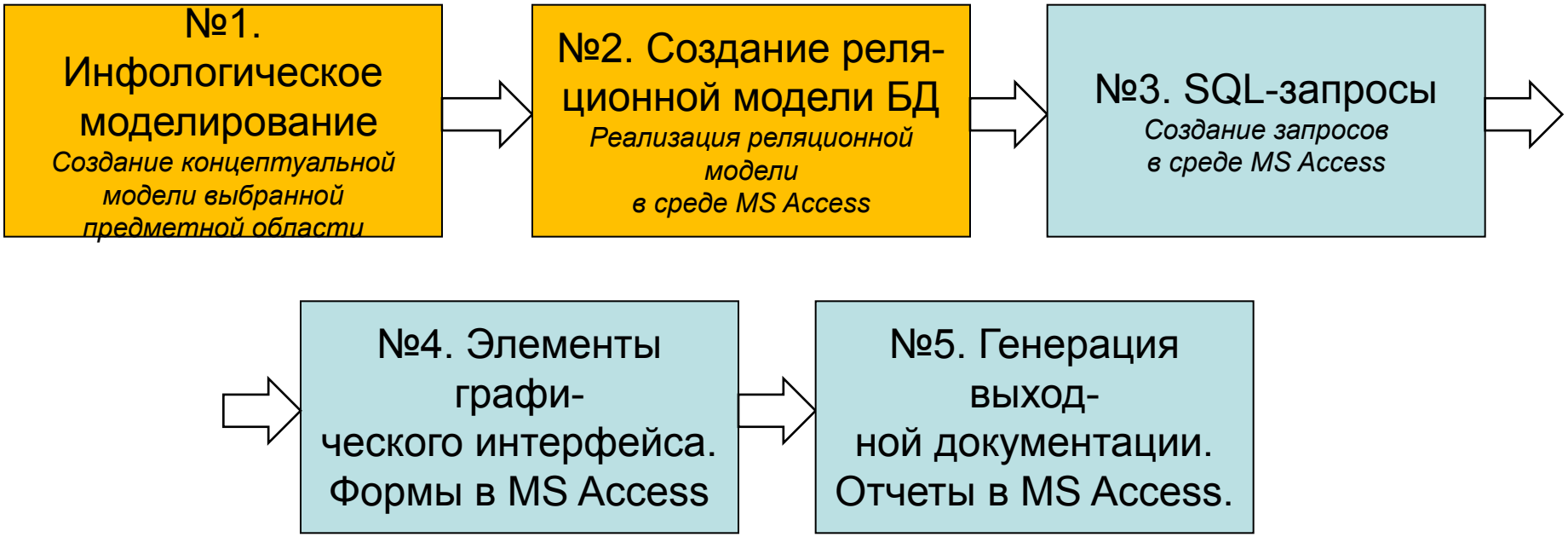
Из учебного плана

- Лекции
- Лабораторные и практические занятия
- Экзамен
- Курсовая работа

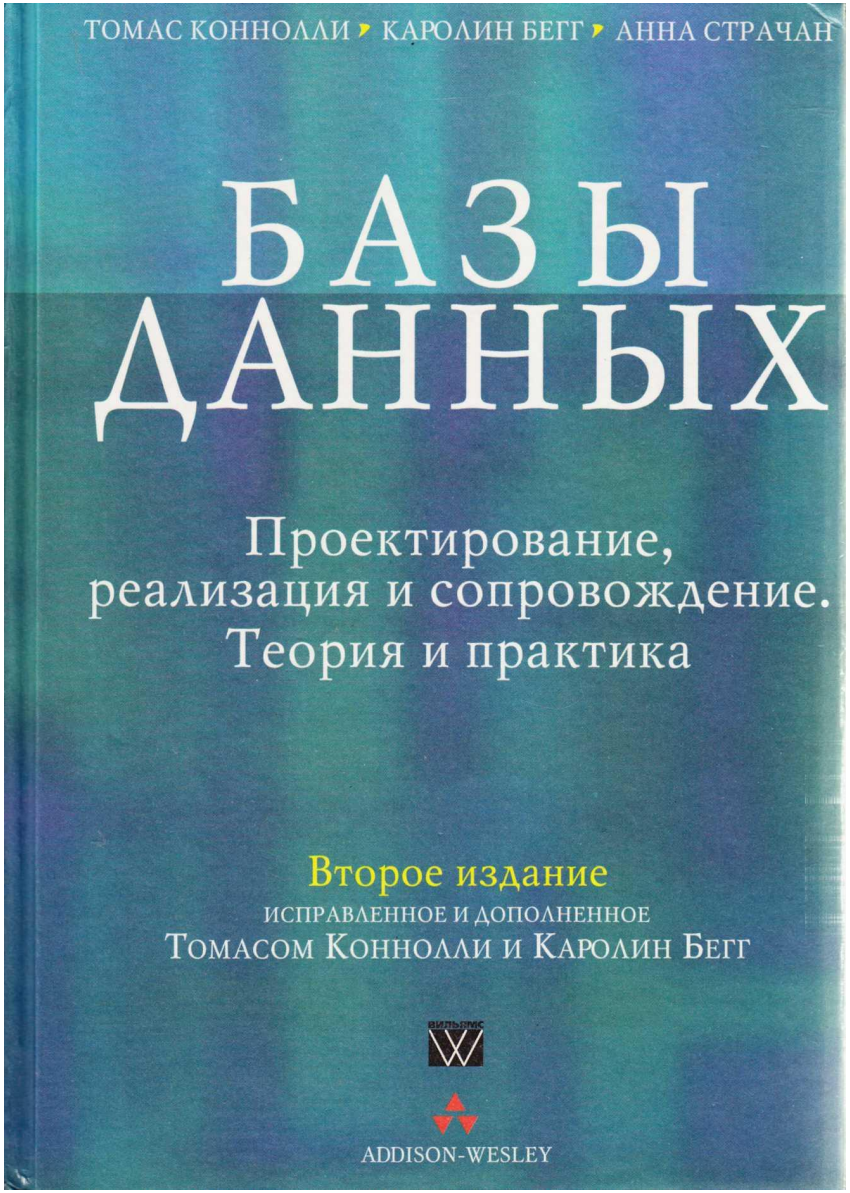
План лекционного курса БД



План лабораторных работ



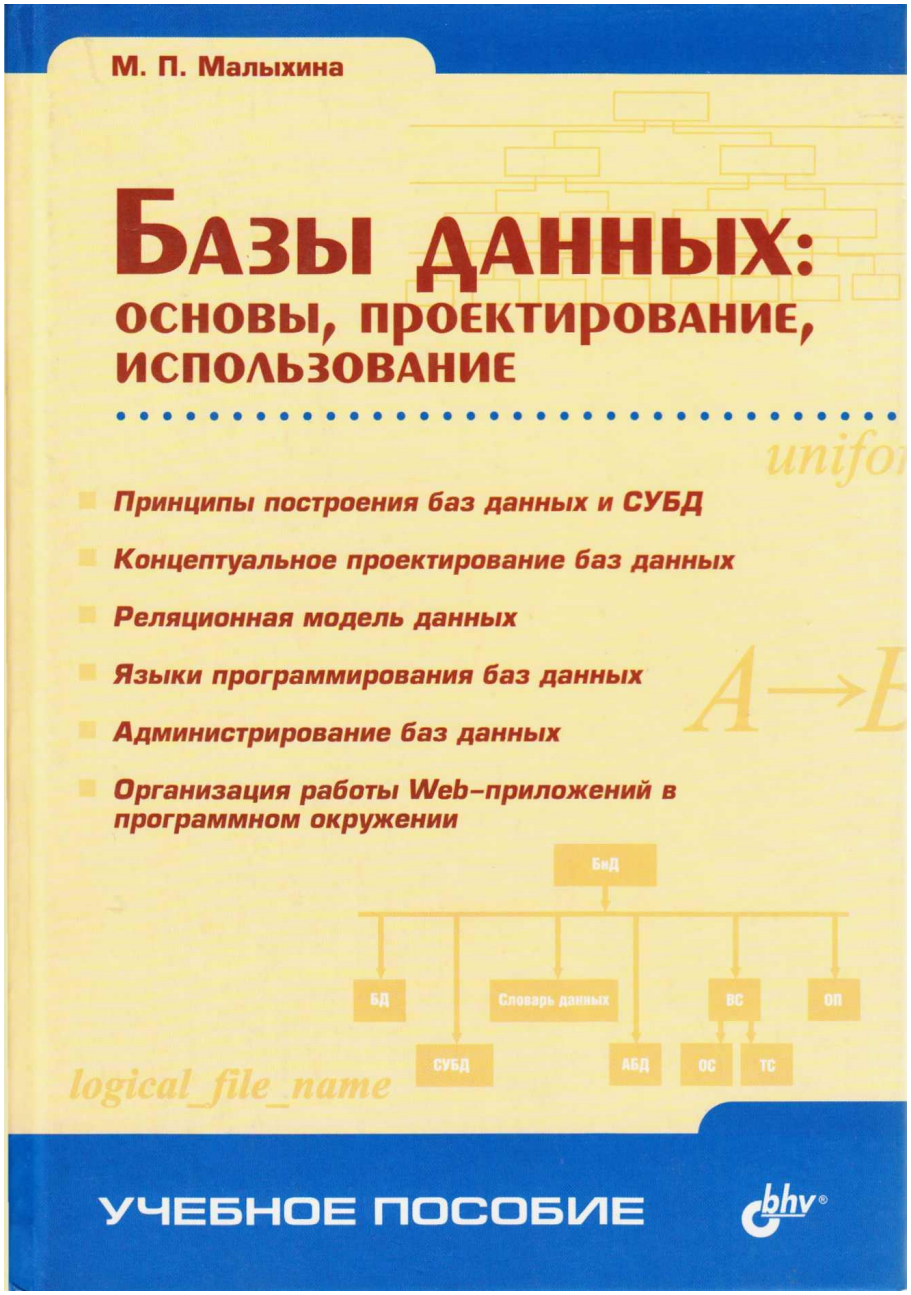
- **К. Дж. Дейт.**
- **Введение в системы баз данных (An Introduction to Database Systems)**
- 8-е издание
- Вильямс, 2006 г. Твердый переплет, 1328 стр.



- **Каролин Бегг, Томас Коннолли.**
- **Базы данных. Проектирование, реализация и сопровождение. Теория и практика.**

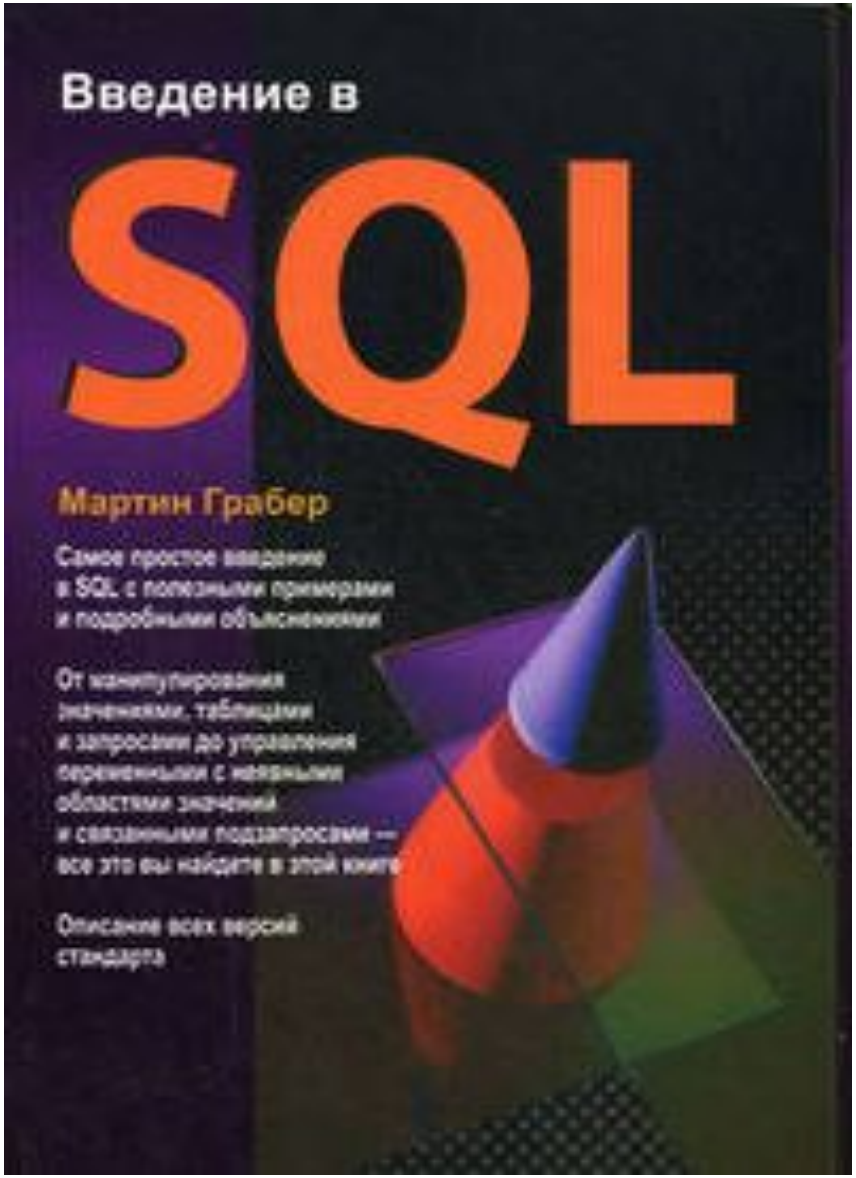
Третье издание.
1436 стр.,
2003 г.
Издательство:
Вильямс.

- **Карпова Т.С.**
- **Базы данных: модели, разработка, реализация.** –СПб.: Питер,2001.-304с.



- **Малыхина М.П. Базы данных: Основы, проектирование, использование:** БХВ-Петербург, 2004.– 512 с.

- **Хомоненко А.Д.**
- **Цыганков В.М.**
- **Мальцев М.С.**
- **Базы данных:** Учебник для высших учебных заведений/ Под ред.проф.А.Д. Хомоненко – 4-е изд.,доп. и перераб. – СПб.:КОРОНА принт,2004.-736 с.



- **Мартин Граббер**
- **Введение в SQL**
- Издательство: Лори, Пер. с англ., 2008.- 375 с. .

- **Когаловский М.Р. Энциклопедия технологий баз данных: Эволюция технологий. Технологии и стандарты. Инфраструктура. Терминология. -** М.: Финансы и статистика, 2005. - 800 с.


Г.А. Черноморов
БАЗЫ ДАННЫХ
в среде промышленных СУБД
ВЫСШЕЕ ОБРАЗОВАНИЕ
Новочеркасск 2006

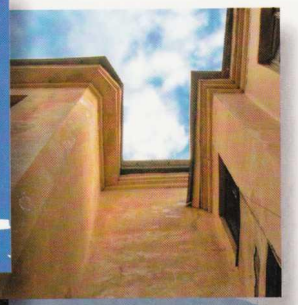

СИСТЕМЫ БАЗ ДАННЫХ
Теория и практика использования в Internet и среде Java
ВИЛЬЯМС
ADDISON WESLEY
Грег Риккарди

ВЛАДИСЛАВ ПИРОГОВ

bhv®

SQL Server 2005

ПРОГРАММИРОВАНИЕ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ



- Основы проектирования реляционных баз данных
- Описание языка Transact-SQL
- Программные объекты на стороне сервера
- Использование .NET-технологий
- Программирование на стороне клиента

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

Access 2007

НЕДОСТАЮЩЕЕ РУКОВОДСТВО

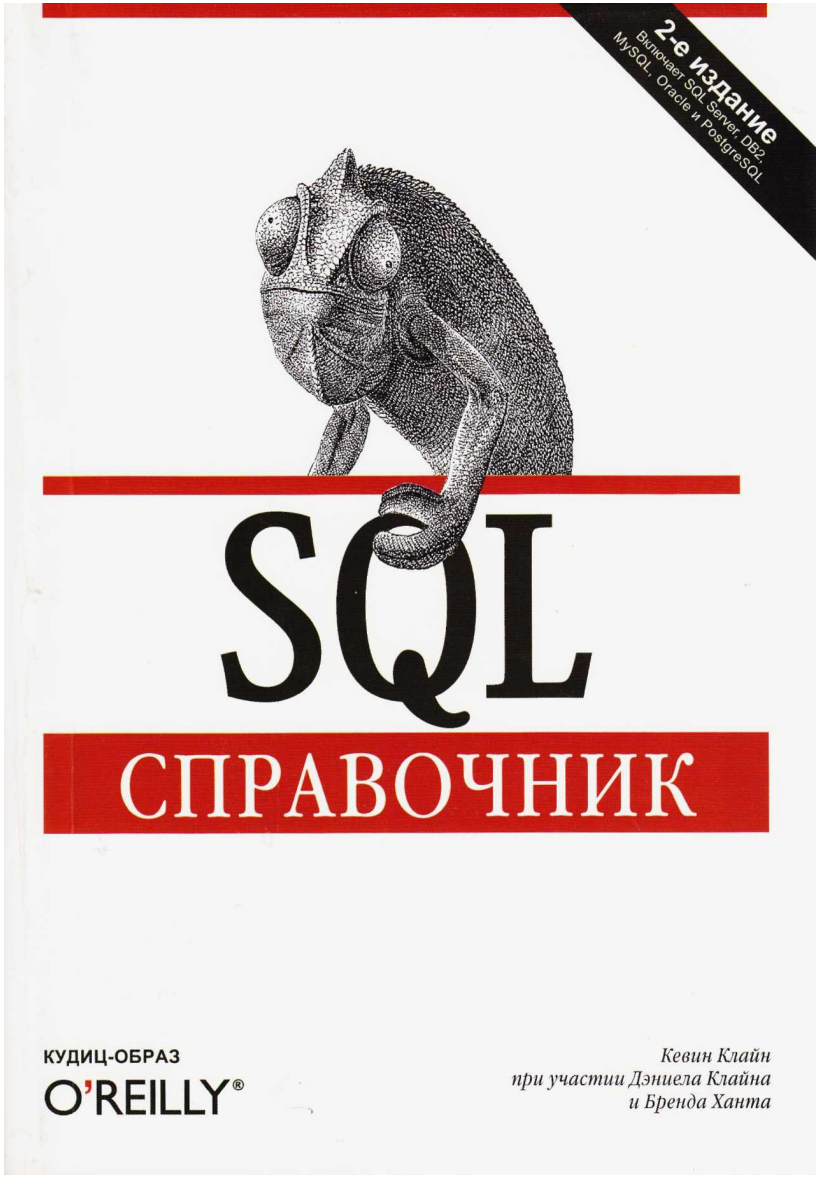
Книга, которая должна быть на полке

«Серия «Недостающее руководство» – просто самая разумная и полезная серия руководств».
Кевин Келли, соучредитель журнала «Wired»

POGUE PRESS™
O'REILLY®

Мэтью Мак-Дональд


В. Кирстен, М. Иринге
Б. Рёриг, П. Шульте
СУБД Caché
объектно-ориентированная разработка приложений
учебный курс
ПРОГРАММИРОВАНИЕ
CD-ROM прилагается
разработчикам корпоративных приложений
студентам
Springer
ПИТЕР


2-е издание
Включает SQL Server, DB2, MySQL, Oracle и PostgreSQL
SQL
СПРАВОЧНИК
КУДИЦ-ОБРАЗ
O'REILLY®
Кевин Клайн
при участии Дэниела Клайна
и Бренда Ханта

Две основных области использования вычислительной техники

В истории вычислительной техники традиционно выделяют две основных области использования:

1. **Применение для численных расчетов** (дало развитие алгоритмизации, ПО для математических расчетов, т.е. структура данных простая, алгоритмы обработки сложные).
2. **Накопление и обработка информации** (использование средств вычислительной техники в автоматических или автоматизированных информационных системах) - данные сложные и их много, алгоритмы обработки простые (относительно).

Фундаментальные понятия теории баз данных

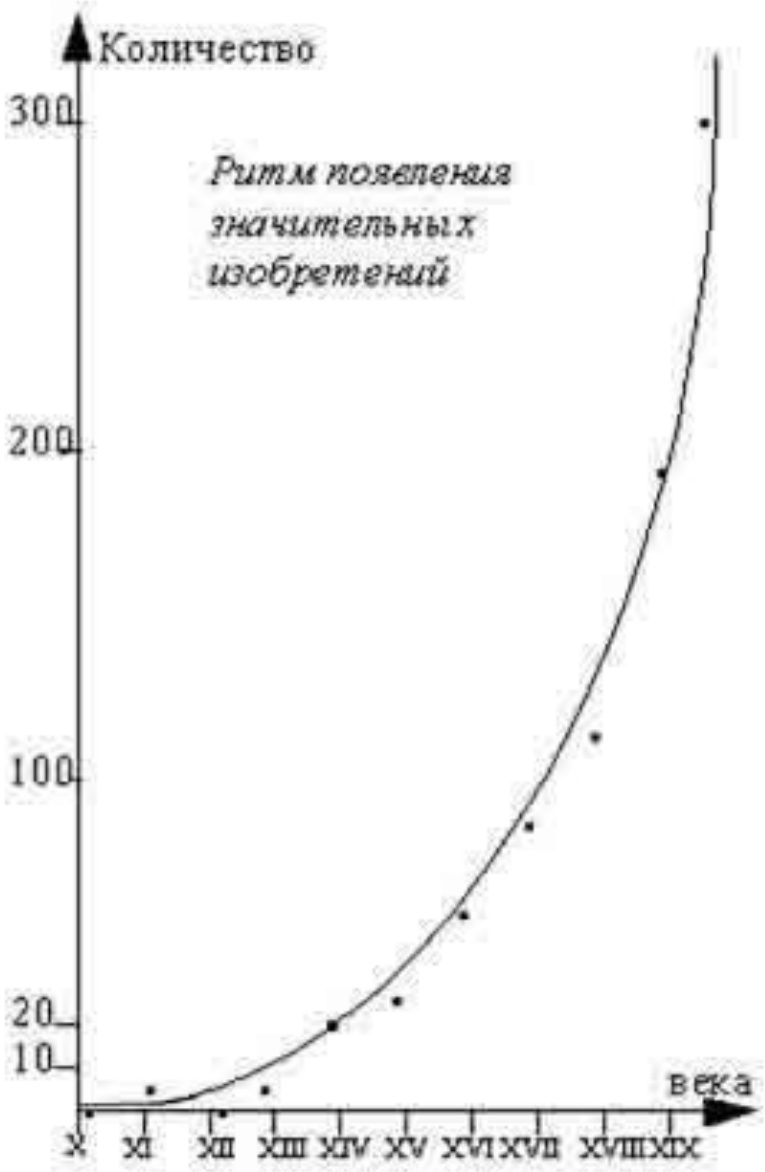
- **Информация** – это сведения об окружающем мире, имеющие специфическую интерпретацию или смысл.
- **Данные** (от лат. *data*) – это представление фактов и идей в формализованном виде, пригодном для передачи и обработки в некотором информационном процессе.
- **База данных (БД)** – организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для решения информационных потребностей пользователей.

Фундаментальные понятия теории баз данных

- **Информационная система (ИС) – это комплекс вычислительного и коммуникационного оборудования, программного обеспечения, лингвистических средств и информационных ресурсов,** который обеспечивает их сбор, хранение, актуализацию, распространение и обработку в целях поддержки какого-либо вида деятельности.
- **Система управления базами данных (СУБД) – это комплекс программ, позволяющих создавать и использовать БД,** обеспечивая безопасность, на дёжность хранения и целостность данных, а также предоставляя возможности администрирования.

Интересные факты

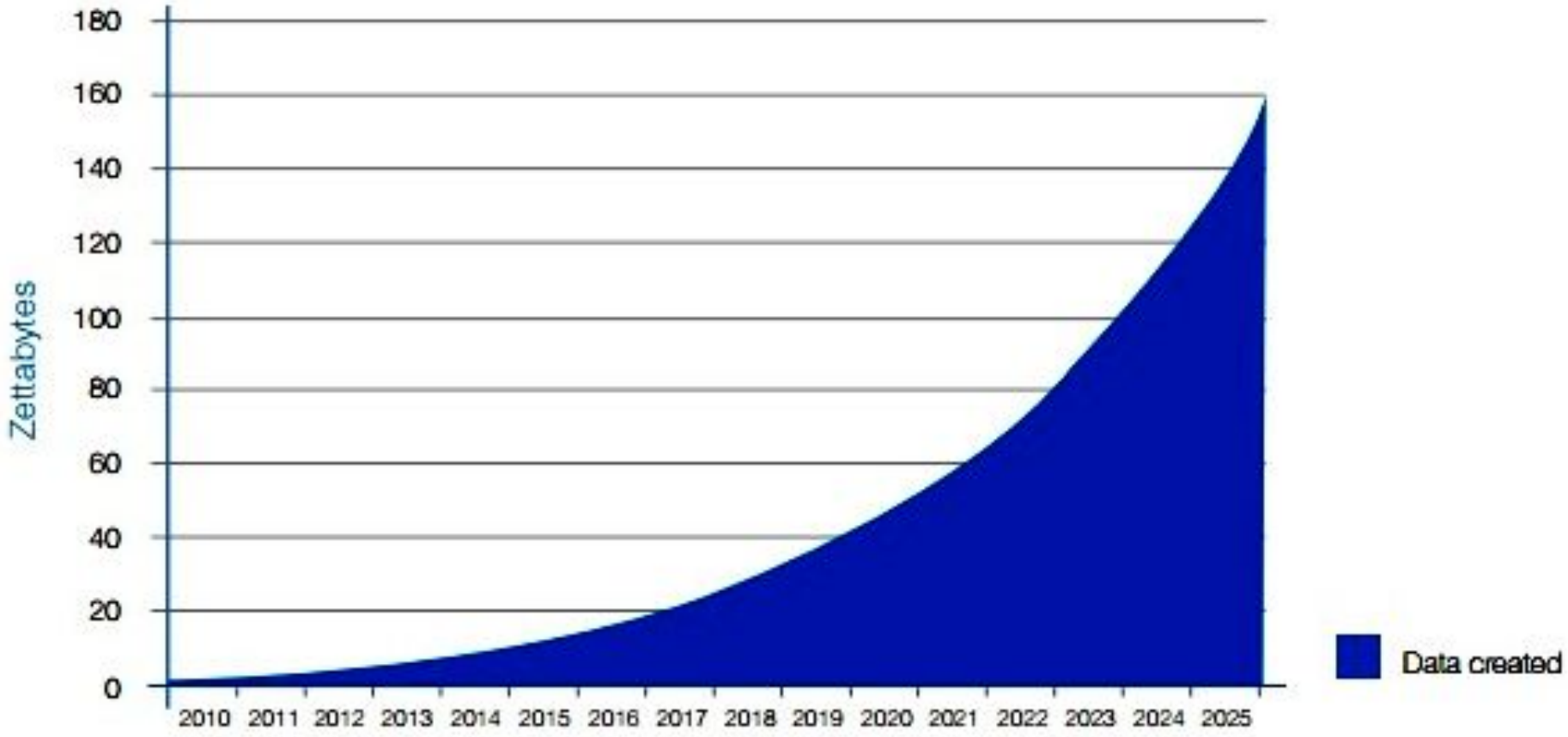
- Реляционная база данных Yahoo > 2 петабайт;
- Архив с базой данных OpenStreetMap занимает порядка 70Гб;
- Практически ни одно приложение не обходится без БД;
- Практически невозможно найти готовые решения;
- Установочные .msi файлы для Windows – это БД;
- В большинстве вакансий разработчика упоминается SQL, по данным hh.ru



Курс «Базы данных» посвящен решению проблемы рациональной организации информации

График Мэмфорда, отражающий экспоненциальный рост числа значительных изобретений во времени

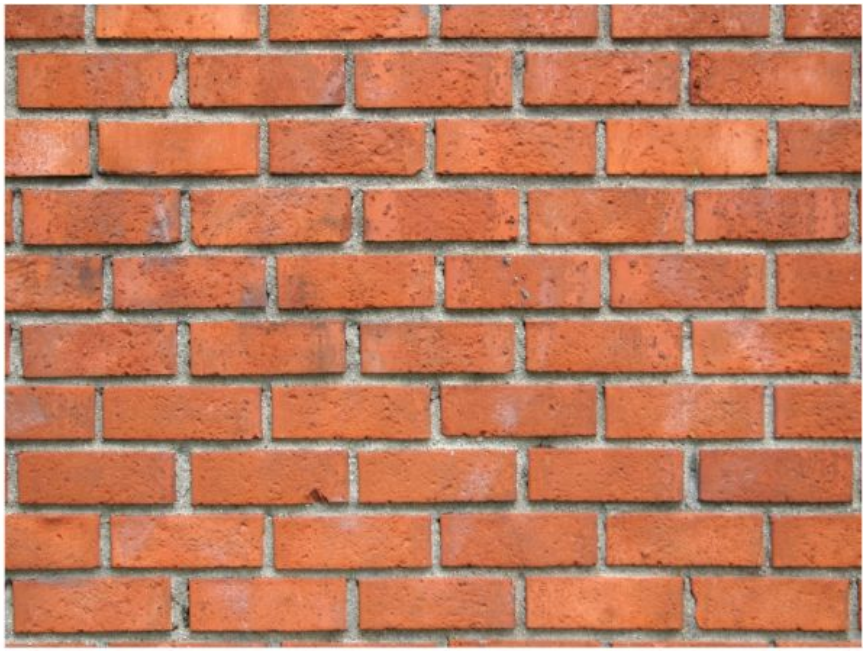
Рост объема генерируемых данных



Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017

Информация

База данных



Структурирование

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование – это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные в произвольной форме, например, в текстовом файле.

Рассмотрим пример…

Неструктурированные данные

«Для того чтобы добраться до села Дудкино, нужно сначала долететь на самолете до Иванова. Далее - на электричке до Орехова. Затем пересесть на паром и переправиться через реку Слоновую в поселок Ольховка, а оттудауже на попутной машине ехать в село Дудкино».

Первый способ структурирования: списочный

Как ехать в село Дудкино?

1. До Иванова на самолете.
2. Далее до Орехова на электричке.
3. До поселка Ольховка на пароме.
4. До села Дудкино на попутной машине.

Второй способ структурирования: табличный

Откуда	Куда	Транспорт
Москва	Иваново	самолет
Иваново	Орехово	электричка
Орехово	Ольховка	паром
Ольховка	Дудкино	попутная машина

Третий способ структурирования: графический

Выводы

Все четыре информационных сообщения несут одинаковые сведения, но отличаются форме представления и по форме восприятия информации.

В исходной форме представления нужную информацию добыть сложнее всего.

В списке сообщение разбивается на смысловые блоки.

В таблице данные распределяются по однотипным столбцам, хорошо организуются в памяти компьютера.

Диаграмма наиболее удобна для восприятия, но мало пригодна для обработки.

Рассмотрим более подробно структуризацию в виде таблицы на еще одном примере...

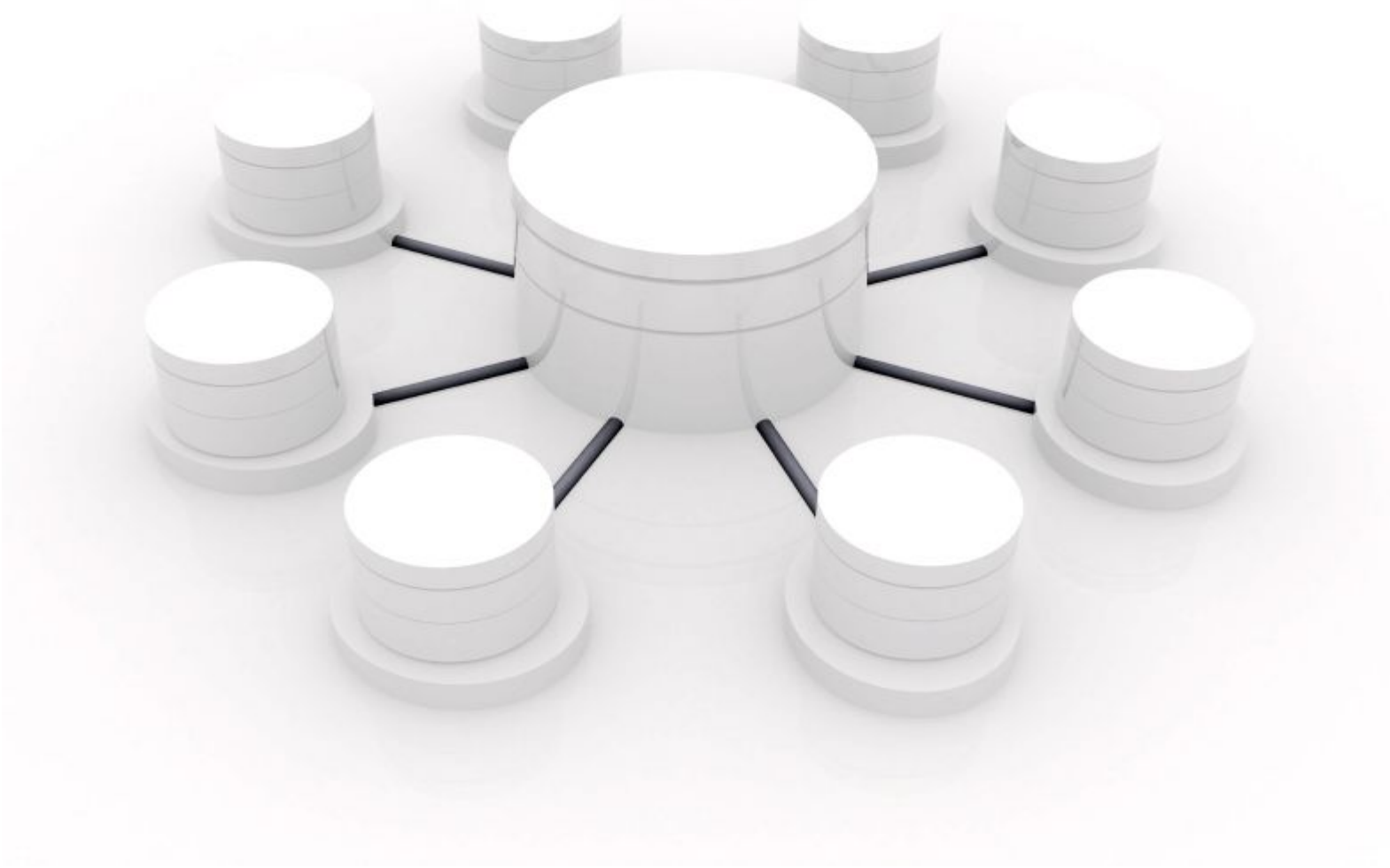
Пример: неструктурированные данные

На олимпиаду отправлены: студентка НПК 2-5а Кравцова Т.А., зачисленная 31 августа 2020 года, со средней успеваемостью 4,4 балла, krav_tanya@mail.ruНа олимпиаду отправлены: студентка НПК 2-5а Кравцова Т.А., зачисленная 31 августа 2020 года, со средней успеваемостью 4,4 балла, krav_tanya@mail.ru; староста ФИТУ 4-6 Егоров А.В., круглый отличник, a.v.egorov@list.ru и его одногруппница Абрамян М.Э. с успеваемостью 4,7 балла; староста ФГГНГД 3-2б Жуков С.Г., зачисленный 31.08.18, с успеваемостью 4,1 балла, zhook716@yandex.ru; и т.д.

Пример: структурированные данные

Факультет	Группа	Студент	Ста-роста	Зачис-ление	E-mail	Ср. балл
НПК	2-5а	Кравцова Т.А.		31.08.20	krav_tanya @mail.ru	4,4
ФИТУ	4-6	Егоров А.В.	✔		a.v.egorov @list.ru	5,0
ФИТУ	4-6	Абоян М.Э.				4,7
ФГГНГД	3-2б	Жуков С.Г.	✔	31.08.19	zhook716 @yandex.ru	4,1

История и этапы эволюции концепций БД. Функции СУБД. Поколения СУБД.



Основные этапы эволюции концепций БД

Этап 1. Простые файлы данных (начало 60-х годов).

Этап 2. Совершенствование методов доступа к файлу (конец 60-х годов)

Этап 3. Появление первых систем управления БД (начало 70-х г.) .

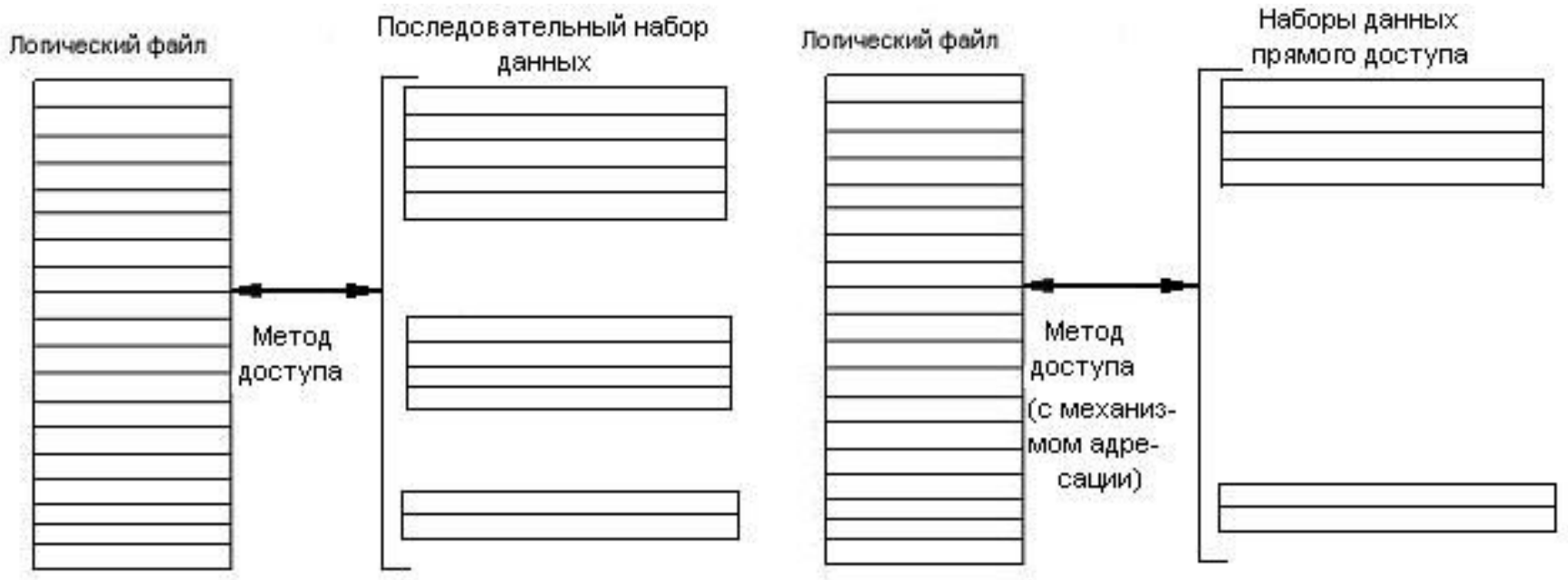
Этап 4. Современные системы баз данных

1 этап Простые файлы данных

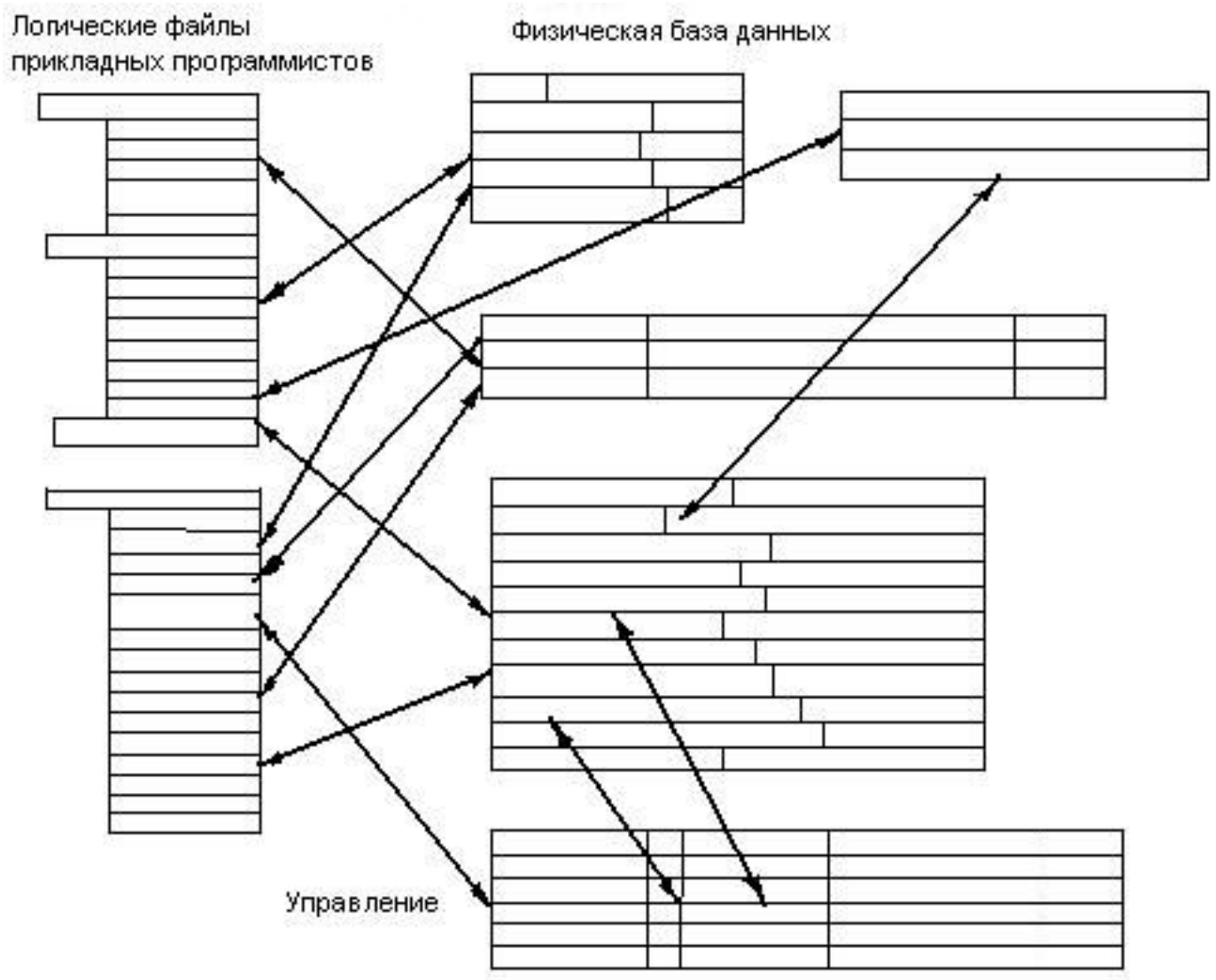
Проблемы традиционных систем управления файлами

- структура записи файла была известна только программе, которая с ним работала;
- при изменении структуры файла требовалось изменять структуру программы (или множества программ);
- отсутствие централизованных методов управления доступом к информации
- неэффективная параллельная работа многих пользователей с одними и теми же файлами;
- сложность восстановления.

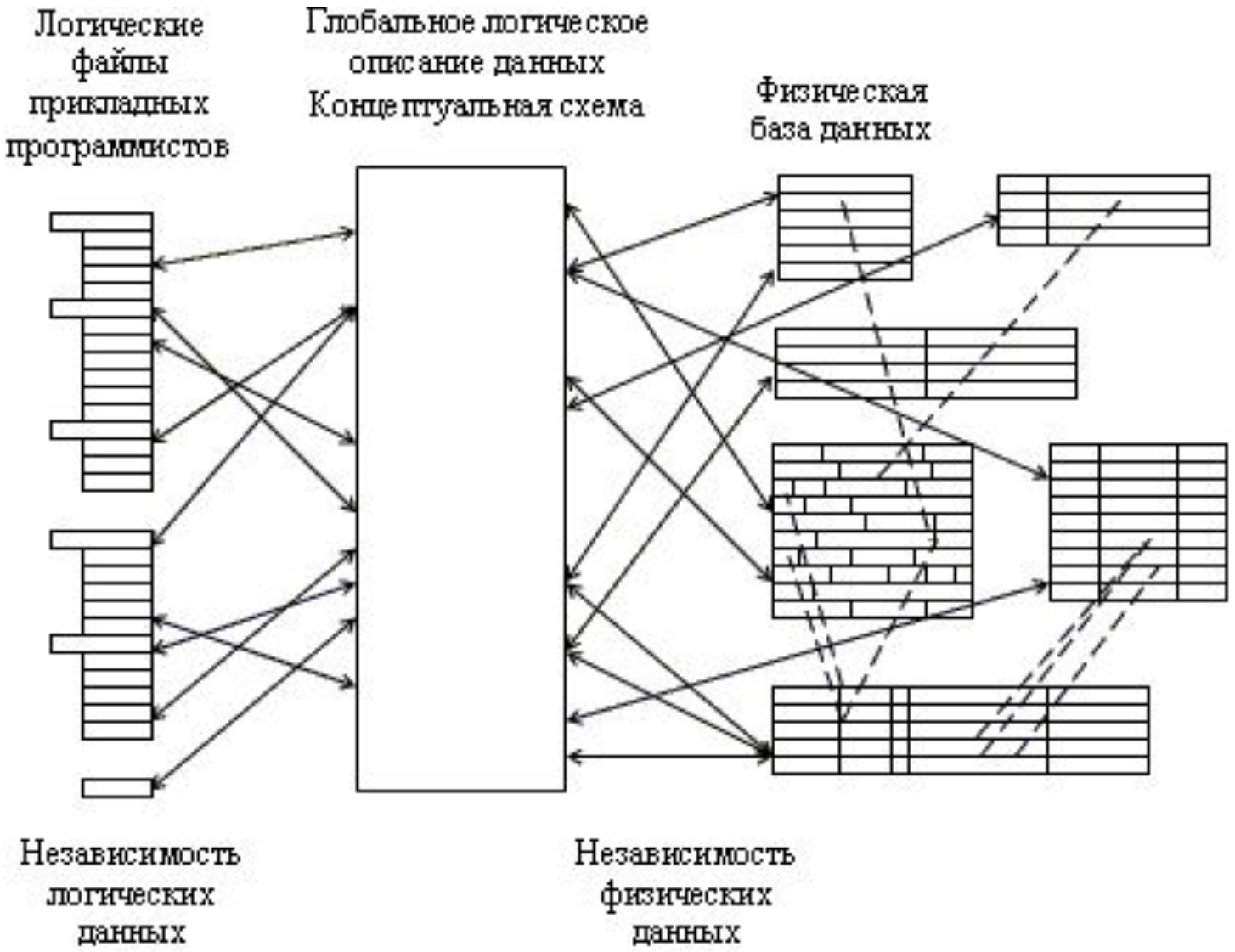
2 этап Совершенствование методов доступа к файлу



3 этап Появление первых СУБД



4 этап Современные СУБД



Основные функции СУБД

- 1. Обеспечивает пользователю возможность создавать новые БД и определять их схему (логическую структуру данных) с помощью специального языка, названного языком определения данных(DDL).
- 2. Позволяет "запрашивать" данные и изменять их с помощью подходящего языка. Этот язык называется языком запросов или языком манипулирования данными(DML).
- 3. Управляет данными во внешней и оперативной памяти, защищая их от случайной порчи или неавторизованного использования.
- 4. Контролирует доступ к данным сразу множества пользователей, не позволяя запросу одного из них влиять на запрос другого, и не допуская одновременного доступа, который может испортить данные.
- 5. Журнализация изменений и восстановление БД.

Краткая история развития БД

До 1963 года – Обработка файлов

	Описание наиболее значимых событий
1959	Образована Ассоциация по языкам систем обработки данных (***Conference on Data System Languages***, **CODASYL**) – организация, являющаяся автором языка программирования COBOL, языковых спецификаций сетевой модели данных, а также ряда важных концептуальных и аналитических материалов по технологиям баз данных

1963-1980 Эра обработки нереляционных баз данных. Иерархическая и сетевая модели данных. Первое поколение СУБД

	Описание наиболее значимых событий
1963	Завершена разработка первой СУБД **IDS** (Integrated Data Storage) – Интегрированное хранилище данных, компания General Electric, под руководством **Чарльза Бахмана** (C. Bachman) – **сетевая модель данных**
1968-1969	Компания IBM выпустила первую версию своей СУБД **IDMS** (Integrated Database Management System) для платформы IBM/360 – **иерархическая модель данных**

1980- н. время. Реляционные базы данных. Второе поколение СУБД

	Описание наиболее значимых событий
1970	Публикация работы **Эдгара Кодда** (E.Codd) в журнале «*Communication of A CM*», которая послужила основой разработки теории реляционных баз данных.
1975	Опубликован отчет рабочей группы **ANSI/X3/SPARC о трехуровневой архитектуре СУБД**
1976	Опубликована широко известная статья **Питера Чена** (P. Chen), в которой предложена модель данных «сущность-связь» – ER-модель
1978	Опубликован заключительный отчет исследовательской группы ANSI/X3/SPARC по базам данных
1979-1984	В компании IBM инициирован исследовательский проект System R* с целью изучения методов реализации систем распределенных реляционных баз данных

1980	Выпуск первых СУБД для мэйнфреймов: DB2 (корпорация IBM) , Oracle (Oracle)
1981	За цикл работ по созданию теории реляционных баз данных ACM удостоила Э. Кодда Тьюринговской премии
1982	Первые СУБД для микрокомпьютеров
1986	Принят стандарт ANSI языка SQL-86
1987	Принят стандарт ISO SQL-87 — первый международный стандарт языка SQL
1991	Компания Microsoft выпустила Access
1992	Принят международный стандарт ISO SQL-92
1999	Принят стандарт ISO/IEC/ANSI SQL 1999 Издательство «Вильямс» (Киев, Москва, СПб) выпустило в русском переводе 6-е издание книги Криса Дейта «Введение в системы баз данных» (K. J. Date)

1985- настоящее время

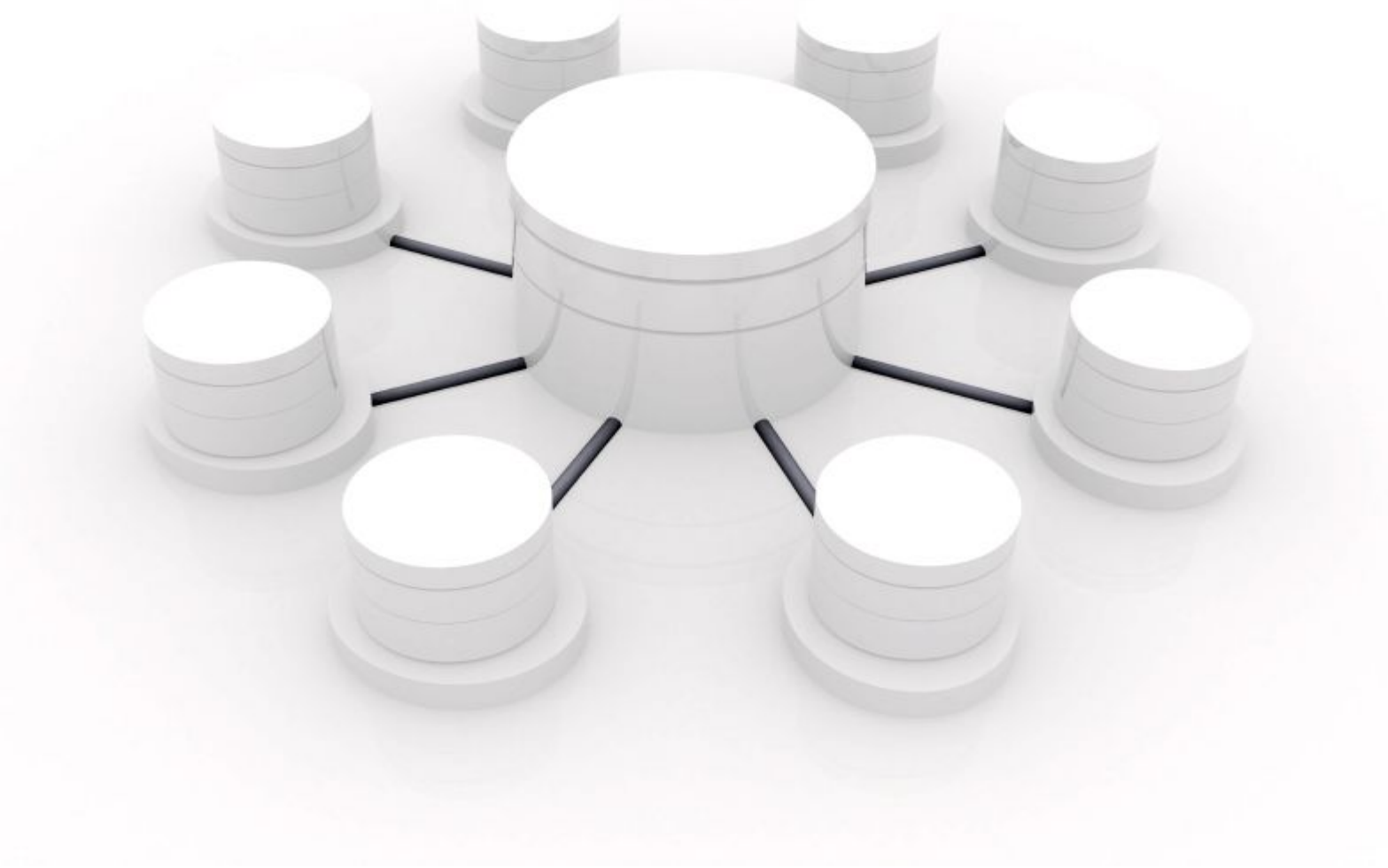
В ответ на все возрастающую сложность приложений баз данных появились **две** новые разновидности систем: **объектно-ориентированные СУБД,** или ОО СУБД (Object-Oriented DBMS — OODBMS), и **объектно-реляционные СУБД,** или ОР СУБД (Object-Relational DBMS — ORDBMS). Реализации подобных моделей представляют собой СУБД **третьего поколения**.

	Описание наиболее значимых событий
1985	Развитие интереса к объектно-ориентированным СУБД (ООСУБД)
1995	Первые приложения баз данных для Интернета
1997	Применение XML к обработке баз данных
2000-н.в.	Новые стандарты, технологии, модели, архитектуры систем БД

Поколения СУБД. Их характеристика.

- ***К СУБД первого поколения*** *относят СУБД на основе сетевой модели данных (их иногда называют CODASYL-системы) и системы на основе иерархических подходов.*
- ***СУБД второго поколения – реляционные***
- ***СУБД третьего поколения – объектно-реляционные и объектно-ориентированные.***

Классификация СУБД. Архитектура СУБД. Основные компоненты СУБД.



- Классификация СУБД. Архитектура СУБД. Схема ее работы и характеристика основных компонентов. Роль администраторов баз данных. Уровни представления информации в базах данных. Логическая и физическая независимость данных и средства ее обеспечения.

Критерии классификации СУБД

По степени универсальности:

- СУБД общего назначения (СУБД ОН) и специализированные СУБД (СпСУБД).

По используемой модели данных

- иерархические, сетевые, реляционные; объектно-ориентированные СУБД.

По методам организации хранения и обработки данных :

- централизованные (локальные, файл – серверные, клиент-серверные) и распределённые СУБД.

По сфере применения

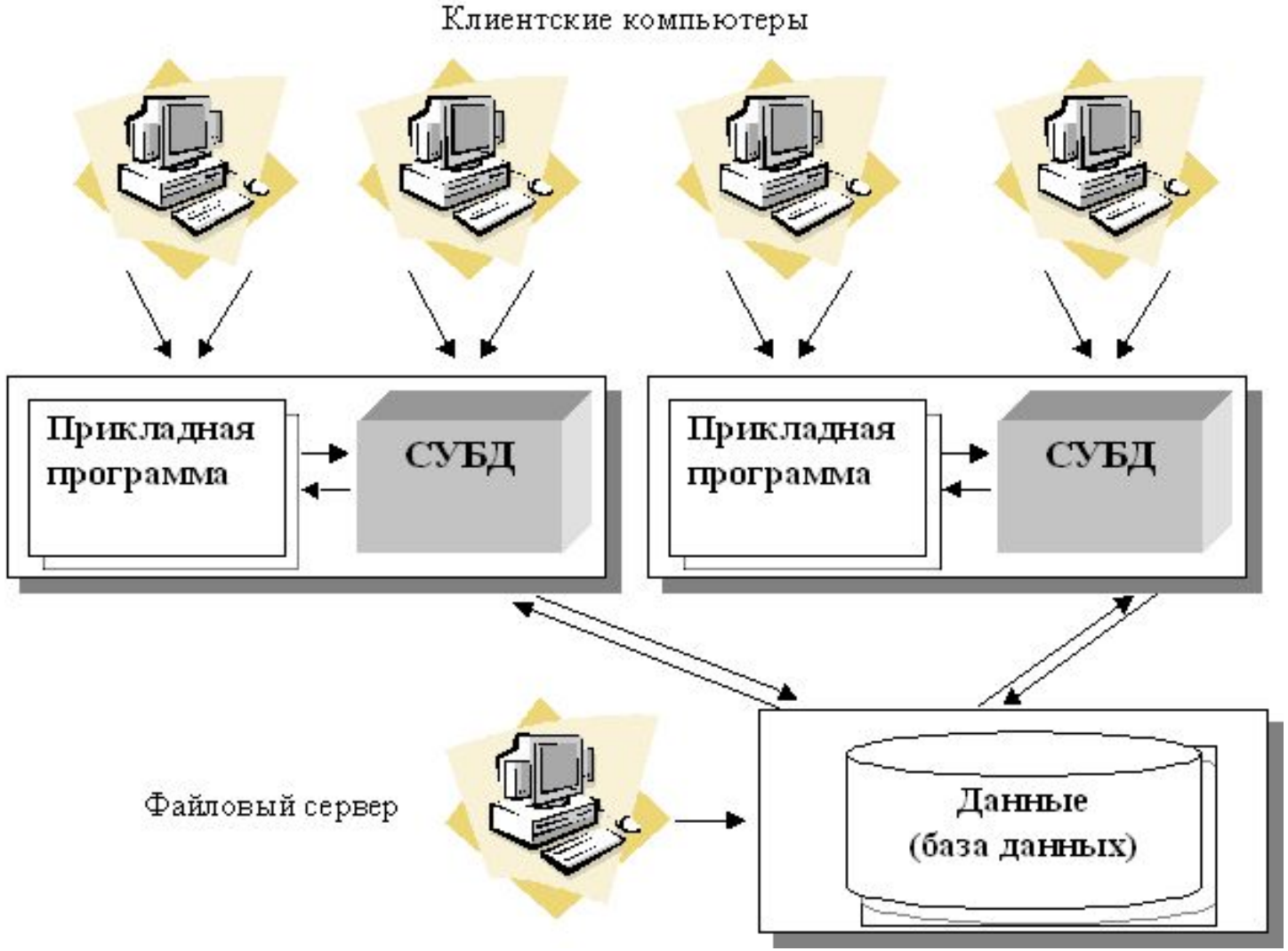
- справочные системы и системы обработки данных.

Классификация по масштабу систем:

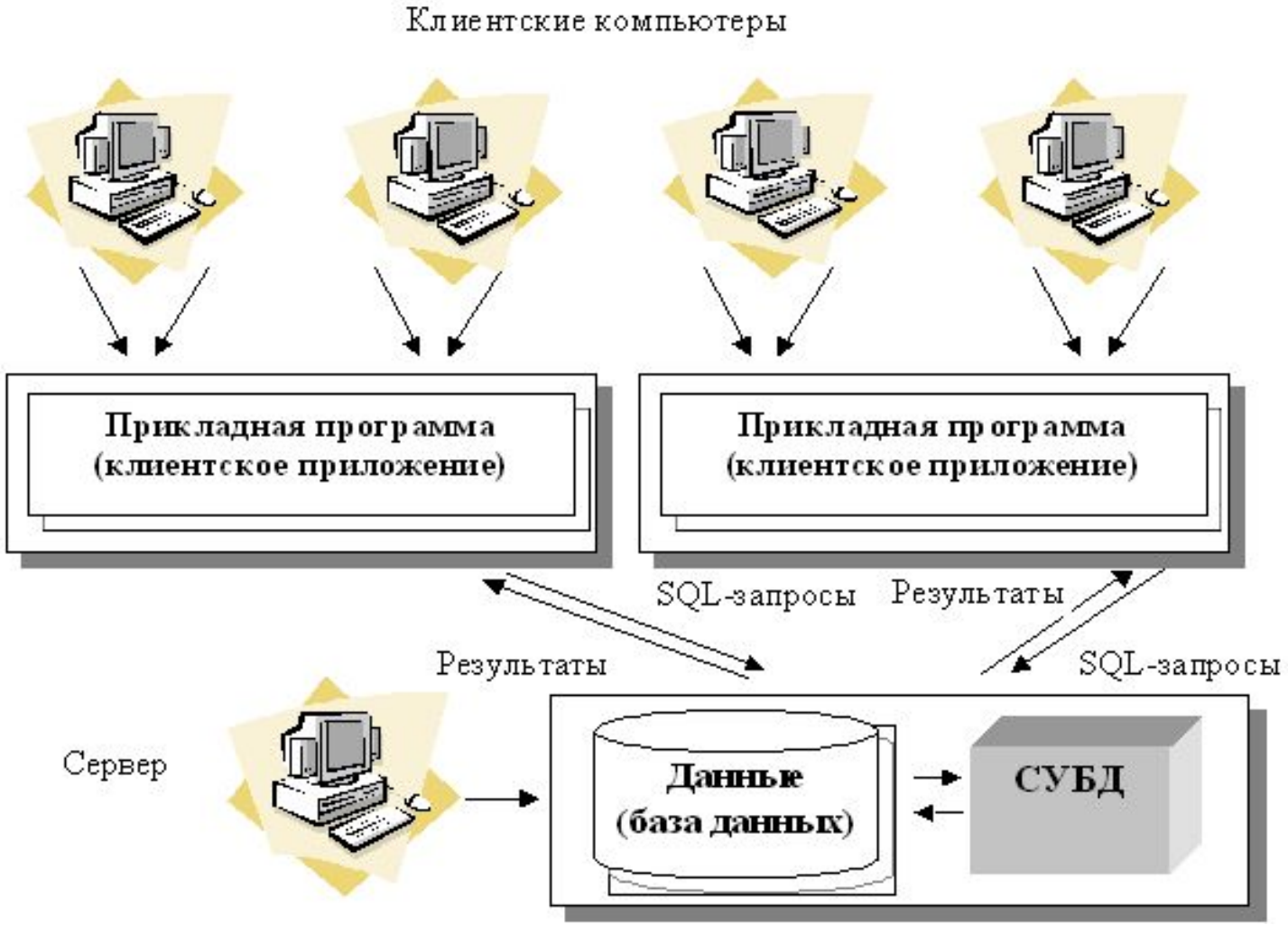
- персональные; уровня группы, отдела, предприятия; корпоративные; географически распределенные.

Локальная СУБД

Файл-серверная СУБД



Клиент-серверная СУБД



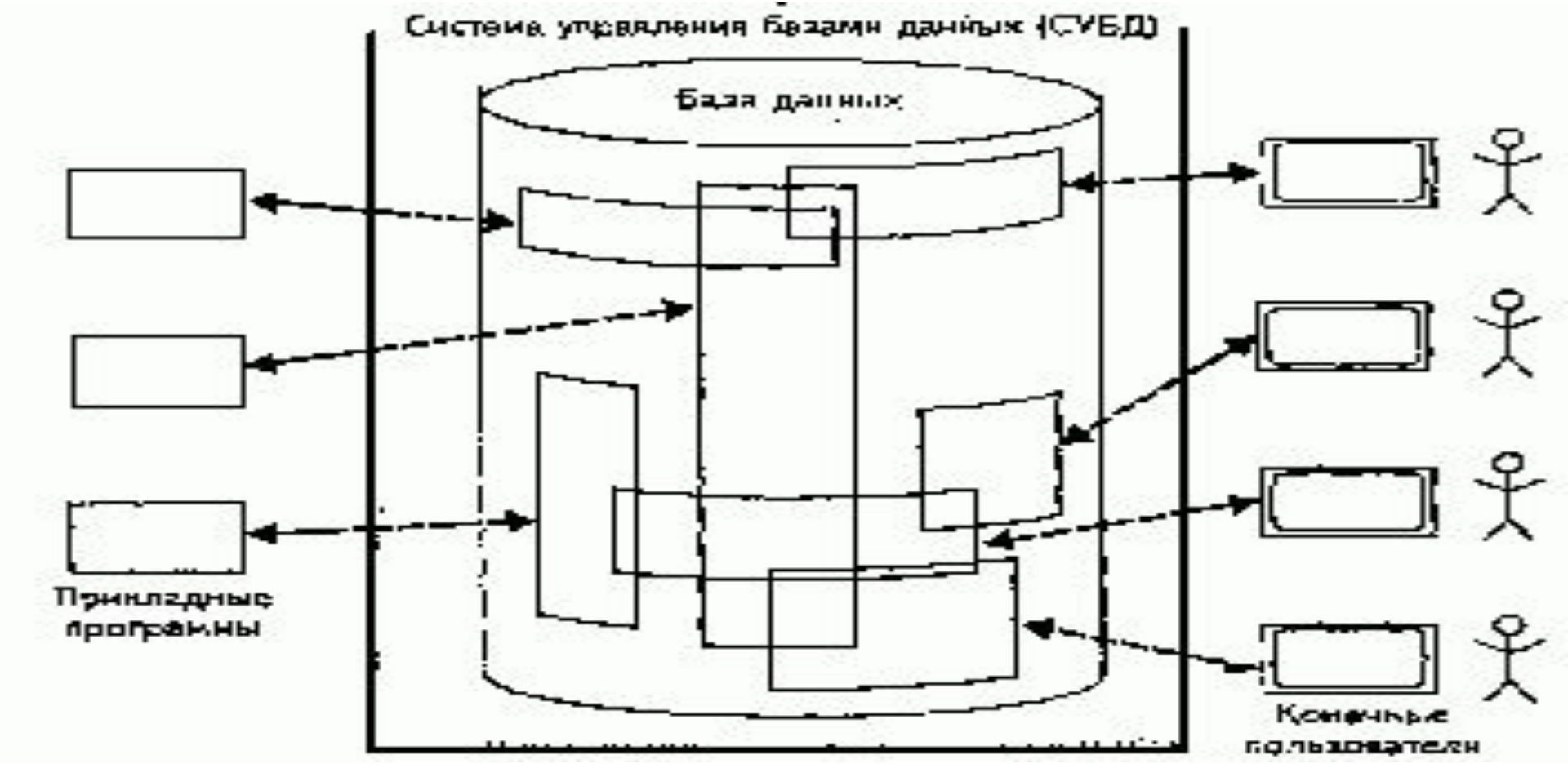
Распределенная СУБД

Распределенная СУБД – это СУБД, поддерживающая работу с распределенными базами данных. Одно из определений распределенной БД:

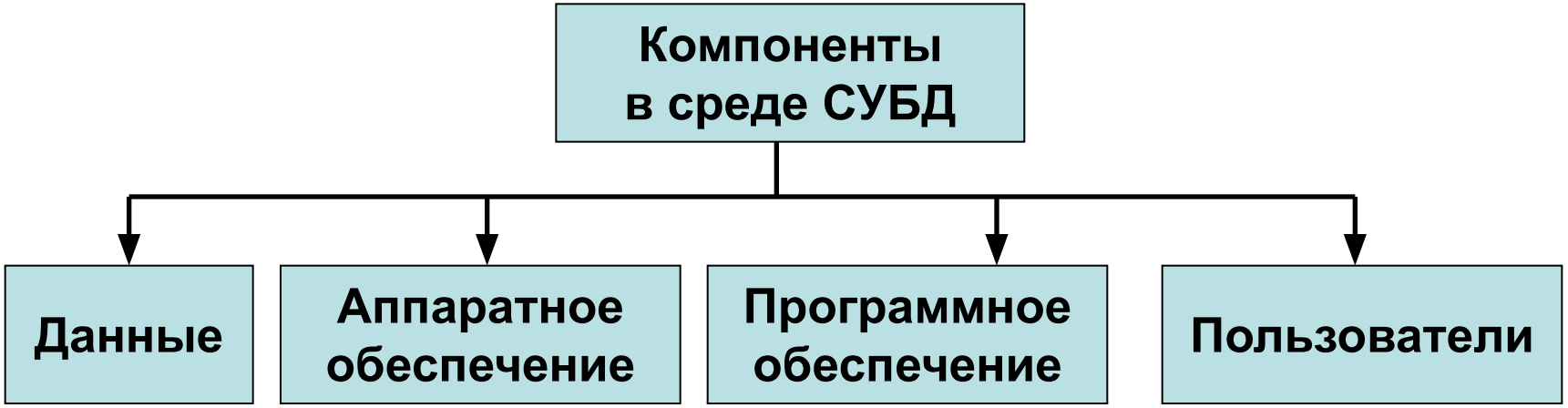
"Распределенная БД - это множество физических баз данных, которые выглядят для пользователя как одна логическая БД". К сожалению на сегодняшний день ни одна СУБД полностью не реализует это определение. Наиболее близко к его реализации подошли следующие СУБД:

- Informix On-Line фирмы Informix Software;
- Ingres Intelligent Database фирмы Ingres Corp;
- Oracle (version 7) фирмы Oracle Corp;
- Sybase System 10 фирмы Sybase Inc.

Упрощенная схема СУБД



Основные компоненты в среде СУБД

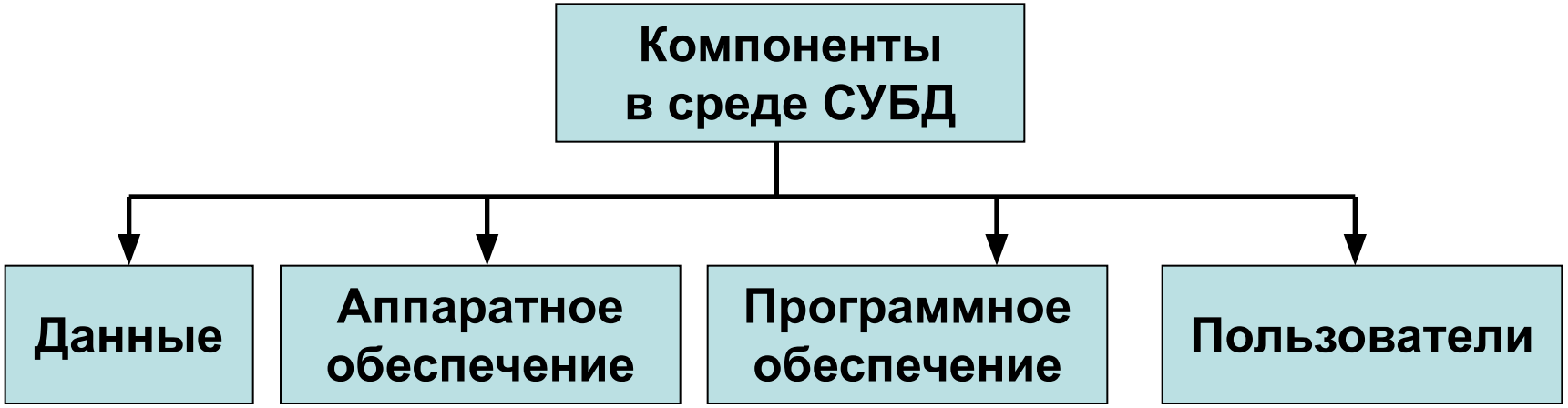


Данные должны быть интегрированными и общими.

Интегрирование – возможность представлять базу данных как объединение нескольких отдельных файлов данных полностью или частично не перекрывающихся.

Общие – возможность использования отдельных областей данных в БД несколькими различными пользователями, причем даже в одно и тоже время(одновременный доступ).

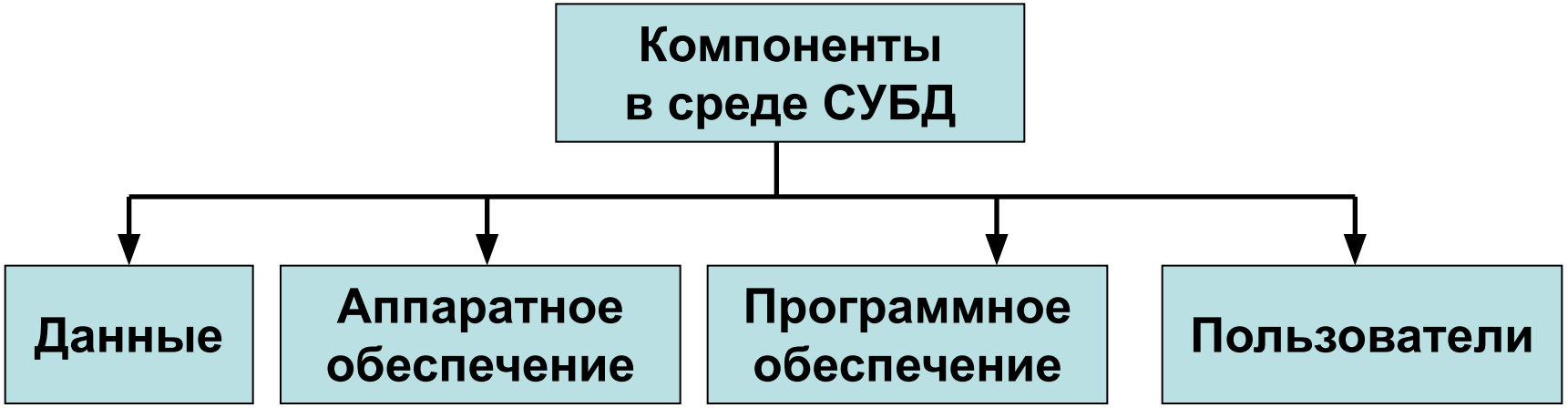
Основные компоненты в среде СУБД



Накопители для хранения информации (обычно диски с перемещаемыми головками) вместе с подсоединенными устройствами ввода-вывода, контроллерами устройств, каналами ввода-вывода и т.д.

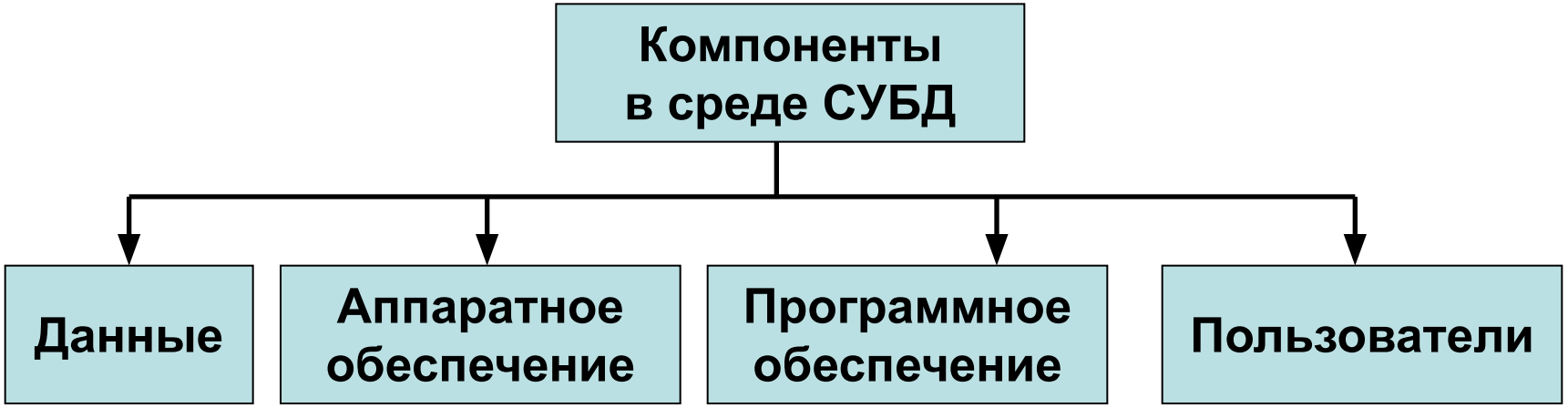
Процессор или процессоры вместе с основной памятью, которая используется для поддержки работы программного обеспечения системы

Основные компоненты в среде СУБД



Диспетчер базы данных (database manager),или система управления базами данных СУБД (database management system(DBMS)). СУБД предоставляет пользователю возможность рассматривать БД как объект более высокого уровня по сравнению с аппаратным обеспечением, а также поддерживает выражаемые в терминах высокого уровня пользовательские запросы(SQL).Кроме СУБД, в программном обеспечении – утилиты, средства разработки приложений, средства проектирования, генераторы отчетов и другие.

Основные компоненты в среде СУБД



Работающие с базами данных пользователи обладают различными знаниями, навыками и сталкиваются с решением различных задач:

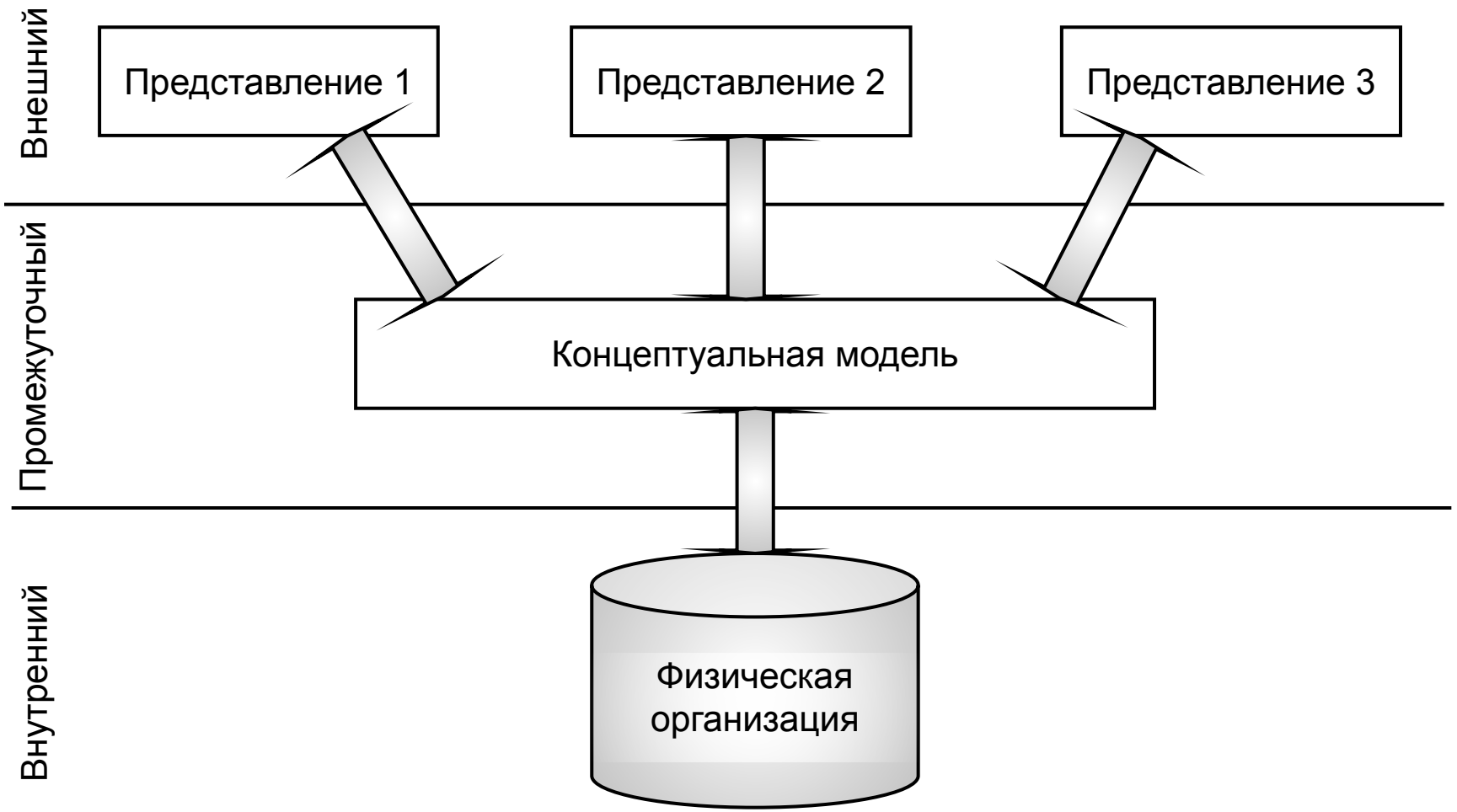
- конечные пользователи;

- разработчики баз данных;

- разработчики приложений;

- администраторы баз данных.

Схема трехуровневой архитектуры ANSI для СУБД



Логическая и физическая независимость уровней при работе с данными

Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных. Это именно то, чего не хватало при использовании файловых систем.

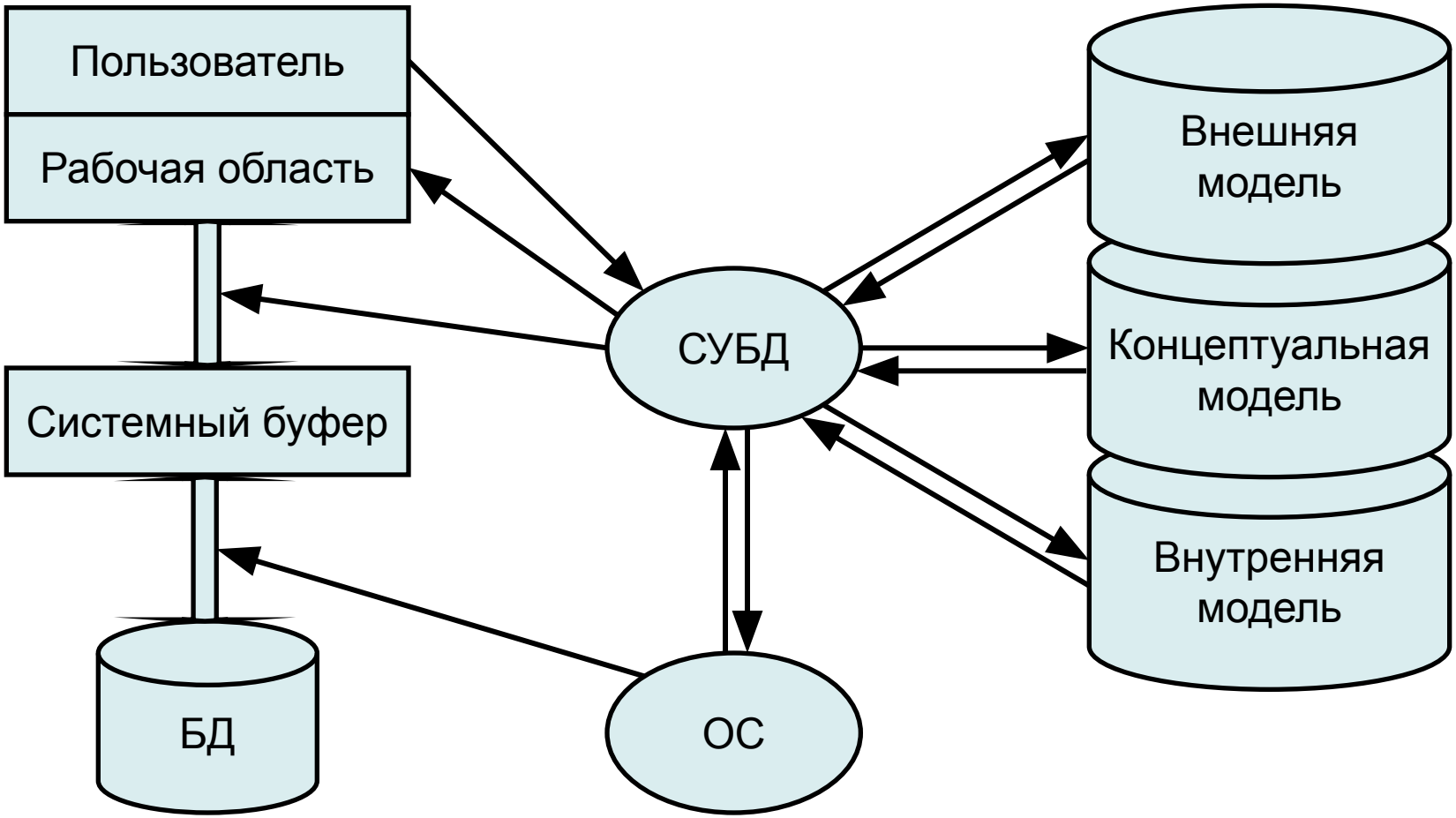
Выделение концептуального уровня позволило разработать аппарат централизованного управления базой данных.

Определение схемы и подсхемы БД.

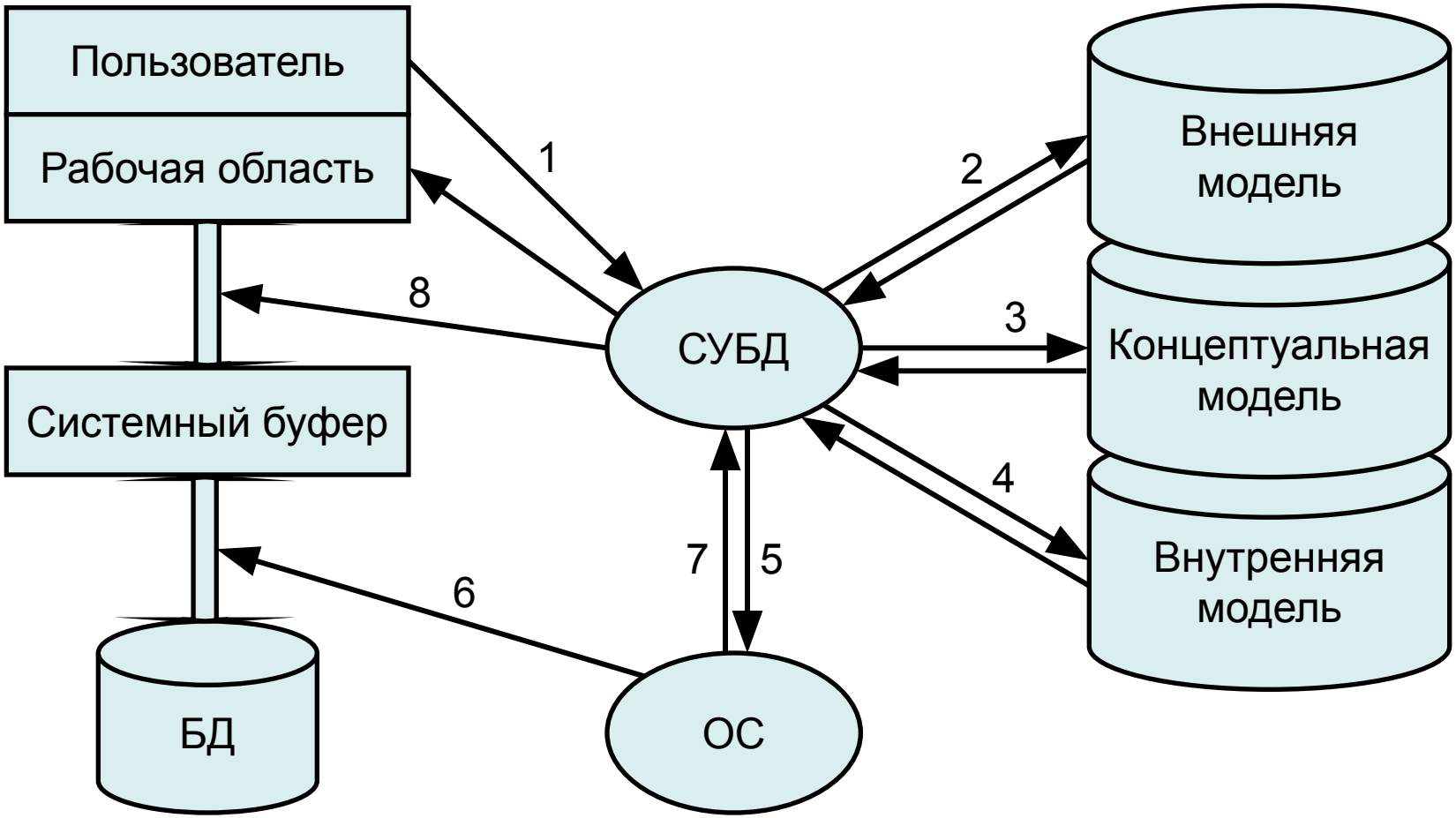
С понятием «трехуровневая архитектура баз данных» связаны понятия «схема» и «подсхема».

Описание общей логической структуры базы данных называют *схемой*. Ее называют иногда *общей моделью* данных. **На основе одной схемы можно составить много различных подсхем** (в зависимости от требований пользователей к БД).

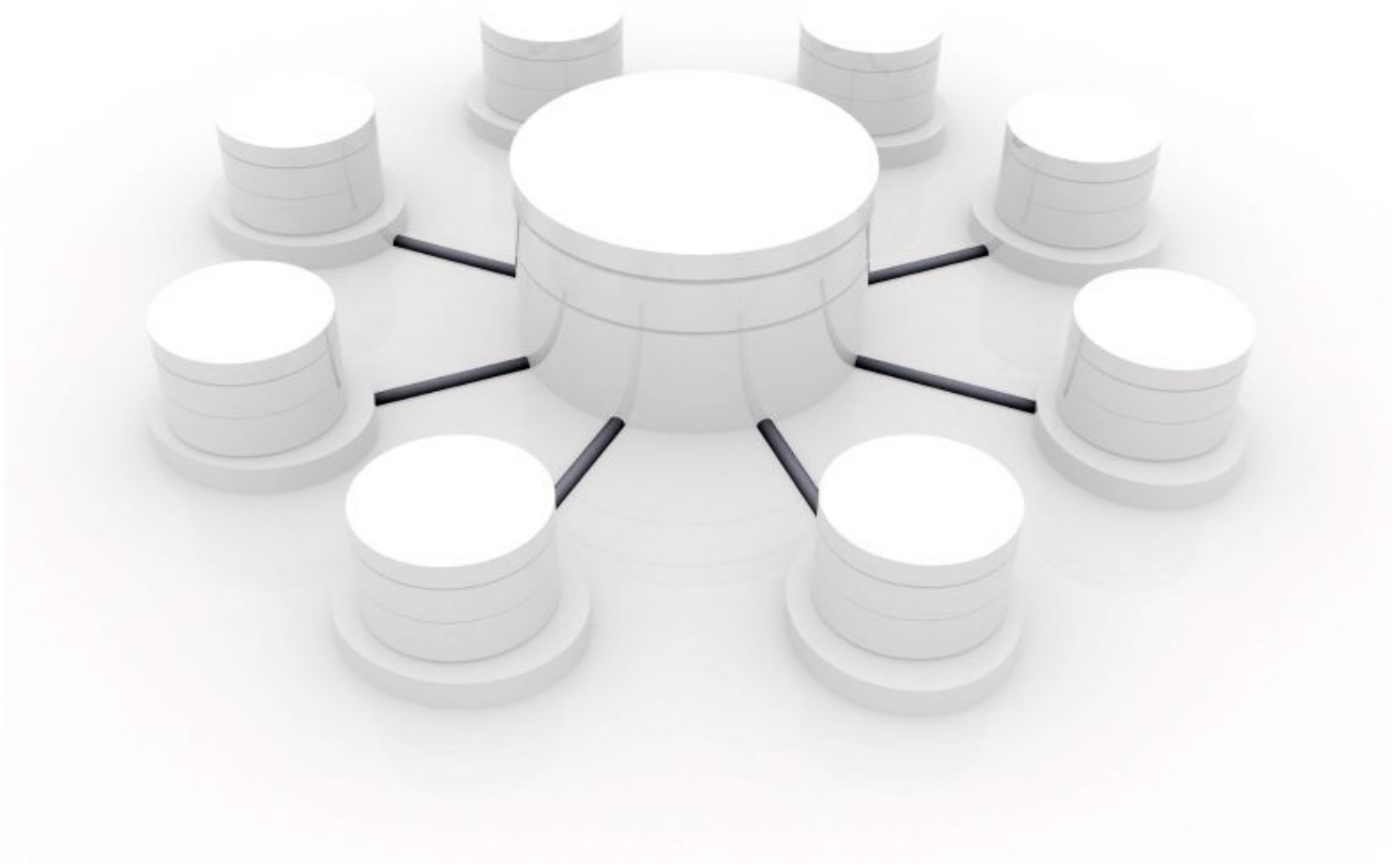
Процесс прохождения пользовательского запроса



Процесс прохождения пользовательского запроса



Модель данных. Классификация моделей данных. Этапы проектирования баз даных



Определение понятия «модель»

Модель - это такой материальный или мысленно представляемый объект, который в процессе познания (созерцания, анализа и синтеза) замещает объект-оригинал.

- Модель — это упрощенное представление реального устройства, процесса, явления.
- Процесс построения и исследования моделей называется моделированием, облегчает изучение имеющихся в реальном устройстве (процессе, явлении) свойств и закономерностей. Применяют для нужд познания (созерцания, анализа, синтеза).

Определение понятия «модель данных»

Модель данных **— это некоторая интерпретация данных, связанная с этапом проектирования БД, которая трактуется как сведения, имеющие определенную структуру.**

Модель данных – это логическое определение объектов, связанное с этапом проектирования БД.

К этому понятию примыкает понятие *МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ*

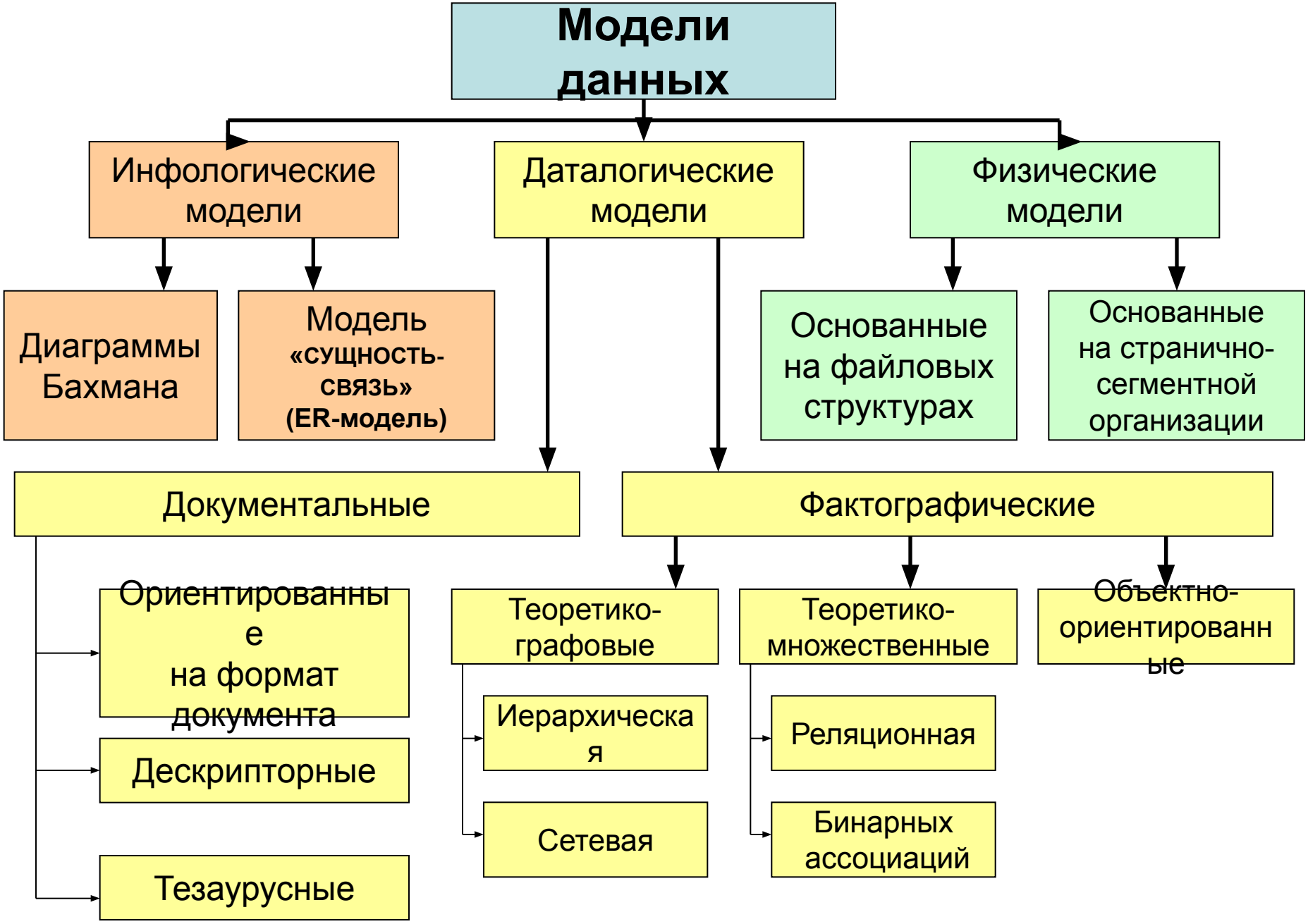
Из «Энциклопедии технологий баз данных» М. Р. Когаловского

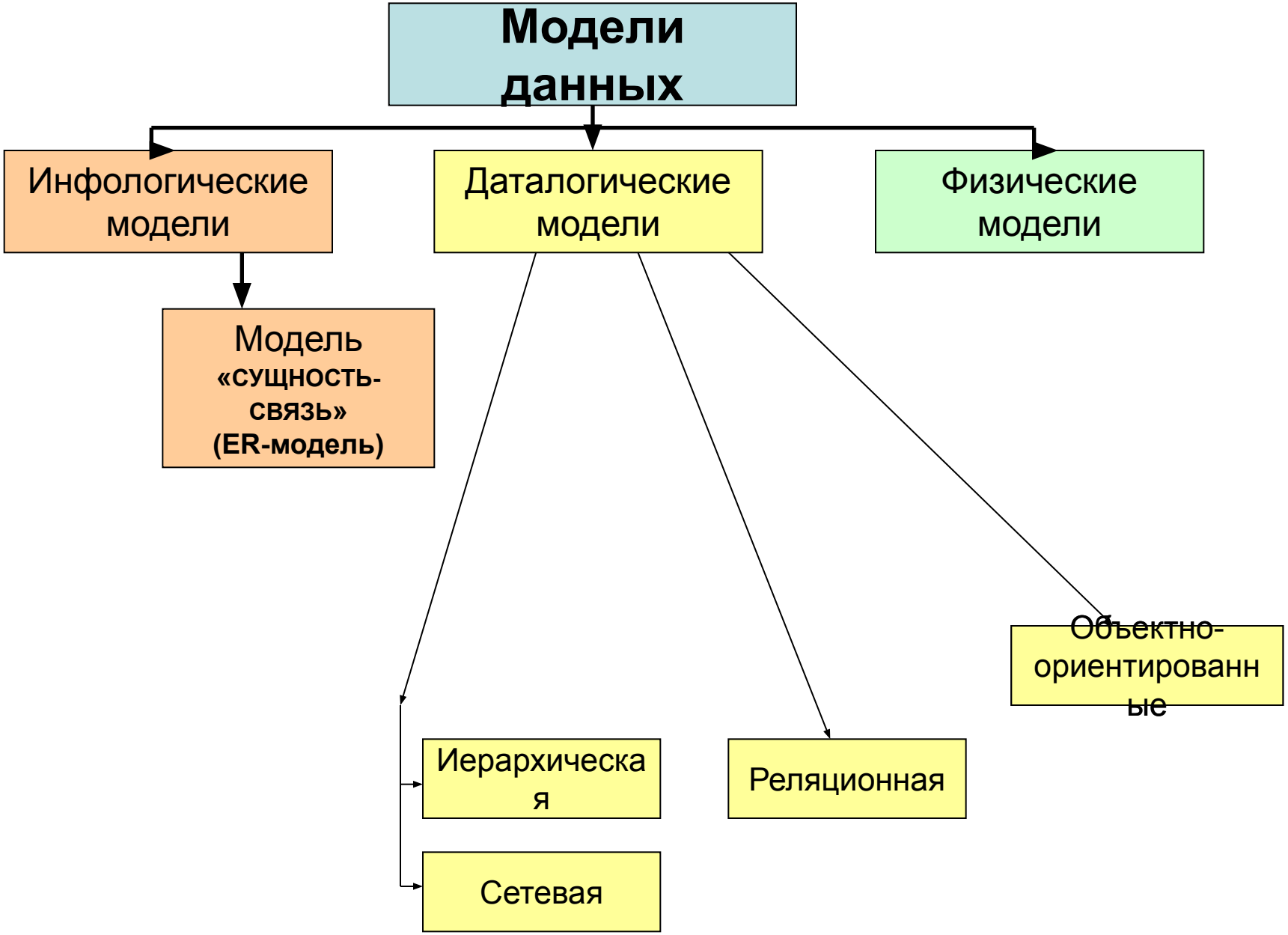
М.087 Модель данных
Data Model
Аббрев. англ.: DM

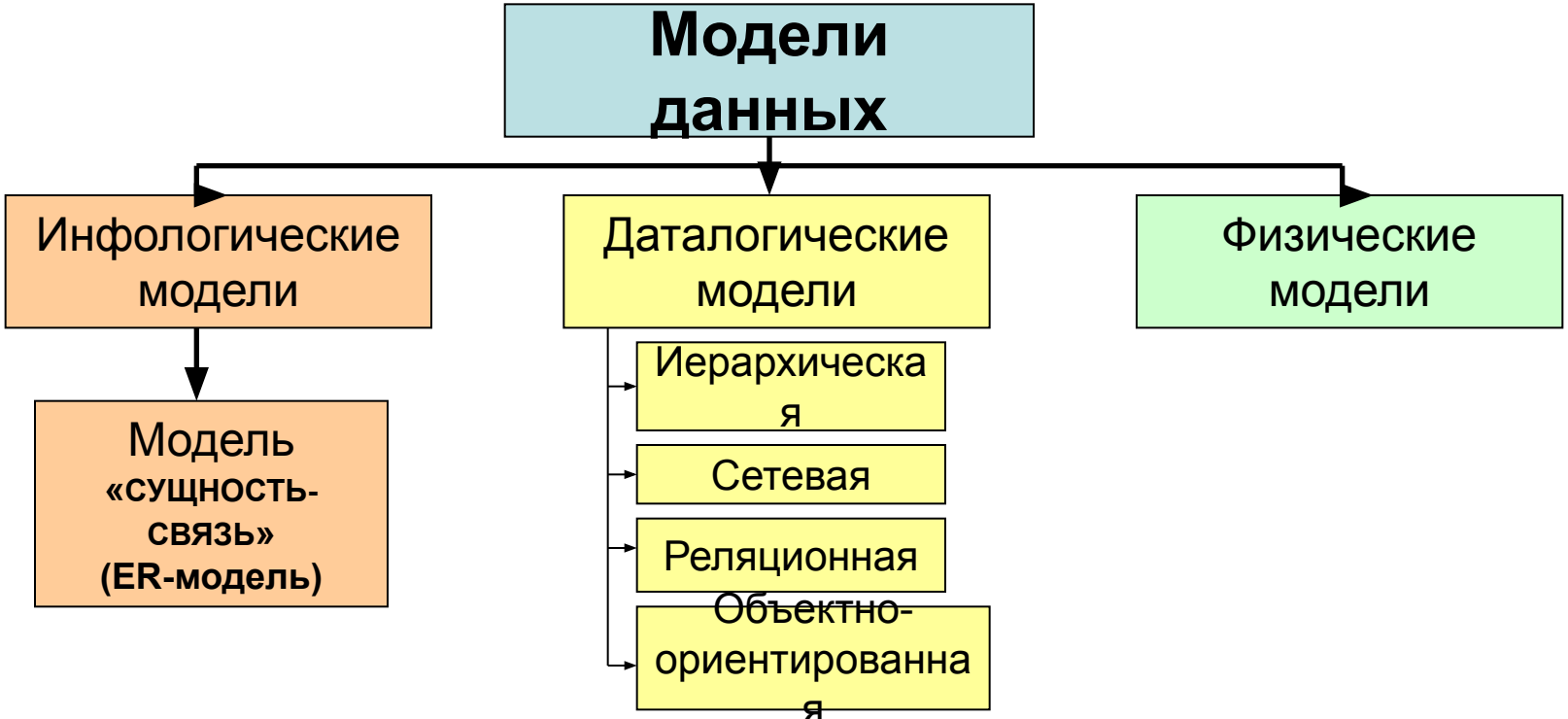
1. Совокупность правил структурирования данных в базах данных, допустимых операций над ними и ограничений целостности, которым они должны удовлетворять. Механизмы каждой СУБД конструируются на основе той или иной модели данных, реализуя комплекс языковых средств определения данных и манипулирования данными, воплощающих ее концепции. Устарелая трактовка этого понятия интерпретирует модель данных как структуру конкретной базы данных.

2. Система типов данных, типов связей между ними и допустимых видов ограничений целостности, которые могут быть для них определены. Здесь имеется в виду современное понимание типа данных как носителя свойств, определяющих и состояние экземпляров типа, и их поведение.

3. Метамодель для описания моделей предметной области в среде выбранной СУБД.


Модели данных
Инфологические модели
Даталогические модели
Физические модели
Диаграммы Бахмана
Модель «сущность-связь» (ER-модель)
Основанные на файловых структурах
Основанные на странично-сегментной организации
Документальные
Фактографические
Ориентированные на формат документа
Дескрипторные
Тезаурусные
Теоретико-графовые
Теоретико-множественные
Объектно-ориентированные
Иерархическая
Сетевая
Реляционная
Бинарных ассоциаций


Модели данных
Инфологические модели
Даталогические модели
Физические модели
Модель «сущность-связь» (ER-модель)
Иерархическая
Сетевая
Реляционная
Объектно-ориентированные



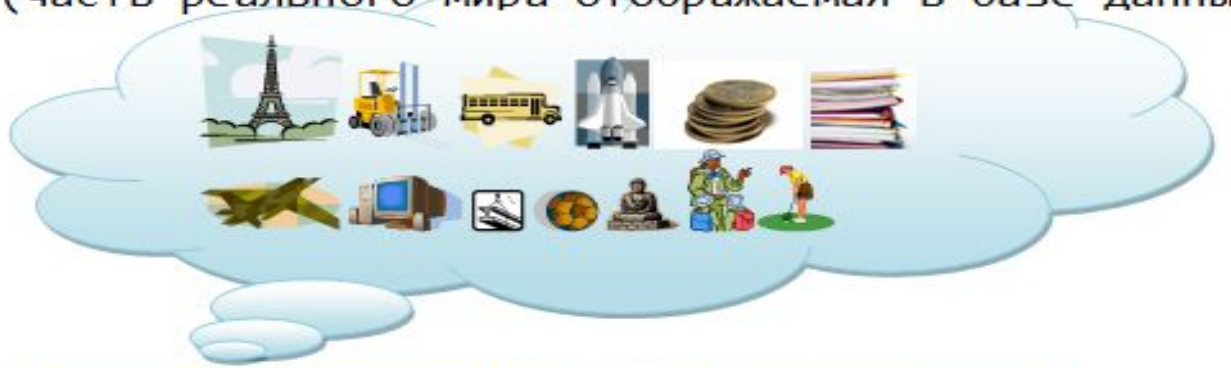
Инфологическое моделирование связано со 2-м этапом проектирования БД: созданием формализованного описания предметной области

Логическое (или даталогическое) моделирование осуществляется после этапа выбора СУБД. Этот тип модели полностью зависит от типа модели, поддерживаемой выбранной системой.

Физическое моделирование заключается в выборе эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы.

Предметная область

(часть реального мира отображаемая в базе данных)



ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Формализованное, обобщенное, не привязанное к каким-либо СУБД описание предметной области

Модели описания, используемые СУБД

ДАТАЛОГИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Описание на языке конкретной СУБД

ФИЗИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Описание хранимых данных на уровне операционной системы

БАЗА ДАННЫХ

Этапы проектирования БД

- *1. **Системный анализ и словесное описание информационных объектов предметной области**.*
- *2. **Информационно-логическое (инфологическое) проектирование** - создание инфологической модели предметной области – частично формализованного описания объектов предметной области в терминах некоторой семантической модели. Наиболее традиционная из них называется моделью сущности – связи (E/R- модель, entity-relationship), имеет графическую природу (прямоугольники, стрелки, ромбы).*
- *3. **Выбор СУБД** и других инструментальных программных средств.*
- *4. **Даталогическое (или логическое) проектирование**, т.е. описание БД в терминах принятой даталогической модели данных (наиболее распространена реляционная, т. е. E/R-модель преобразуем в реляционную).*
- *5. **Физическое проектирование БД**, т.е. выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.*

Подходы к выбору состава и структуры предметной области

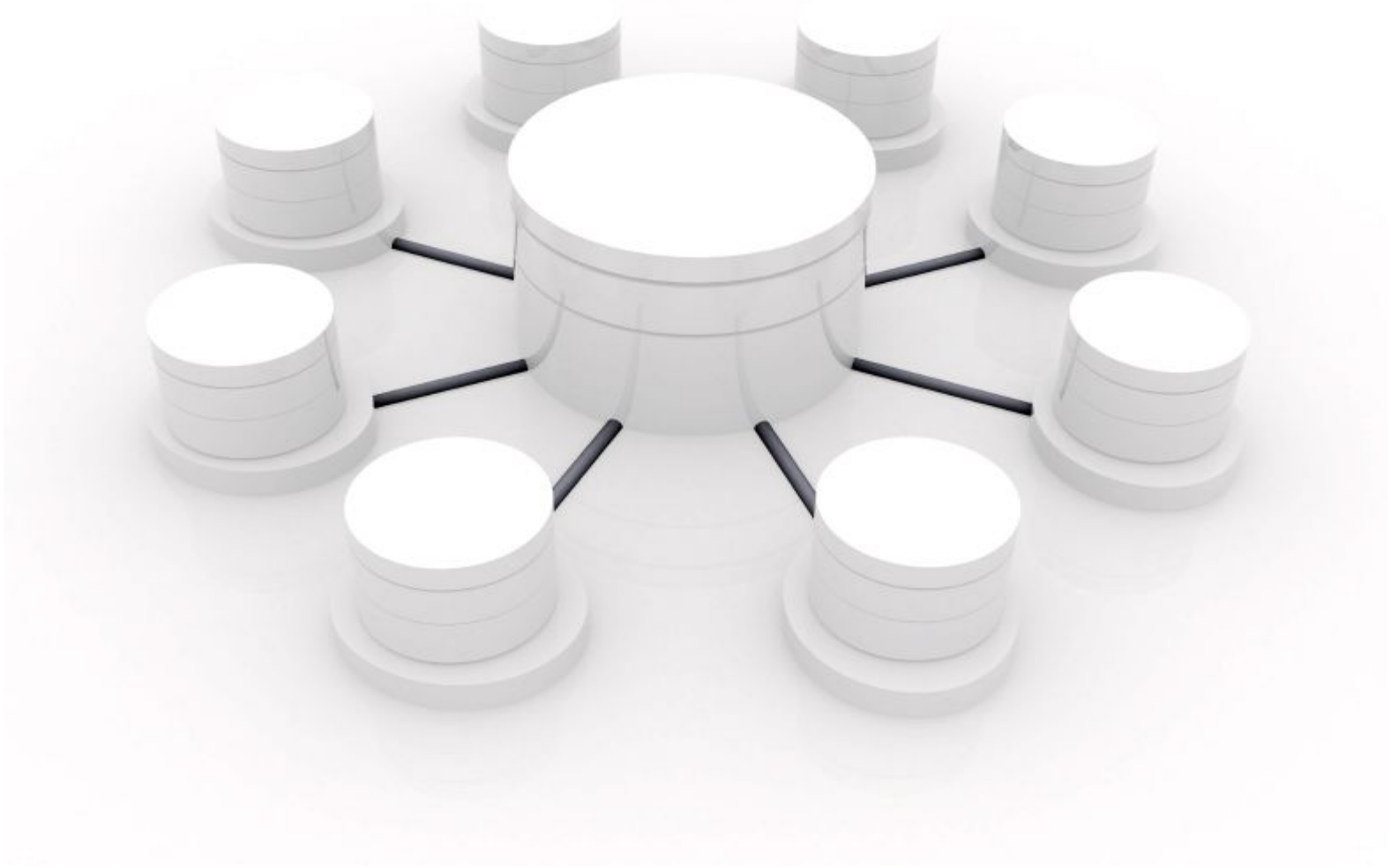
- *1. Функциональный подход – он реализует принцип движения “ от задач ” и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае мы можем четко выделить необходимый минимальный набор объектов предметной области, которые должны быть описаны.*

- *2. Предметный подход – когда информационные потребности будущих пользователей БД жестко не фиксируются. Они могут быть многоаспектными и динамичными. БД, конструируемая при этом, называется предметной.*

Примерная тематика баз данных для лабораторных работ

1. Расписание маршрутов движения транспорта.
2. Каталог учебных дисциплин.
3. Учет прихода и расхода товара со склада.
4. Журнал посещаемости лекций и лабораторных работ.
5. Учет оборудования организации.
6. Учет программного обеспечения. ПО на CD, FDD, HDD и т.д.
7. Картотека библиотеки.
8. БД кафедры. Группы, студенты, успеваемость.
9.Телефонный справочник университета.
10. Учет мебели.
11. Меню столовой.
12. Комнатные растения.
13. Фильмотека. Фильмы, актеры, режиссеры.
14. Футбольный турнир.
15. БД компьютерных игр.
16. Справочник по радиодеталям.
17. Словарь компьютерных терминов.
18. БД городов.
19. Географическая база данных.
20. БД рецептов.
21. БД по радиодеталям.
22. Электронный словарь - переводчик.
23. БД автопарка. Учет автомобилей: за кем закреплен, километраж, путевки.
24. Медицинская БД. Картотека больных
25. БД web-ссылок по темам, категориям и т.д.
26. БД контрольных вопросов по дисциплинам, темам и разделам.
27. БД абитуриентов.
28. БД анкетирования.
29. БД буфета.
30. Учет помещений университета.

Инфологическое моделирование.
Модель «сущность – связь» в нотации Мартина.



Инфологическая модель предметной области – это частично формализованное описание объектов предметной области в терминах некоторой семантической модели. Более традиционная из них называется моделью сущность – связь (E/R- модель, entity-relationship), имеет графическую природу (прямоугольники, стрелки, ромбы).

Модель «сущность – связь»

- ***E/R-модель (или модель сущность – связь) создана Питером Ченом в 1976 году.***

E/R-модель стала фактическим стандартом при инфологическом моделировании БД по следующим причинам.

1) большинство современных CASE-средств содержат инструментальные средства для описания данных в формализме этой модели;

2) разработаны методы автоматического преобразования проекта БД из E/R-модели в реляционную, при этом преобразование выполняется в даталогическую модель, соответствующую конкретной СУБД.

Компоненты E/R-модели:

1. ***Сущность – это реальный или представляемый набор однотипных объектов, информация о котором характеризует предметную область. В системе существует множество экземпляров данной сущности (если проводить аналогию с ООП, то множества сущностей – класс, каждая сущность – объект (экземпляр класса).***

2. Атрибуты – значения, описывающие свойства сущности. Набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности, называется ключевым.

3. Связи – бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой. Если есть связь между двумя сущностями, то она определяет взаимосвязь между экземплярами одной и другой сущности.

Типы связей в ER-модели

С точки зрения множественности:

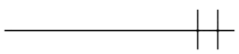
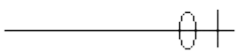
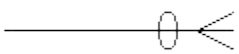
- *Один к одному (1:1),*
- *один ко многим (1:M),*
- *многие ко многим (M:M).*

С точки зрения обязательности:

- *Обязательная (экземпляр первой сущности ДОЛЖЕН быть связан с экземпляром второй сущности)*
- *Необязательная (экземпляр первой сущности МОЖЕТ быть связан с экземпляром второй сущности)*

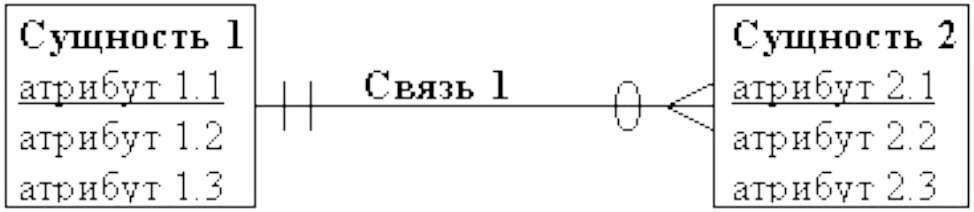
Нотация Мартина

Список атрибутов приводится внутри прямоугольника, обозначающего сущность. Ключевые атрибуты подчеркиваются. Связи изображаются линиями, соединяющими сущности, вид линии в месте соединения с сущностью определяет кардинальность связи.

Обозначение	Кардинальность
	нет
	1,1
	0,1
	M,N
	0,N
	1,N

Нотация Мартина

Имя связи указывается на линии ее обозначающей.
Пример:



Пример описания предметной области «Библиотека»

Пусть требуется разработать информационную систему **для автоматизации учета получения и выдачи книг в библиотеке**. Система должна предусматривать режимы ведения систематического каталога, отражающего перечень **областей знаний**, по которым имеются **книги** в библиотеке. Области знаний в систематическом каталоге могут иметь **уникальный внутренний номер** и **полное наименование**. Каждая книга может содержать сведения из нескольких областей знаний. Каждая книга в библиотеке может присутствовать в нескольких экземплярах. **Книга**, хранящаяся в библиотеке, характеризуется следующими параметрами:

- уникальный шифр (ISBN);
- название;
- фамилии авторов (могут отсутствовать);
- место издания (город);
- издательство;
- год издания;
- количество страниц;
- стоимость книги;
- количество экземпляров книги в библиотеке.

Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN).

В библиотеке **ведется картотека читателей.**

На каждого читателя в картотеку заносятся следующие сведения:

- фамилия, имя, отчество;
- домашний адрес;
- телефон (будем считать, что у нас два телефона — рабочий и домашний);
- дата рождения.

Каждому читателю присваивается **уникальный номер читательского билета**.

Каждый читатель может одновременно держать на руках не более 5 книг. Читатель не должен одновременно держать более одного экземпляра книги одного названия.

Каждая книга в библиотеке может присутствовать в нескольких экземплярах. **Каждый экземпляр** имеет следующие характеристики:

- уникальный инвентарный номер;
- шифр книги, который совпадает с уникальным шифром из описания книг;
- место размещения в библиотеке.

В случае выдачи экземпляра книги читателю в библиотеке хранится специальный вкладыш (**листок читательского требования**), в котором должны быть записаны следующие сведения:

- номер билета читателя, который взял книгу;
- дата выдачи книги;
- дата возврата.

Предусмотреть следующие **ограничения на информацию в системе**:

- Книга может не иметь ни одного автора.
- В библиотеке должны быть записаны читатели не моложе 17 лет.
- В библиотеке присутствуют книги, изданные начиная с 1960 по текущий год.
- Каждый читатель может держать на руках не более 5 книг.
- Каждый читатель при регистрации в библиотеке должен дать телефон для связи: он может быть рабочим или домашним.
- Каждая область знаний может содержать ссылки на множество книг, но каждая книга может относиться к различным областям знаний.

С данной информационной системой должны работать следующие **группы пользователей**:

- библиотекари;
- читатели;
- администрация библиотеки.

При работе с системой **библиотекарь должен иметь возможность решать следующие задачи:**

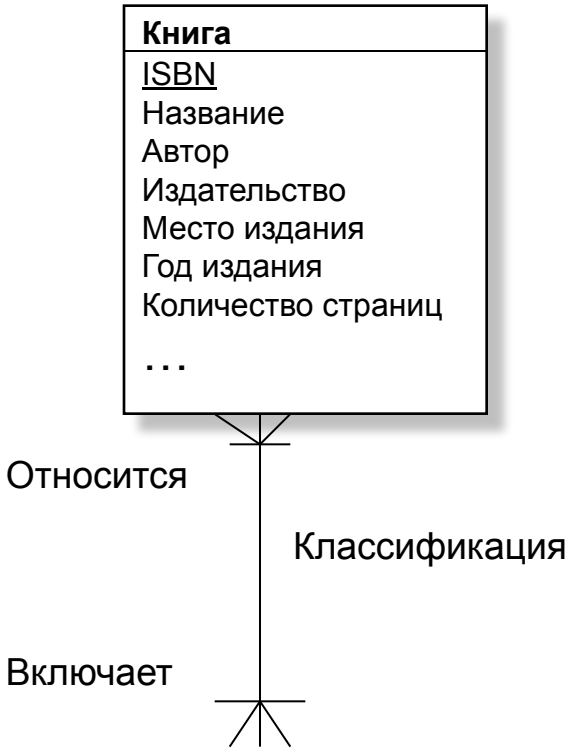
- Принимать новые книги и регистрировать их в библиотеке.
- Относить книги к одной или к нескольким областям знаний.
- ……..

Читатель должен иметь возможность решать следующие задачи:

- Просматривать системный каталог, то есть перечень всех областей знаний, книги по которым есть в библиотеке.
- По выбранной области знаний получить полный перечень книг, которые числятся в библиотеке.

Этот пример показывает, что перед началом разработки необходимо иметь точное представление о том, что же должно выполняться в нашей системе, какие пользователи в ней будут работать, какие задачи будет решать каждый пользователь. И это правильно, ведь когда мы строим здание, мы тоже заранее предполагаем: для каких целей оно предназначено, в каком климате оно будет стоять, на какой почве, и в зависимости от этого проектировщики могут предложить нам тот или иной проект. Но, к сожалению, очень часто по отношению к базам данных считается, что все можно определить потом, когда проект системы уже создан. Отсутствие четких целей создания БД может свести на нет все усилия разработчиков, и проект БД получится «плохим», неудобным, не соответствующим ни реально моделируемому объекту, ни задачам, которые должны решаться с использованием данной БД.

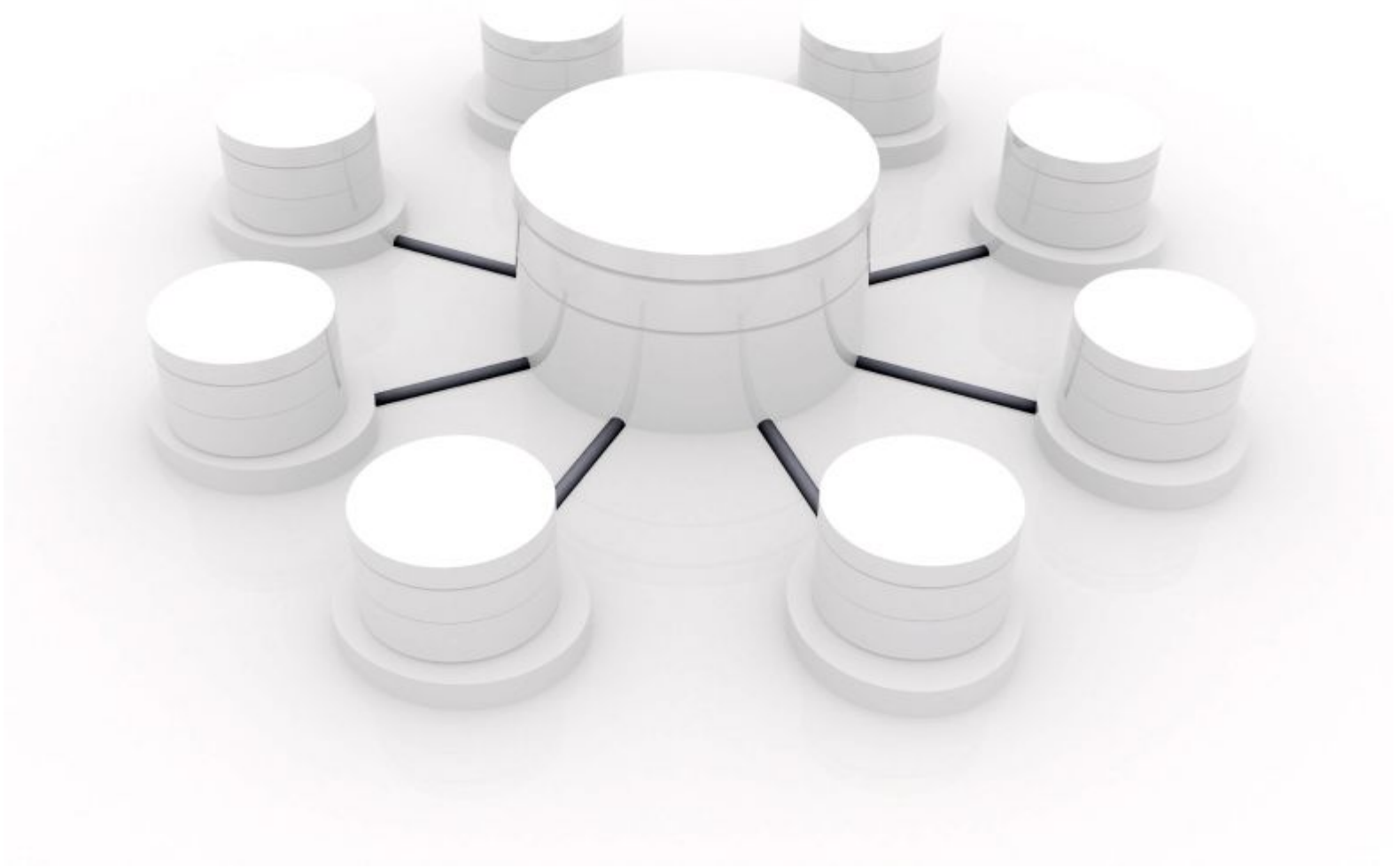
Строим инфологическую модель сущность-связь в нотации Crow’s Foot



Уникальный шифр (ISBN);
Название;
Фамилии авторов (могут отсутствовать);
Место издания (город);
Издательство;
Год издания;
Количество страниц;
Стоимсть книги;
Количество экземпляров книги в библиотеке.

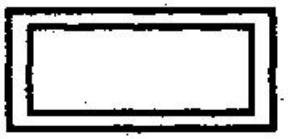
Инфологическая модель БД «Библиотека»

Модель «сущность – связь» в нотации Питера Чена. Расширенная ER-модель.

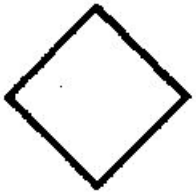


Рассмотрим трактовку инфологической модели ее создателем, Питером Ченом.

Класс сущности



Класс слабой сущности



Тип связи

Определяющий тип связи

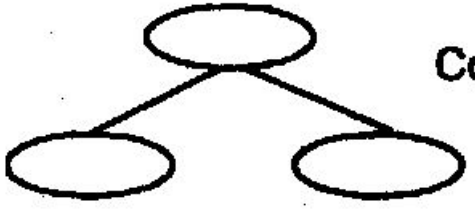
Атрибут

Ключевой атрибут

Дискриминатор или частичный ключ

Производный атрибут

Многозначный атрибут

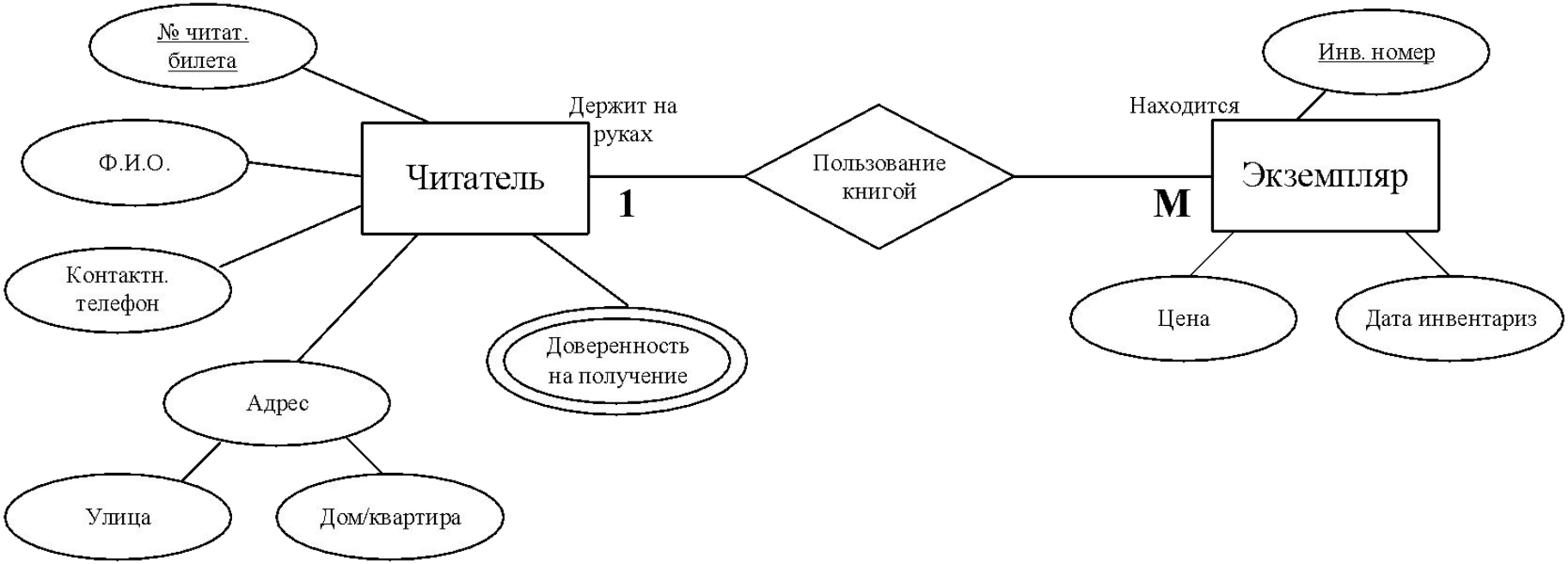


Составной атрибут

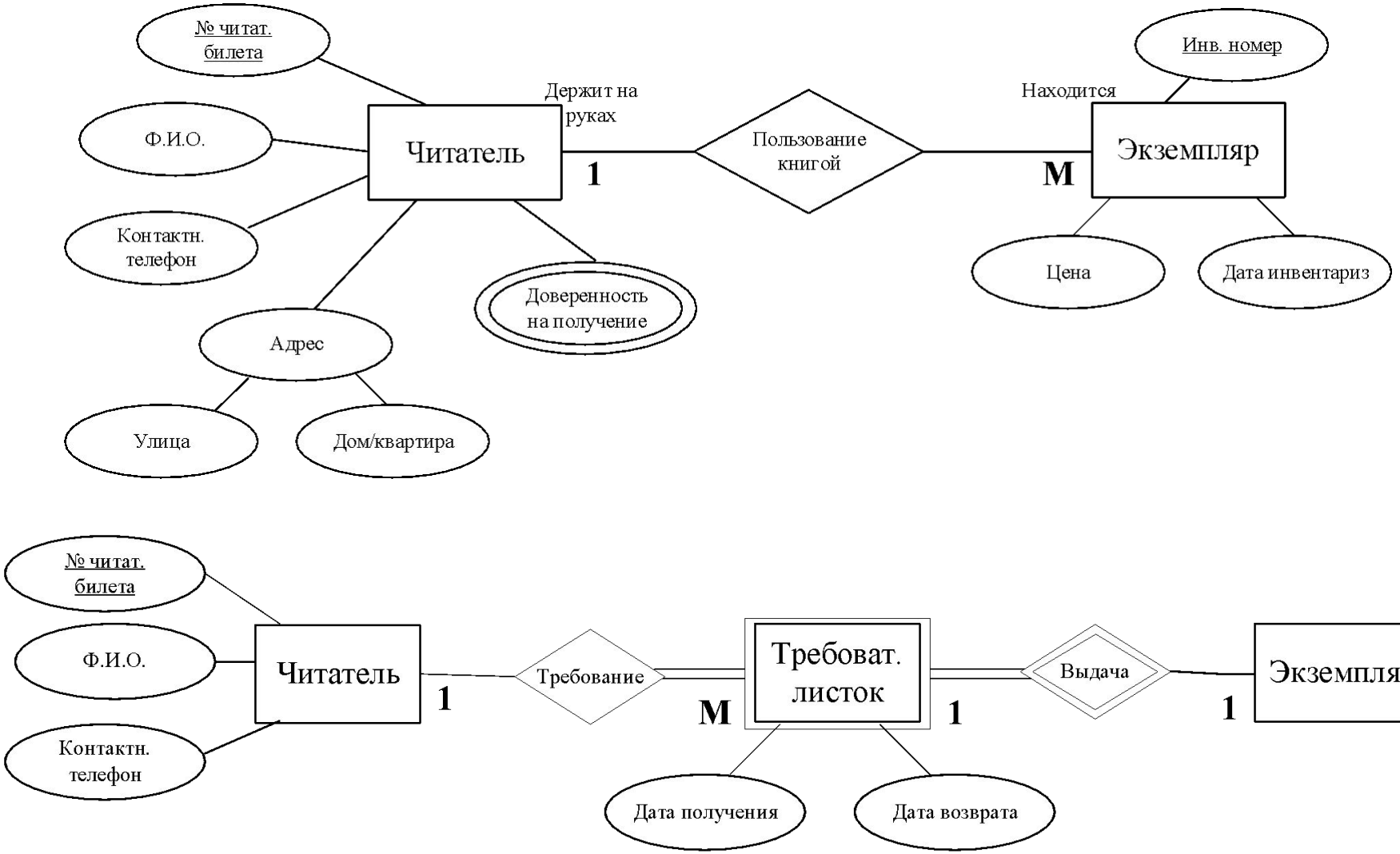
Обозначения кардинальности

1	Не более одной связанной сущности
M	Несколько (ноль и более) связанных сущностей
i . . . j	По меньшей мере i, но не более j связанных сущностей
═══	Должен участвовать в связи
———	Может участвовать в связи

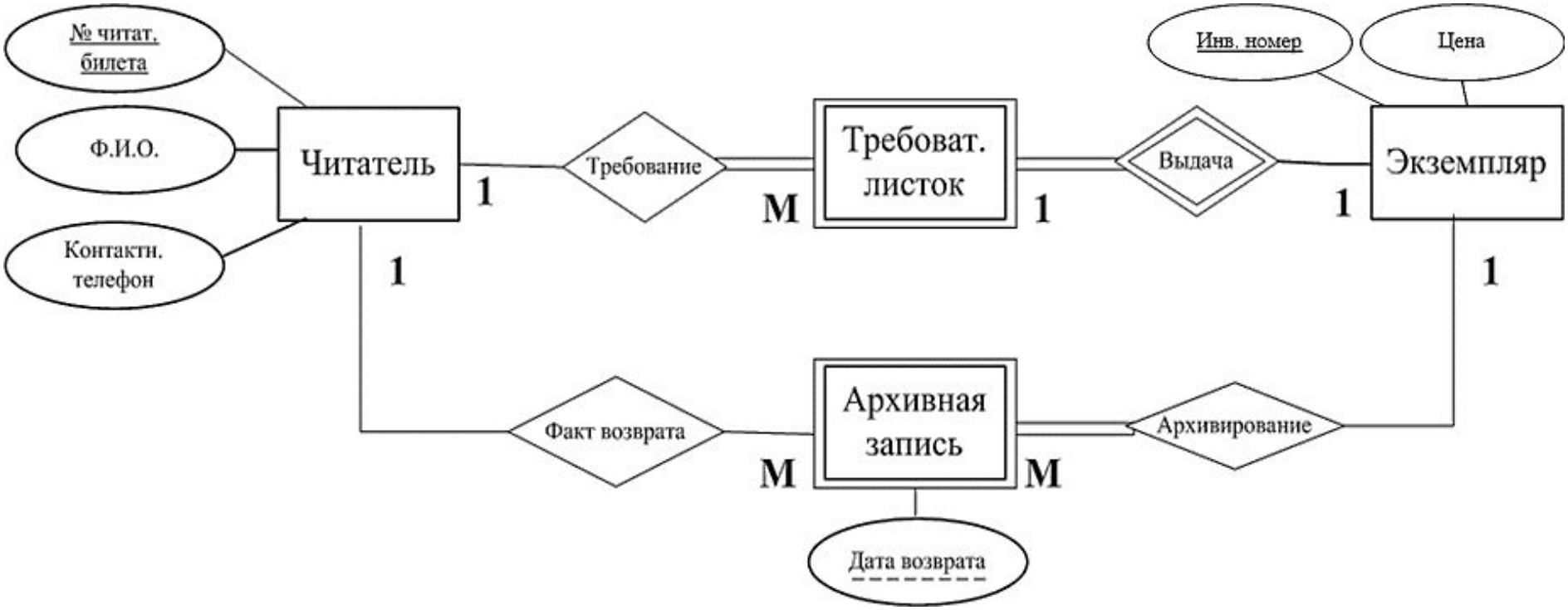
Фрагмент ER-модели «Библиотека»



Пример «слабой» сущности



Пример нескольких связей между сущностями

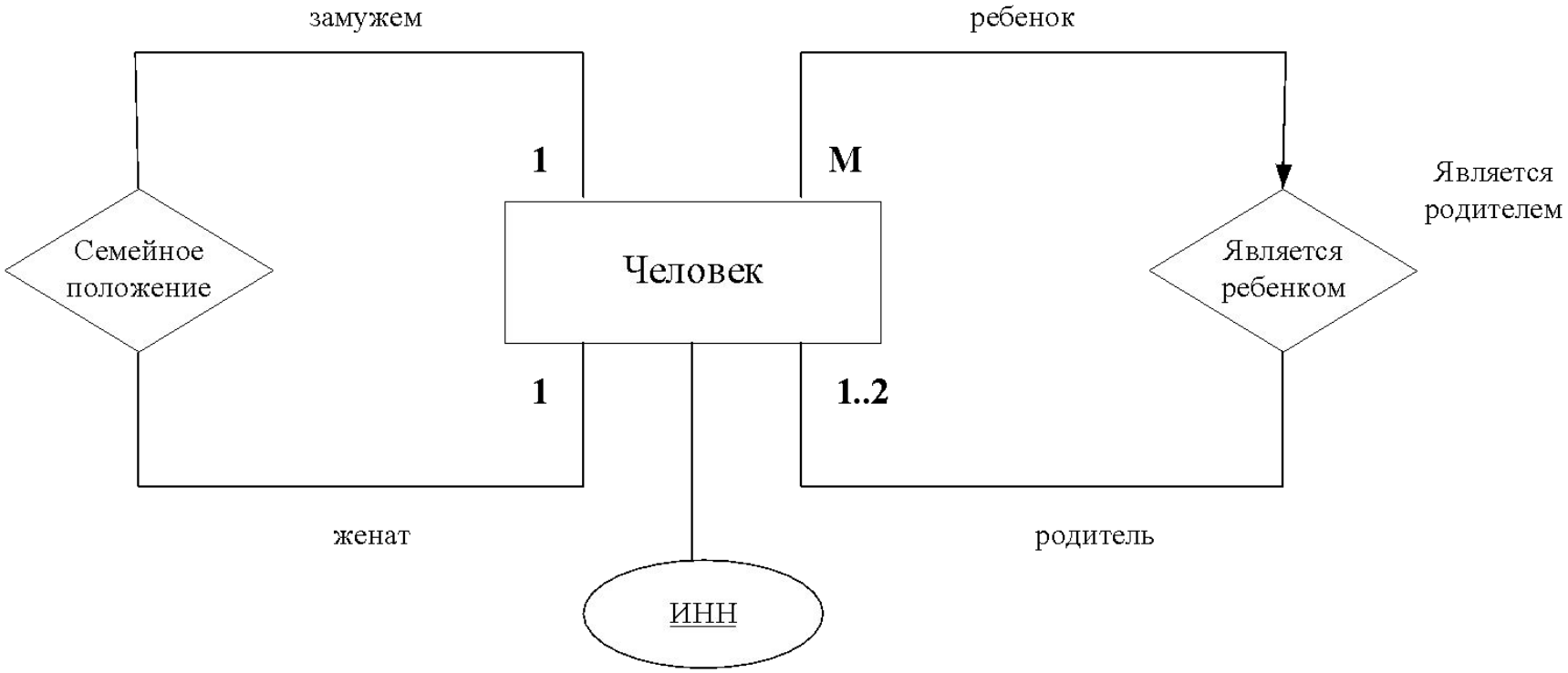


Понятие дискриминатора

Чтобы уникальным образом определить сущность «Архивная запись», следует добавить к ключу этой сущности дополнительный атрибут «Дата возврата». Этот атрибут, называемый дискриминатором, или частичным ключом, должен уникальным образом идентифицировать слабую сущность.

- Дискриминатор – атрибут слабого класса сущностей, идентифицирующий экземпляр этой сущности, которая должна быть обязательно связана с конкретным экземпляром сущности сильного класса.

Пример рекурсивной связи

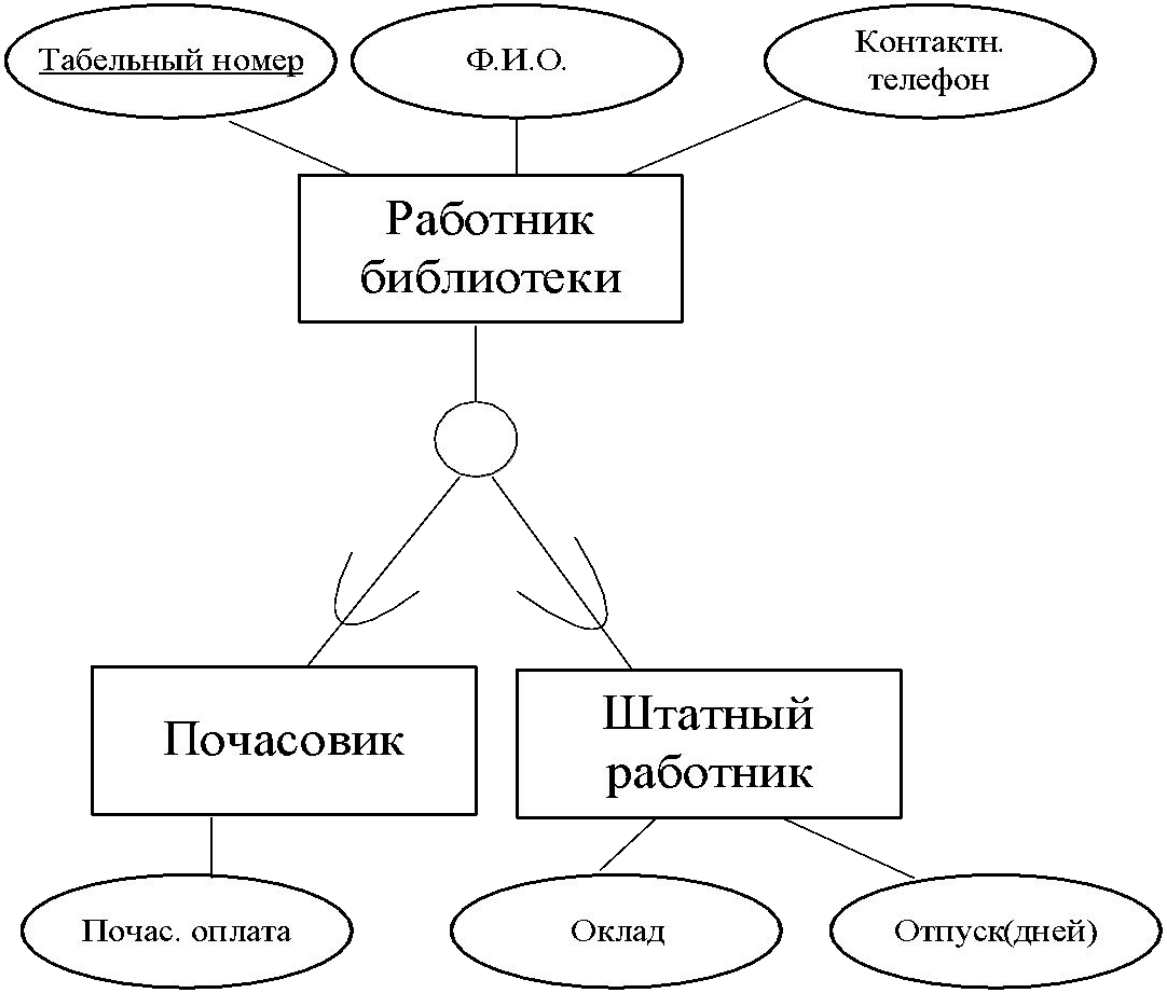


Элементы расширенной ER-модели

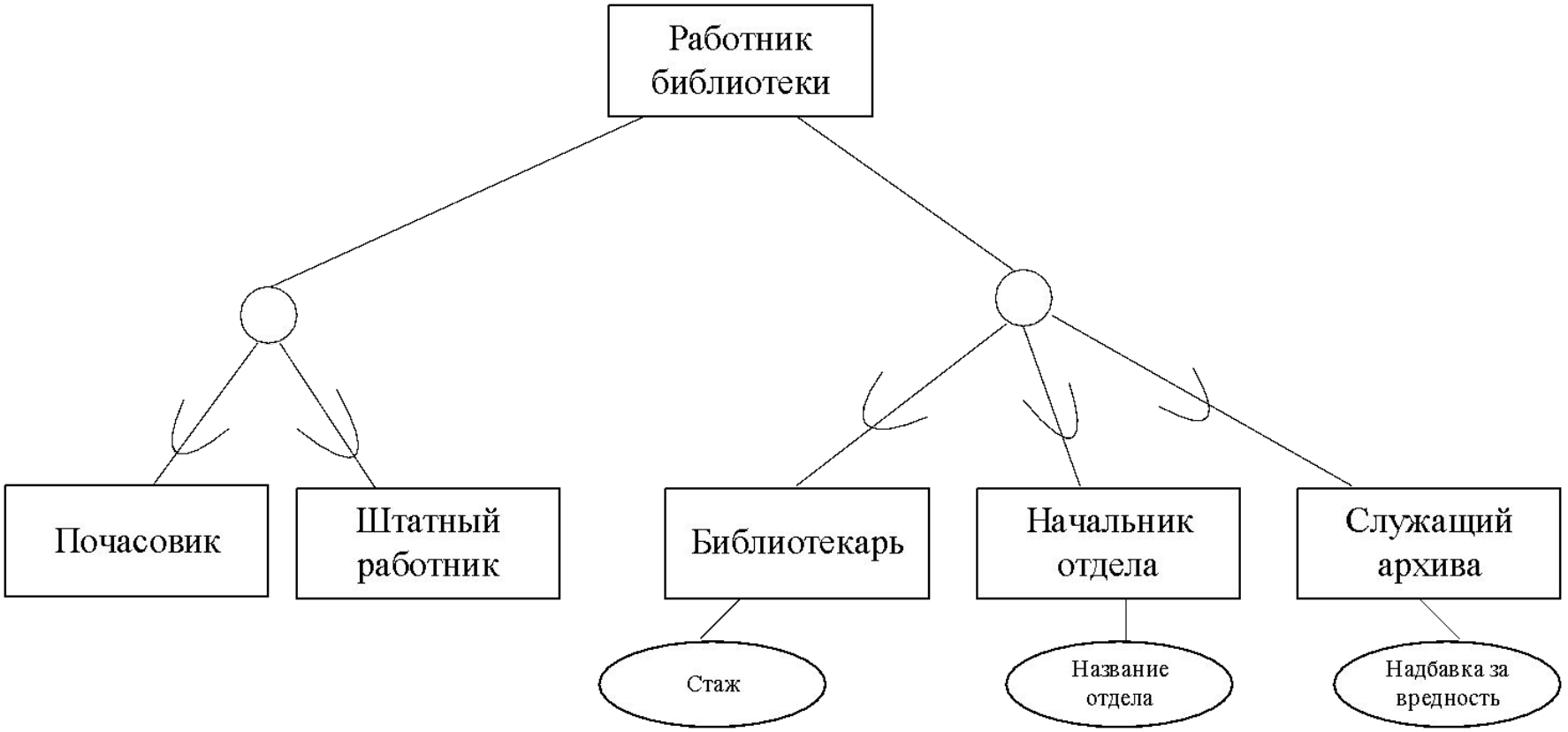
- Специализация – создание одного или более подклассов некоторого класса сущностей, причем подклассы должны иметь собственные атрибуты.

- Специализация является одним из видов образования иерархии классов.

Пример специализации



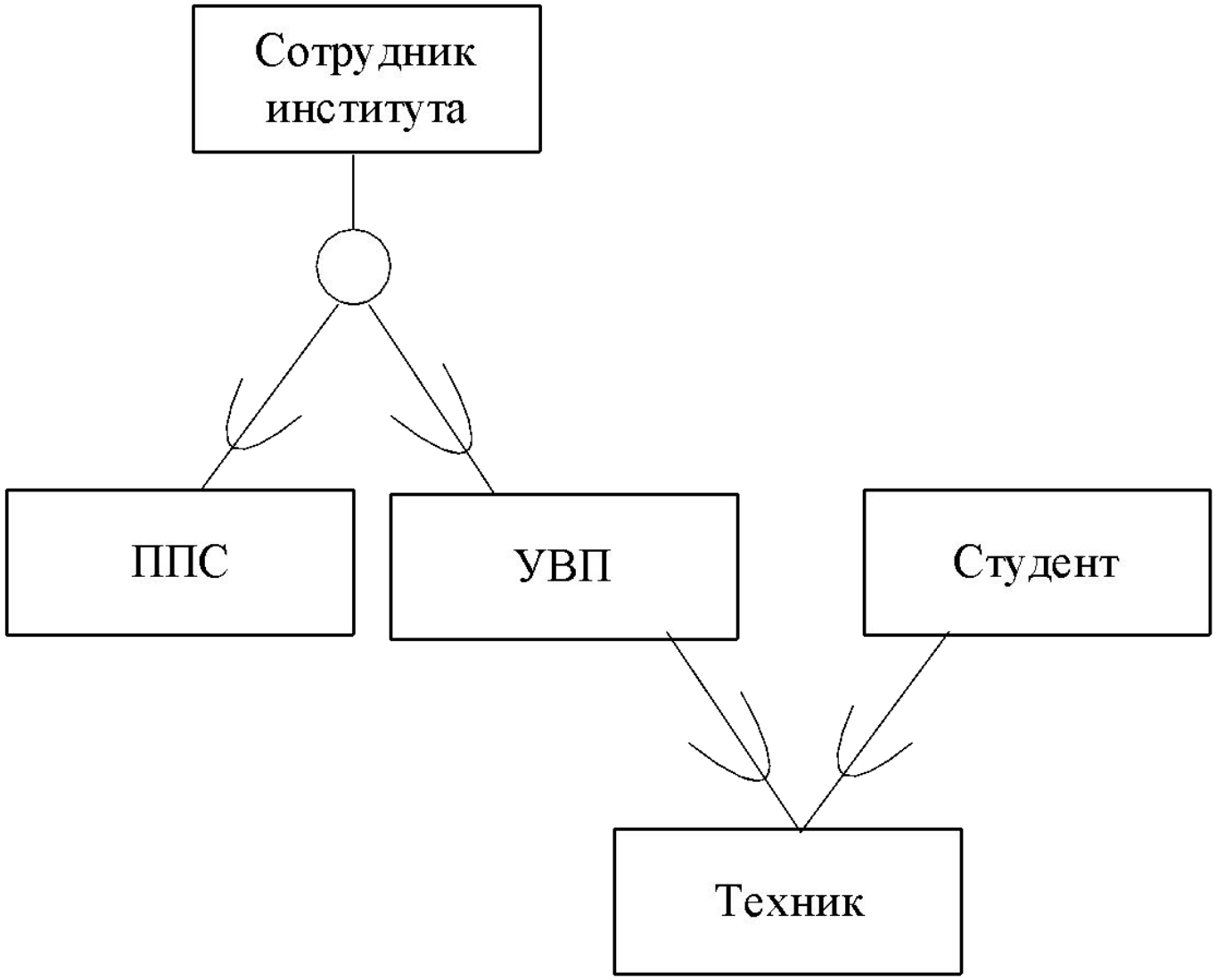
Пример нескольких специализирующих иерархий



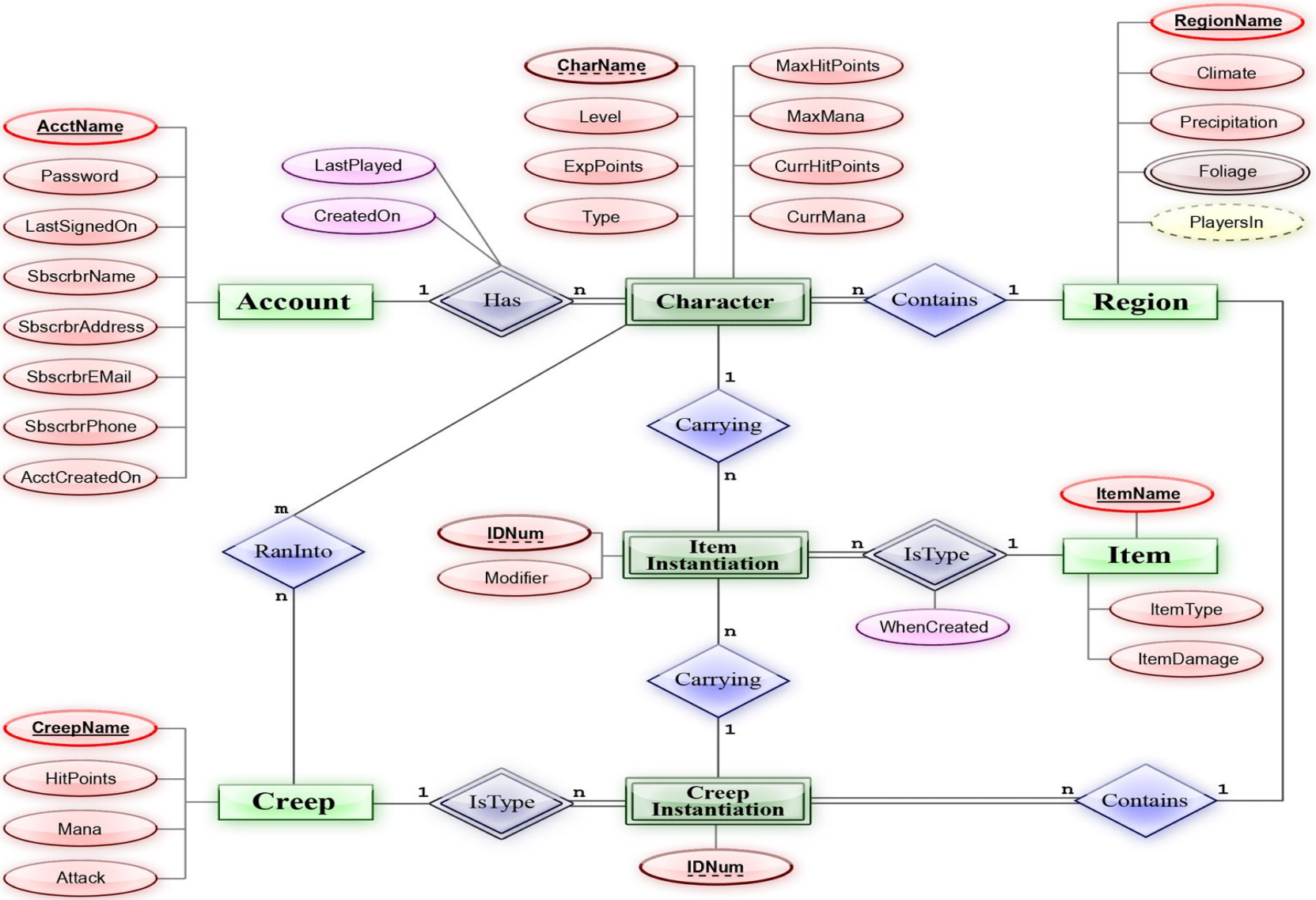
Элементы расширенной ER-модели

- Генерализация – создание суперкласса из общих атрибутов коллекции подклассов.
- Множественное наследование – ситуация, в которой класс сущностей участвует более чем в одной специализирующей иерархии, и следовательно, относится более чем к одному суперклассу.

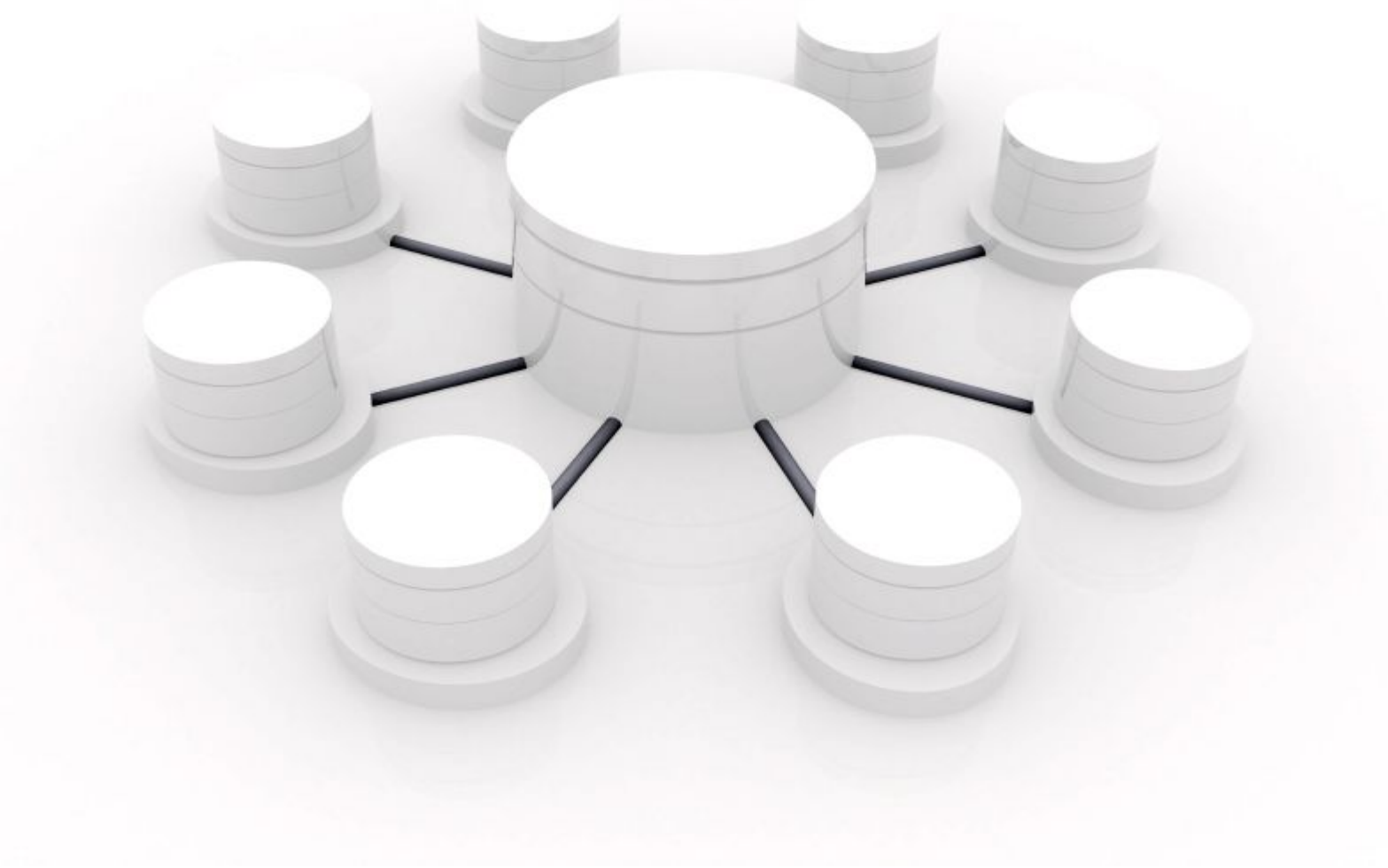
Пример множественного наследования



Автоматизированная ER-модель с использованием нотации П. Чена



Ранние даталогические модели данных: иерархическая и сетевая



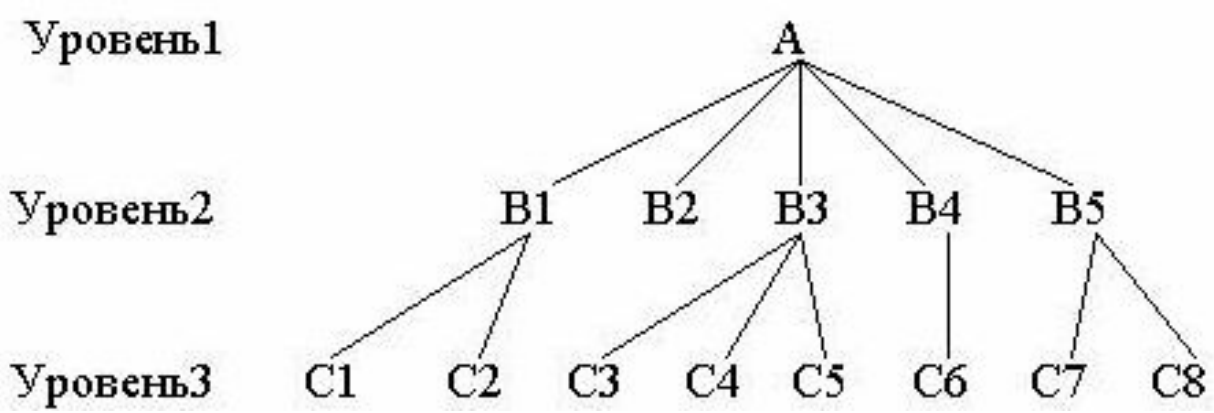
Иерархическая модель данных

Модель данных иерархическая (Hierarchical Data Model)

– это модель данных, в основе которой используется иерархическая древовидная структура данных.

Эта модель появилась первой среди всех даталогических моделей, именно ее поддерживает первая из зарегистрированных промышленных СУБД IMS фирмы IBM

Графическое изображение иерархической структуры



- Иерархическая модель данных строится **по принципу иерархии типов объектов**, то есть один тип объекта является главным, а остальные, находящиеся на низших уровнях иерархии, подчиненные. **Между главным и подчиненными объектами устанавливается взаимосвязь «один ко многим».**
- Узлы и ветви модели образуют иерархическую древовидную структуру. Узел является совокупностью атрибутов, описывающих объект. Наивысший в иерархии узел называется корневым (это главный тип объекта). Корневой узел находится на первом уровне. Зависимые узлы (подчиненные типы объектов) находятся на втором, третьем и т. д. уровнях.

СУБД, поддерживающая иерархическую модель данных.

- Иерархическая модель данных активно использовалась во многих СУБД на платформе мейнфреймов до появления реляционных систем.
- Наиболее известным ее представителем является СУБД ***I(D)MS (Information DataBase Management System)*** ***компании IBM Corp.,*** первая версия которой была разработана в конце 60-х гг. Система IMS эксплуатируется до настоящего времени.

Пример иерархической структуры базы данных

Основные элементы иерархической модели данных

- 1) поле;
- 2) сегмент;
- 3) физическая запись БД;
- 4) база данных.

- Поле - это минимальная неделимая единица данных, доступная пользователю в среде иерархической СУБД

- Сегмент (необходимо различать тип сегмента и экземпляр сегмента) – поименованная совокупность полей в иерархической модели (т.е. тип записи, а экземпляр сегмента – это запись с конкретными значениями)

Понятие физической записи БД:

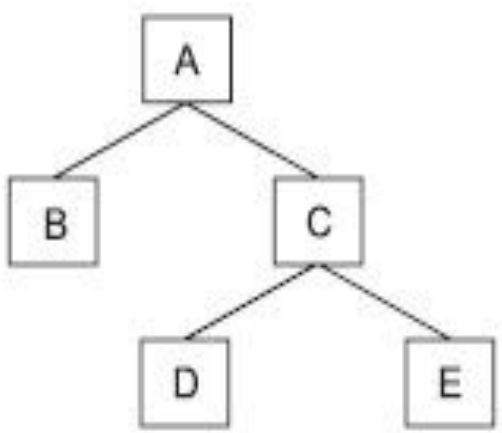
Сегменты объединены в ориентированный древовидный граф:

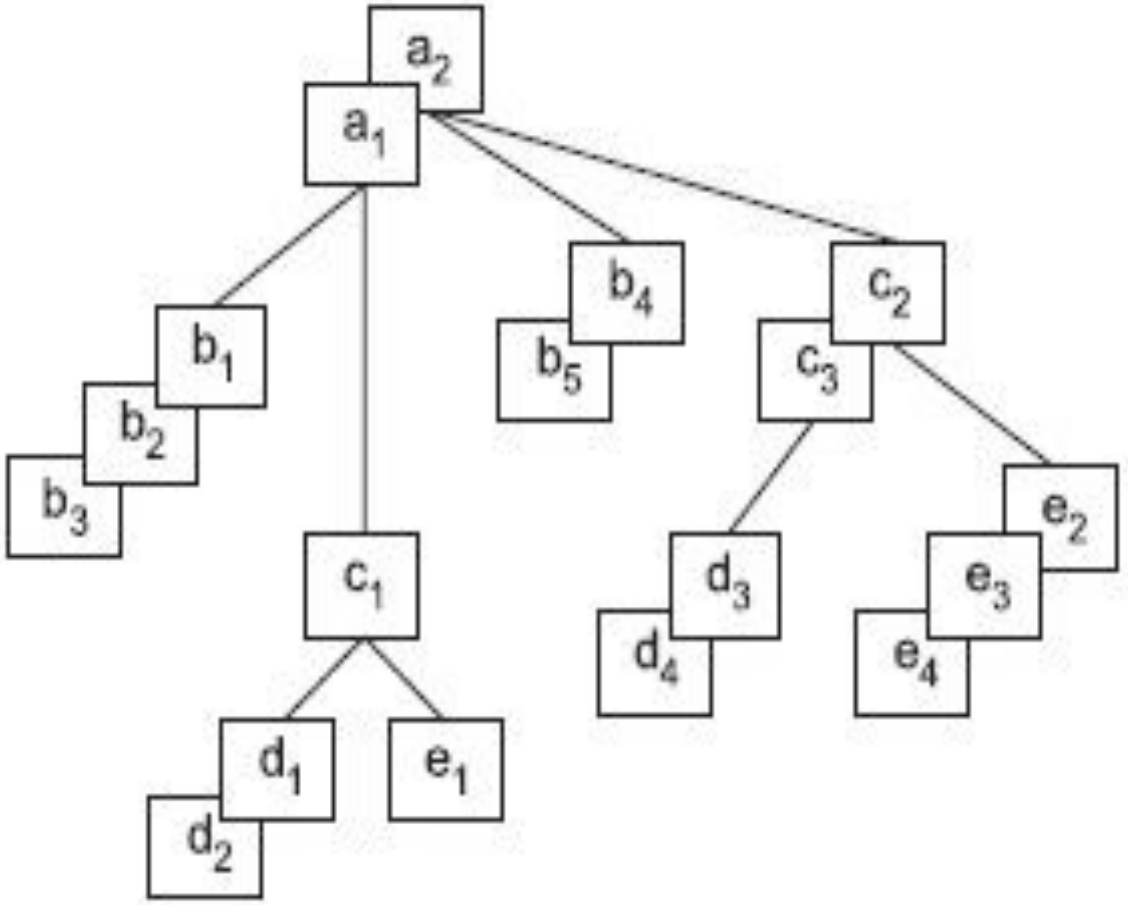
- Физической записью БД называют каждое дерево в рамках модели,

  а совокупность всех деревьев является БАЗОЙ ДАННЫХ

Пример иерархической структуры базы данных

a1 b1 b2 b3 c1 d1 d2 e1
a2 b4 b5 c2 c3 d3 d4 e2 e3 e4

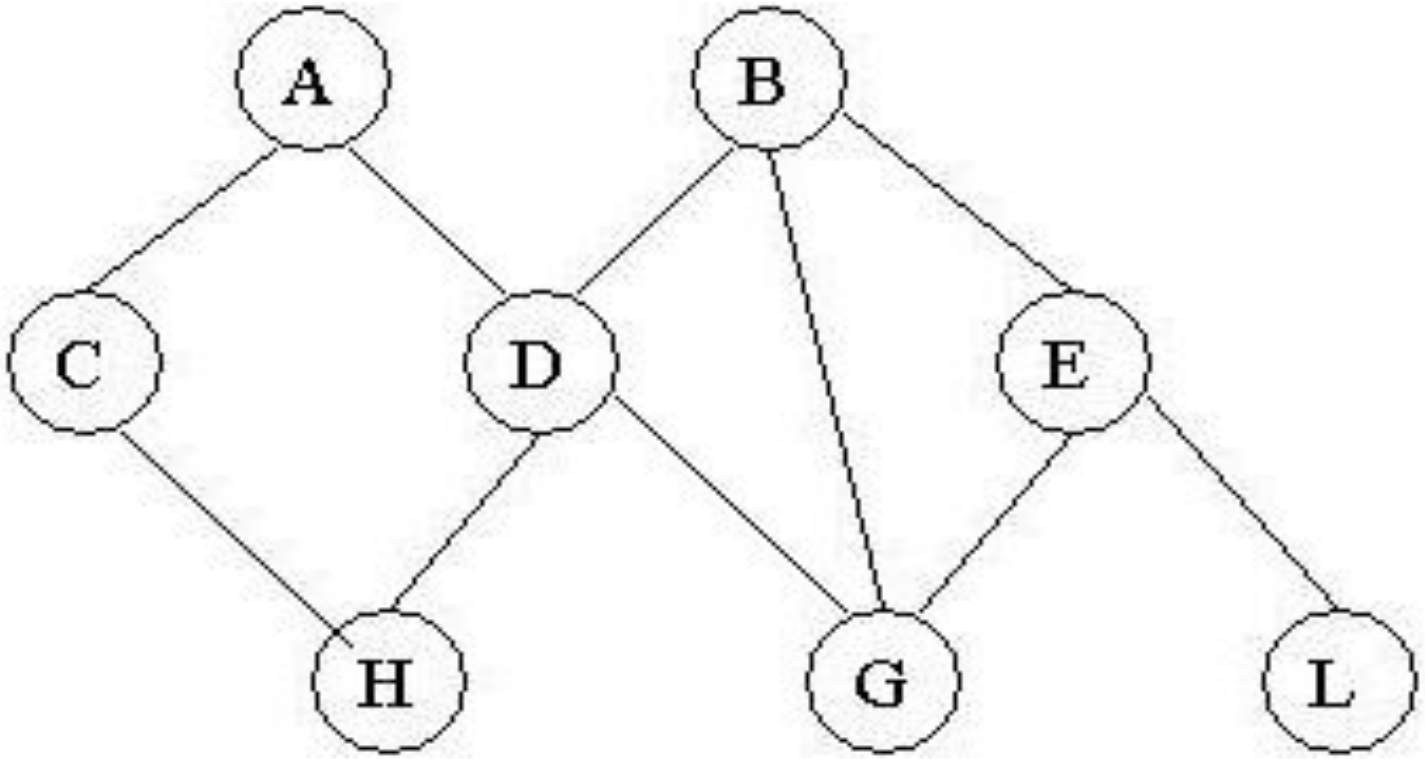




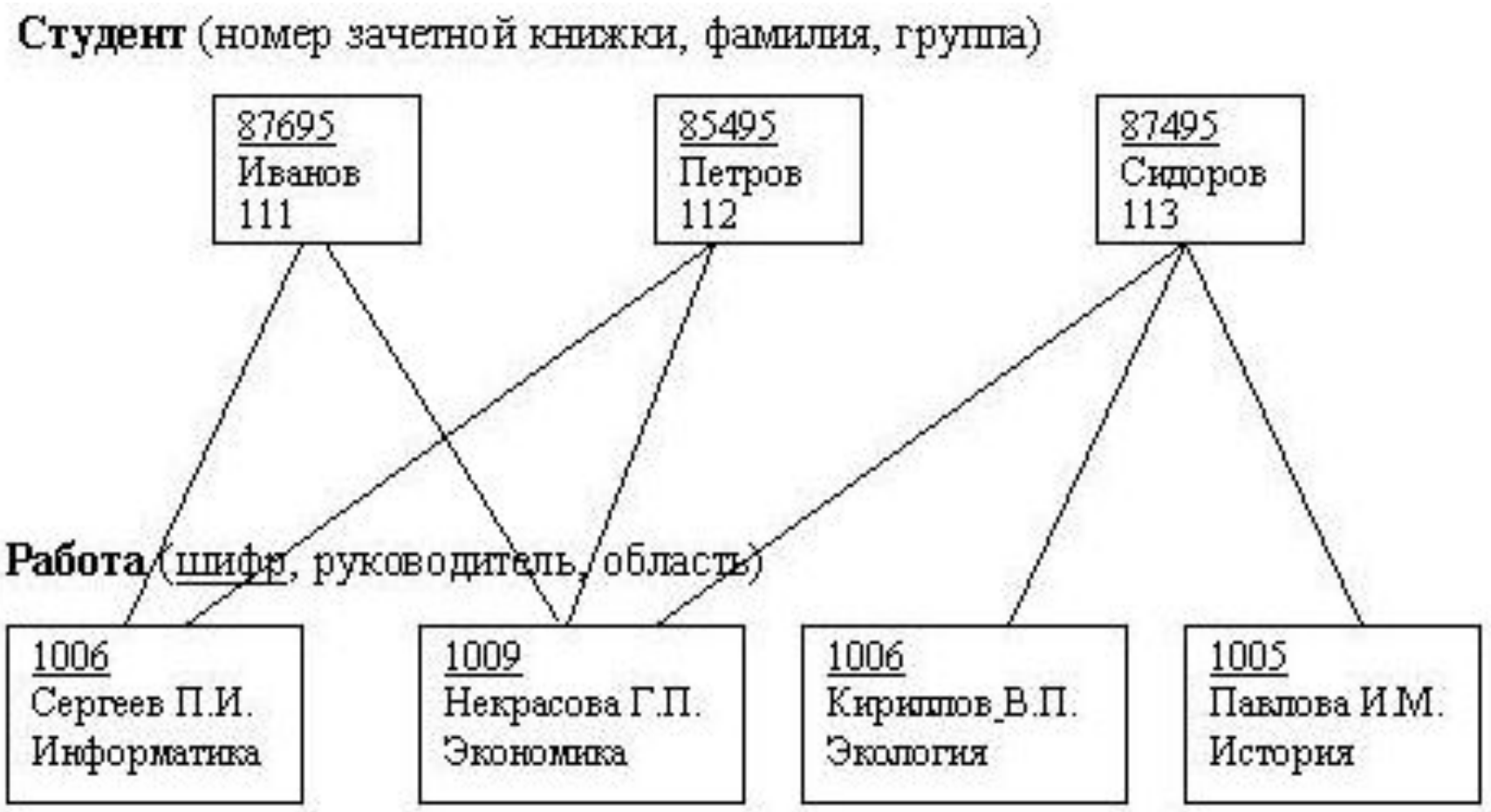
Модель данных сетевая
(Network Data Model)

- – это модель, допустимые структуры данных в которой могут быть представлены в виде *графа общего вида.*
- Вершинами такого графа могут являться данные различных типов — от атомарных элементов данных до записей сложной структуры. Дуги графа представляют *связи* между этими данными.
- Сетевую модель данных обычно ассоциируют с деятельностью организации CODASYL (Conference of Data System Languages) и именем Чарльза Бахмана.

Графическое изображение сетевой структуры



Пример сетевой структуры базы данных



Основные элементы сетевой модели данных

- Элемент данных
- Агрегат данных
- Запись
- Набор данных

- Элемент данных в сетевой модели- минимальная единица информации (соответствует полю в иерархической модели).
- Агрегат данных – следующий уровень обобщения в сетевой модели:

- агрегат типа «вектор», соответствует линейному набору данных;

Адрес			
Город	Улица	Дом	Квартира

- агрегат типа «повторяющаяся группа»

Зарплата	
Месяц	Сумма

- Записью называют множество агрегатов или элементов данных, моделирующее некоторый класс объектов реального мира (понятие записи соответствует понятию «сегмент» в иерархической модели)
- Набором называют двухуровневый граф, связывающий отношением «один-ко-многим» два типа записи. Набор фактически отражает иерархическую связь между двумя типами записей.

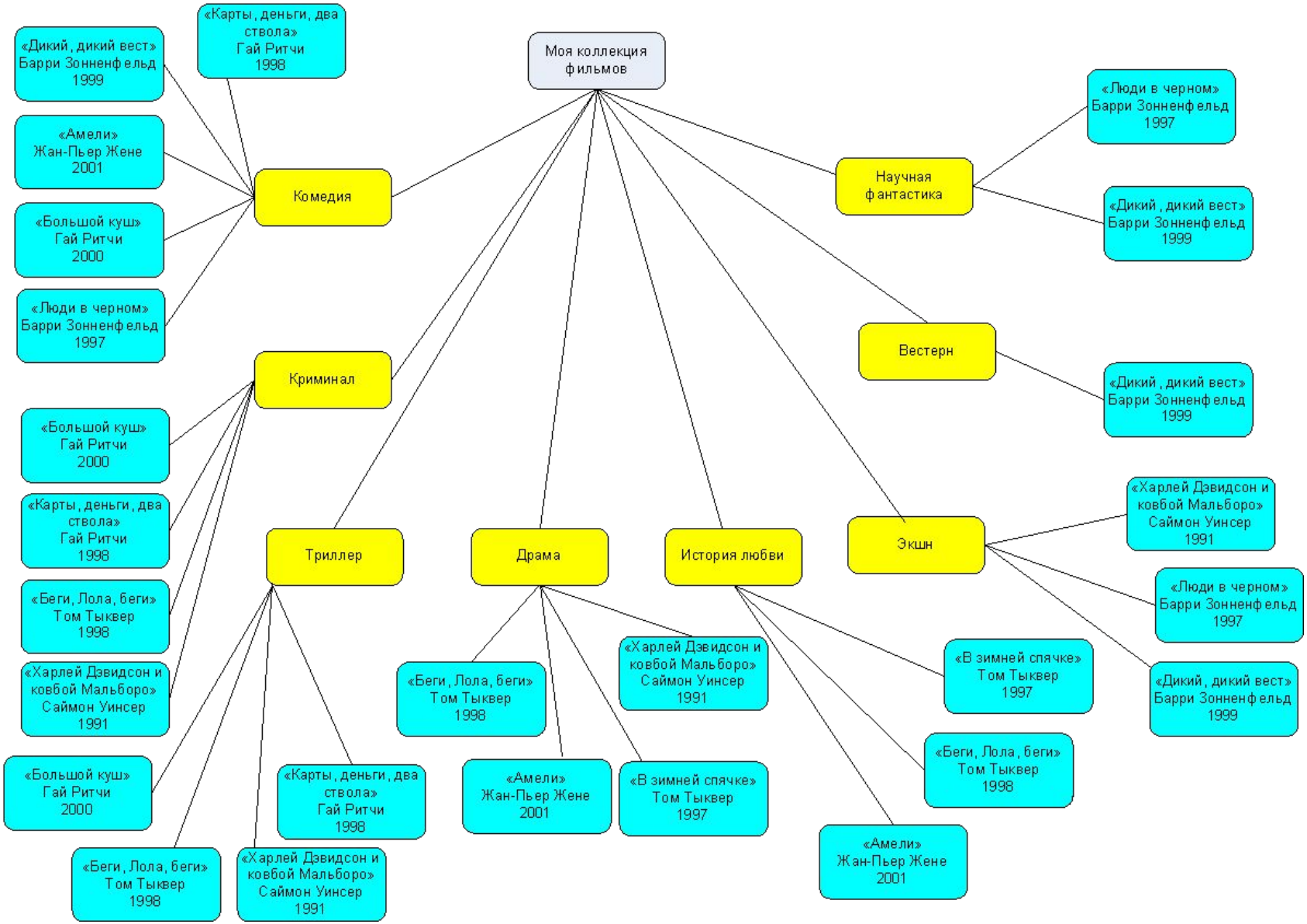
Пример сетевой структуры базы данных

Реляционная модель данных (РМД)

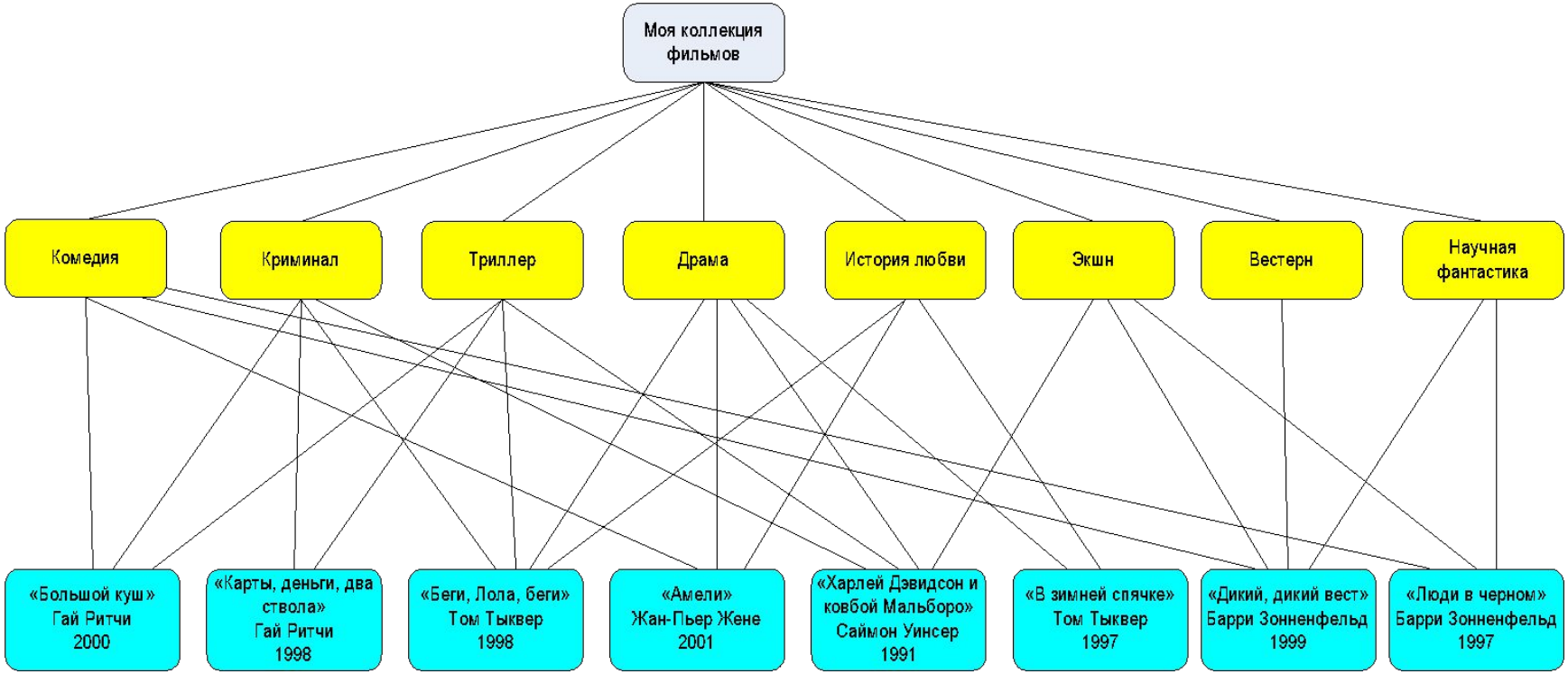
- – логическая модель данных, которая представляет множество взаимосвязанных отношений и включает следующие компоненты:
- Структурный аспект — данные в базе данных представляют собой набор отношений.
- Аспект целостности — отношения отвечают определённым условиям целостности.
- Аспект обработки — поддерживаются операторы манипулирования отношениями (реляционная алгебра, реляционные исчисления).

ПРИМЕРЫ

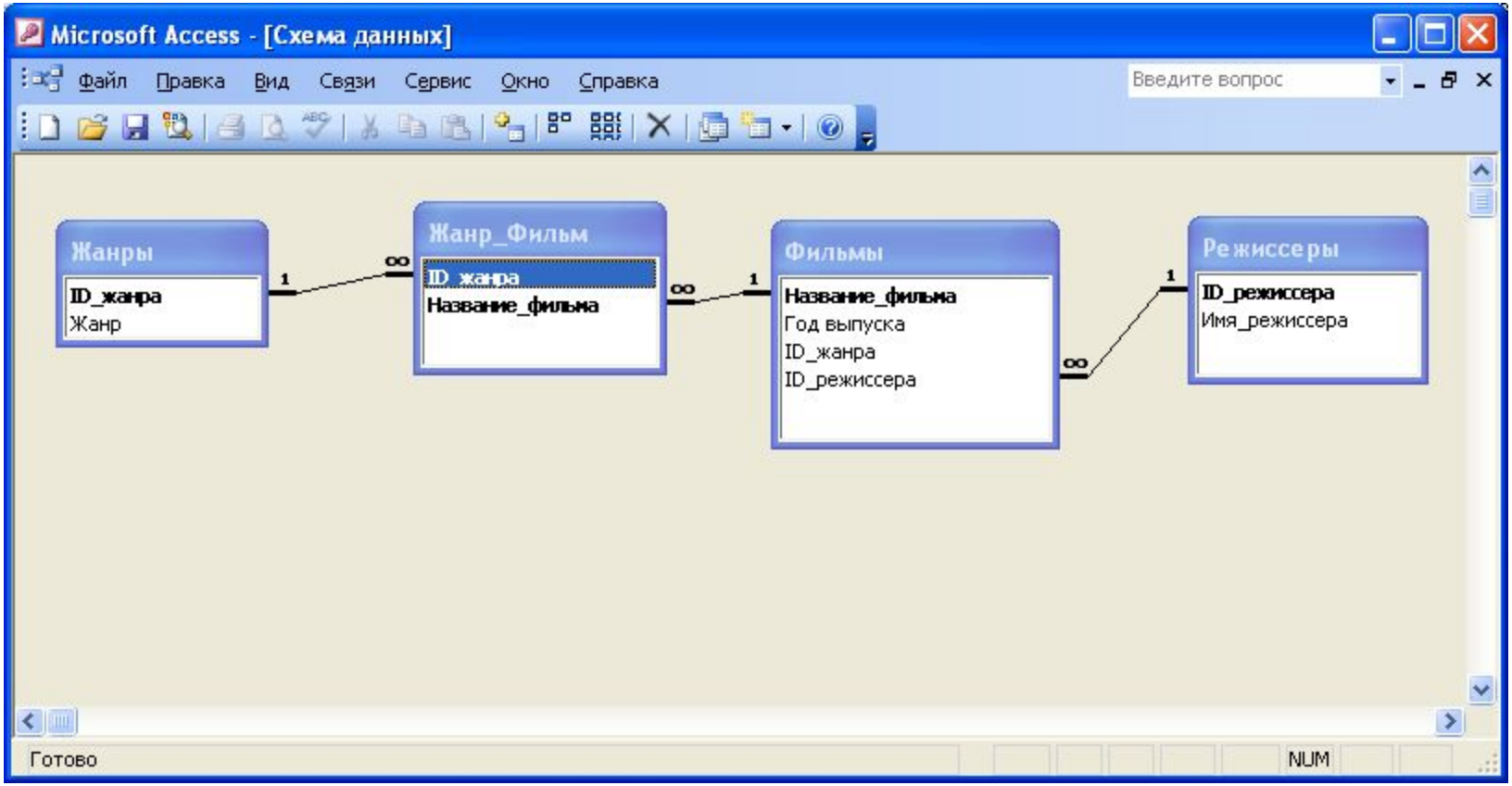
Иерархическая модель БД «Фильмотека»



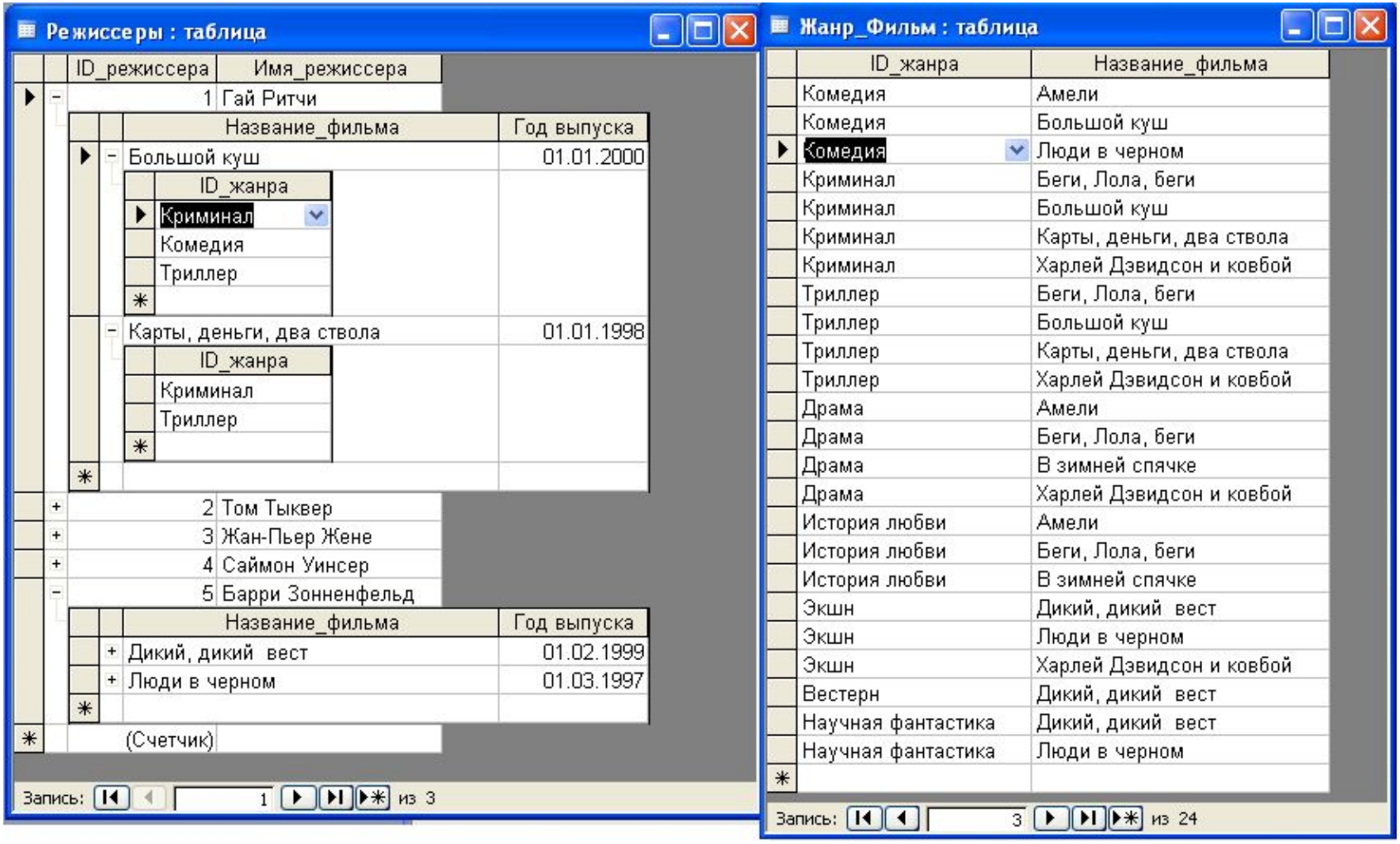
Сетевая модель БД «Фильмотека»



Реляционная модель БД «Фильмотека»



Заполненные таблицы в реляционной модели БД «Фильмотека»



Режиссеры : таблица

ID_режиссера	Имя_режиссера
1	Гай Ритчи

Название_фильма	Год выпуска
Большой куш	01.01.2000

ID_жанра
Криминал
Комедия
Триллер

Название_фильма	Год выпуска
Карты, деньги, два ствола	01.01.1998

ID_жанра
Криминал
Триллер

ID_режиссера	Имя_режиссера
2	Том Тыквер
3	Жан-Пьер Жене
4	Саймон Уинсер
5	Барри Зонненфельд

Название_фильма	Год выпуска
Дикий, дикий вест	01.02.1999
Люди в черном	01.03.1997

(Счетчик)

Запись: 1 из 3

Жанр_Фильм : таблица

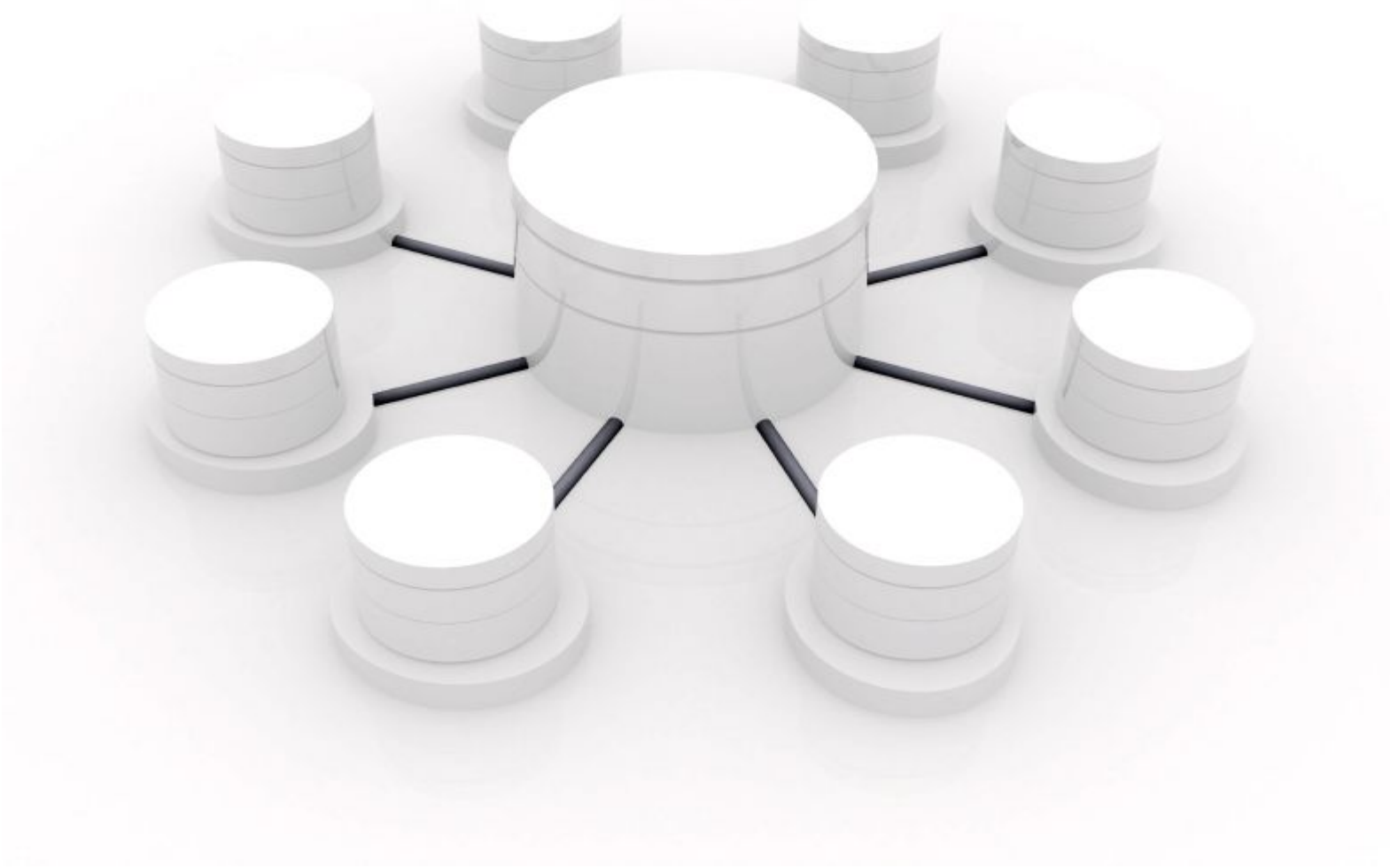
ID_жанра	Название_фильма
Комедия	Амели
Комедия	Большой куш
Комедия	Люди в черном
Криминал	Беги, Лола, беги
Криминал	Большой куш
Криминал	Карты, деньги, два ствола
Криминал	Харлей Дэвидсон и ковбой
Триллер	Беги, Лола, беги
Триллер	Большой куш
Триллер	Карты, деньги, два ствола
Триллер	Харлей Дэвидсон и ковбой
Драма	Амели
Драма	Беги, Лола, беги
Драма	В зимней спячке
Драма	Харлей Дэвидсон и ковбой
История любви	Амели
История любви	Беги, Лола, беги
История любви	В зимней спячке
Экшн	Дикий, дикий вест
Экшн	Люди в черном
Экшн	Харлей Дэвидсон и ковбой
Вестерн	Дикий, дикий вест
Научная фантастика	Дикий, дикий вест
Научная фантастика	Люди в черном

Запись: 3 из 24

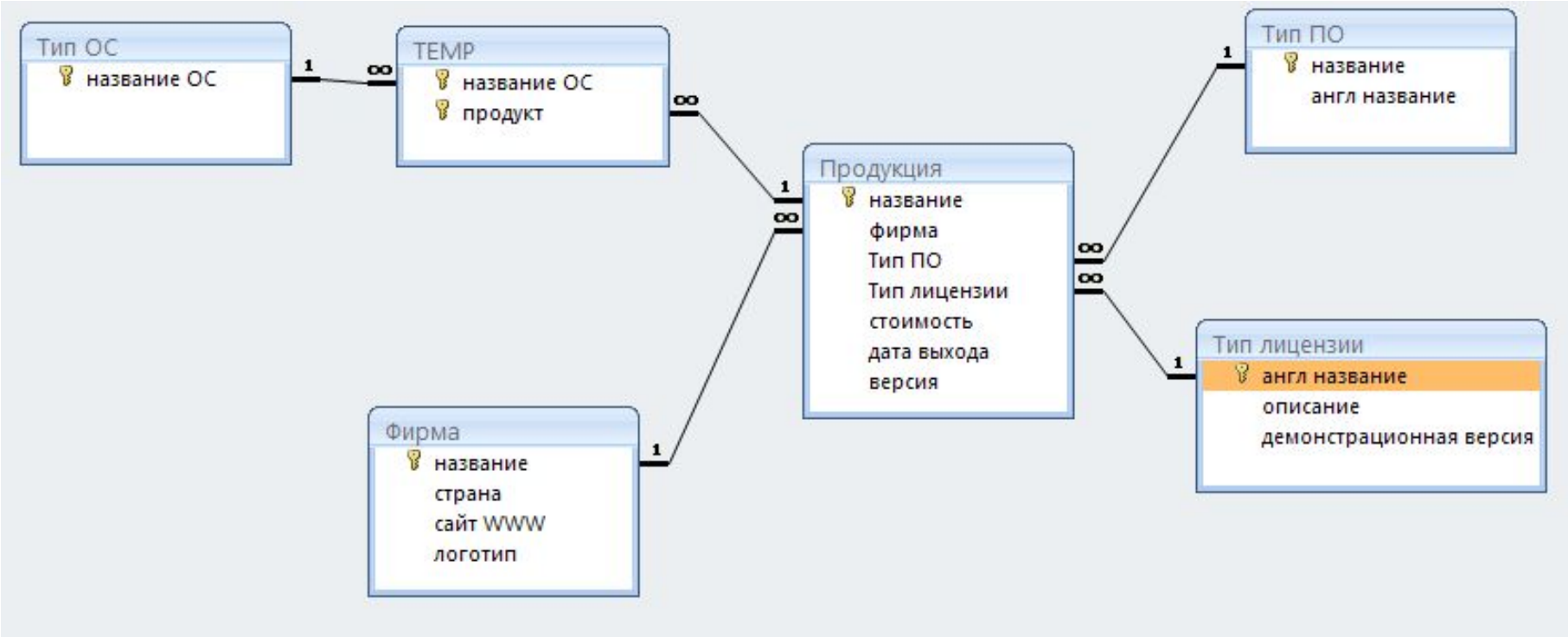
Ранжирование моделей данных по эффективности выполнения практических задач

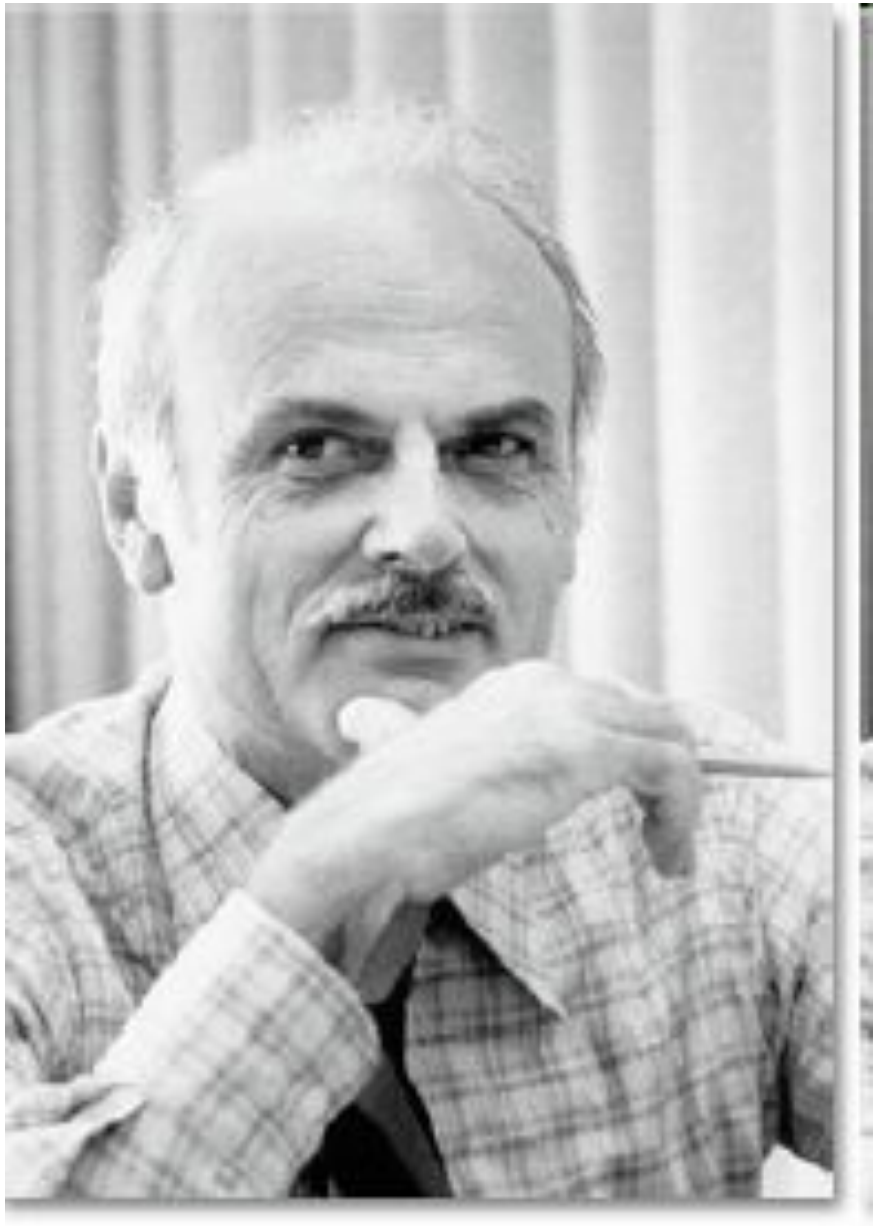
Задача	Иерархическая	Сетевая	Реляционная
Добавить фильм	3	2	1
Удалить фильм	3	2	1
Поиск фильма по названию	3	2	1
Выбор фильмов по жанру	1	3	2
Выбор фильмов определенного режиссера	3	2	1
Выбор фильмов по году выхода на экран	3	2	1
Сортировать фильмы по жанру	3	2	1
Сортировать фильмы по фамилии режиссера	3	2	1
Сортировать фильмы по году выхода на экран	3	2	1
Итог (сумма баллов)	25	19	10
			Победитель!

Общие понятия реляционного подхода к организации БД. Реляционная модель данных.



Пример реляционной БД





- Автор реляционной модели данных - доктор университета штата Мичиган (США) **Эдгар Фрэнк Кодд (Codd E.F.)**

(1923 г.- 2003 г)

Первая публикация - 1970 г.

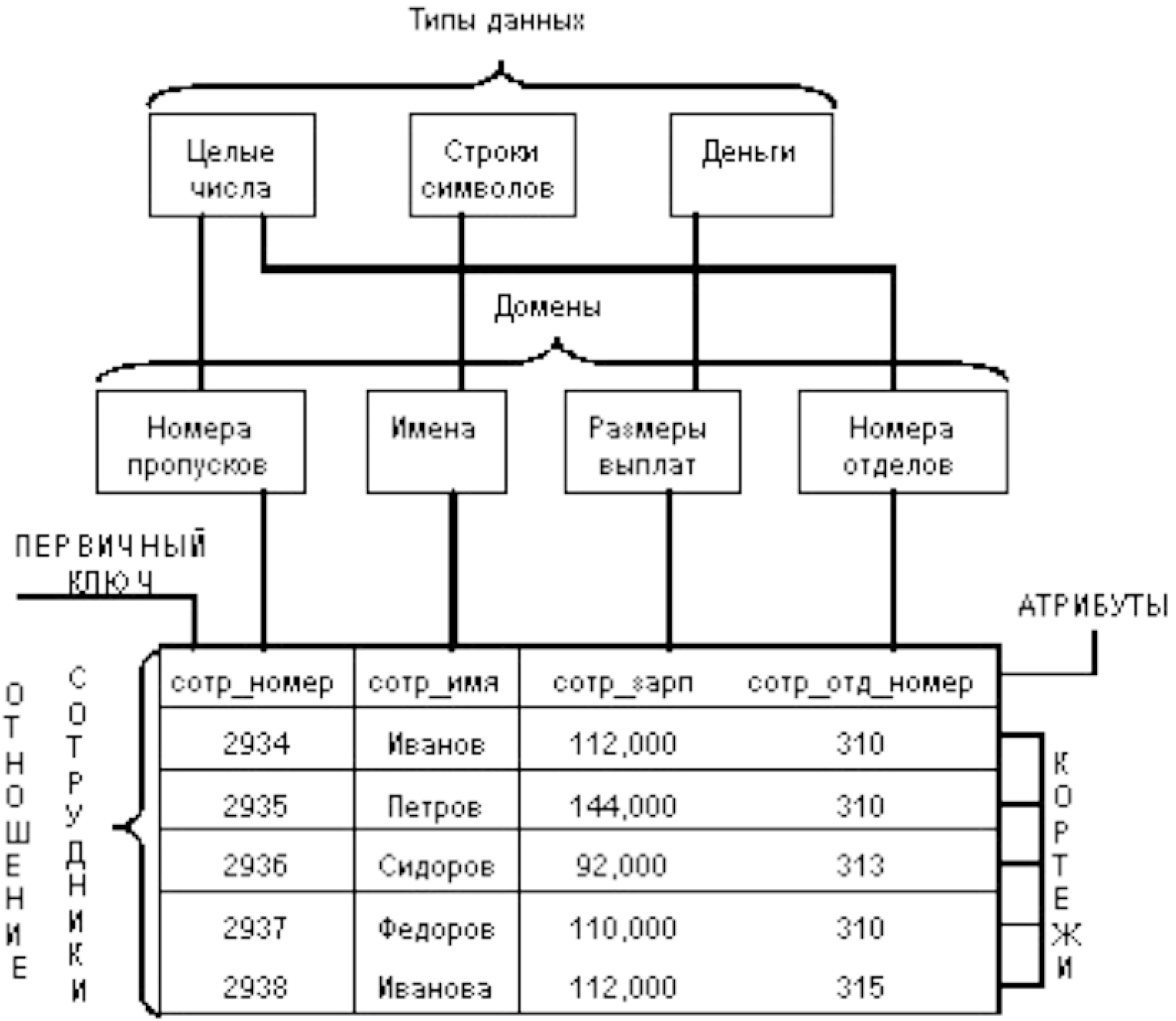
- Теоретическим базисом является

теория множеств

Основные понятия реляционной модели

- 1) отношение (таблица);
- 2) схема отношения (таблицы);
- 2) атрибут;
- 3) первичный ключ;
- 4) кортеж;
- 5) тип данных;
- 6) домен (domain).

Запишем определения основных понятий РМД


Типы данных
Целые числа
Строки символов
Деньги
Домены
Номера пропусков
Имена
Размеры выплат
Номера отделов
ПЕРВИЧНЫЙ КЛЮЧ
АТРИБУТЫ
ОТНОШЕНИЕ СОТРУДНИКИ
сотр_номер
сотр_имя
сотр_зарп
сотр_отд_номер
2934
Иванов
112,000
310
2935
Петров
144,000
310
2936
Сидоров
92,000
313
2937
Федоров
110,000
310
2938
Иванова
112,000
315
КОРТЕЖИ

Пример отношения «Film»

Title	Year	Length	Filmtype
Звездные войны	1977	124	color
Вспомнить всё	1997	150	color
Мост Ватерлоо	1935	95	Black_white
Побег из Шоушенка	1999	145	color
Люди в черном	1997	106	color
Казино «Рояль»	2006	137	color

Пример схемы отношения:

Film (Title, Year, Length, Filmtype)

Пример схемы БД:

Film (Title, Year, Length, Filmtype);

Studio (Name, Address, presN);

Producer (Name, Address, Sert, NetWorth);

Фундаментальные свойства отношений (таблиц)

1. Атомарность значений атрибутов (каждый атрибут в отношении имеет уникальное имя и простой тип)

- Значения всех атрибутов являются атомарными. Это следует из определения **домена** как потенциального множества значений простого типа данных, т.е. среди значений домена не могут содержаться множества значений (отношения).

  Принято говорить, что в реляционных базах данных допускаются только нормализованные отношения или отношения, представленные в первой нормальной форме.

Фундаментальные свойства отношений

2. Отсутствие кортежей-дубликатов (в таблице нет двух одинаковых строк)

Это свойство, что отношения не содержат кортежей-дубликатов, следует из определения отношения (таблицы) как **множества** кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов.

Фундаментальные свойства отношений

3. Отсутствие упорядоченности кортежей (порядок строк в таблице произвольный)

Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных.

Фундаментальные свойства отношений

4. Отсутствие упорядоченности атрибутов (таблица имеет столбцы, соответствующие атрибутам отношения, и порядок следования атрибутов при обращении к таблице не принципиален)

Атрибуты отношений (таблиц) не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута, имя домена}. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута.

Это свойство теоретически позволяет, например, модифицировать схемы существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов.

Фундаментальные свойства отношений

4. Изменяемость отношений.

Реляционная модель данных рассматривает отношение как структурный тип. Тип определяется схемой отношения.

Все кортежи – знаки типа – удовлетворяют одной и той же схеме.

Тело отношения может изменяться во времени:

- Отдельные кортежи могут добавляться или удаляться.
- Могут изменяться значения атрибутов в существующих кортежах.

Поэтому можно говорить об экземпляре (текущем значении) отношения с заданной схемой

Необходимо знать определения следующих понятий реляционной модели:

- **отношение;**
- **атрибут;**
- **первичный ключ;**
- **кортеж;**
- **домен;**
- **тип данных;**
- **схема отношения;**
- **схема базы данных;**
- **степень отношения;**
- **мощность отношения (кардинальное число).**

Необходимо знать фундаментальные свойства отношений и их трактовку:

1. **Атомарность значений атрибутов**
2. **Отсутствие кортежей-дубликатов**
3. **Отсутствие упорядоченности кортежей**
4. **Отсутствие упорядоченности атрибутов**
5. **Изменяемость отношений (таблиц)**

Соответствие компонентов модели «сущность-связь» и реляционной модели

ER-модель	Реляционная модель
Сущность	Отношение
Атрибут	Атрибут
Ключевой атрибут	Первичный ключ
Экземпляр сущности	Кортеж
Тип атрибута	Домен
Связь	Внешний ключ

Правила перехода от ER-модели к реляционной модели данных

- **1. Каждой сущности ставится в соответствие отношение (таблица) реляционной модели данных.**
- **2. Каждый атрибут сущности становится атрибутом отношения. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (допустимость *Null-значений*).**
- **3. Ключевой атрибут (набор атрибутов) сущности становится *первичным ключом* (*primary key*) соответствующего отношения. Атрибуты, входящие в первичный ключ, получают автоматически свойство обязательности (*Not Null*).**

Правила перехода от ER-модели к реляционной модели данных

- **4. В каждое отношение, соответствующее подчиненной сущности, добавляются атрибуты основной сущности, являющийся первичным ключом основной сущности. В отношении, соответствующем подчиненной сущности, этот набор атрибутов становится внешним ключом (foreign key)**
- **5. Для моделирования необязательного типа связи на физическом уровне у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений (признак Null). При обязательном типе связей атрибуты, входящие во внешний ключ, получают свойство запрета неопределенных значений (Not Null).**

Правила перехода от ER-модели к реляционной модели данных

- **6. Разрешение связей типа многие-ко-многим (в реляционной модели допустимо использование только 1:M и 1:1) производится введением дополнительного связующего отношения, которое связано с каждым исходным отношением связью один-ко-многим, атрибутами этого отношения являются первичные ключи связываемых отношений.**
- **7. Для отражения иерархии классов сущностей при переходе к реляционной модели возможны несколько вариантов представления:**
  - **1) можно создать одно отношение для всех подклассов одного суперкласса. В него включают все атрибуты всех классов.**
  - **2) для каждого класса и для суперкласса создаются свои отдельные отношения.**

Инфологическая модель БД «Библиотека»

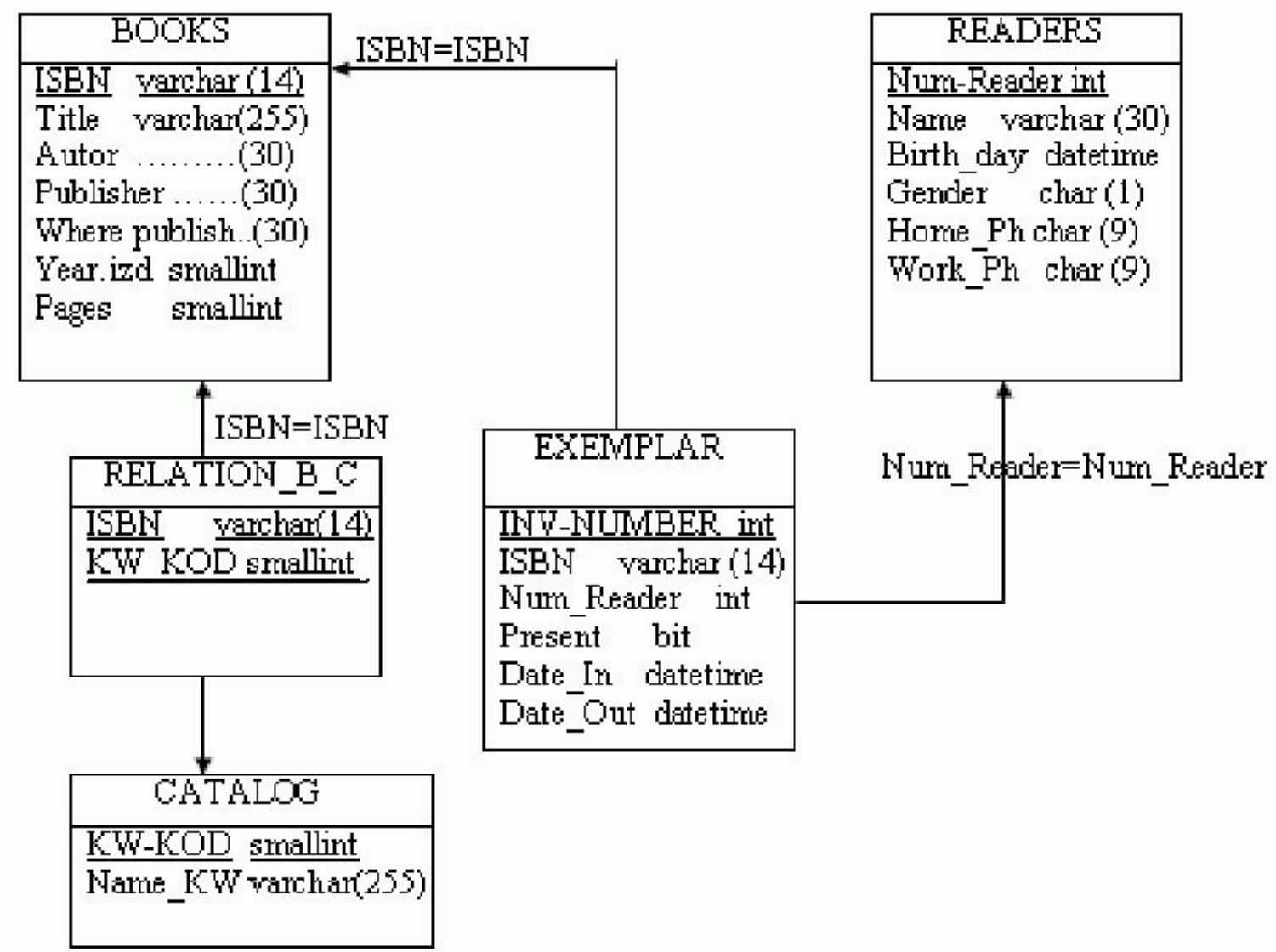
Таблица 1. Имена и описания классов сущностей БД «Библиотека»

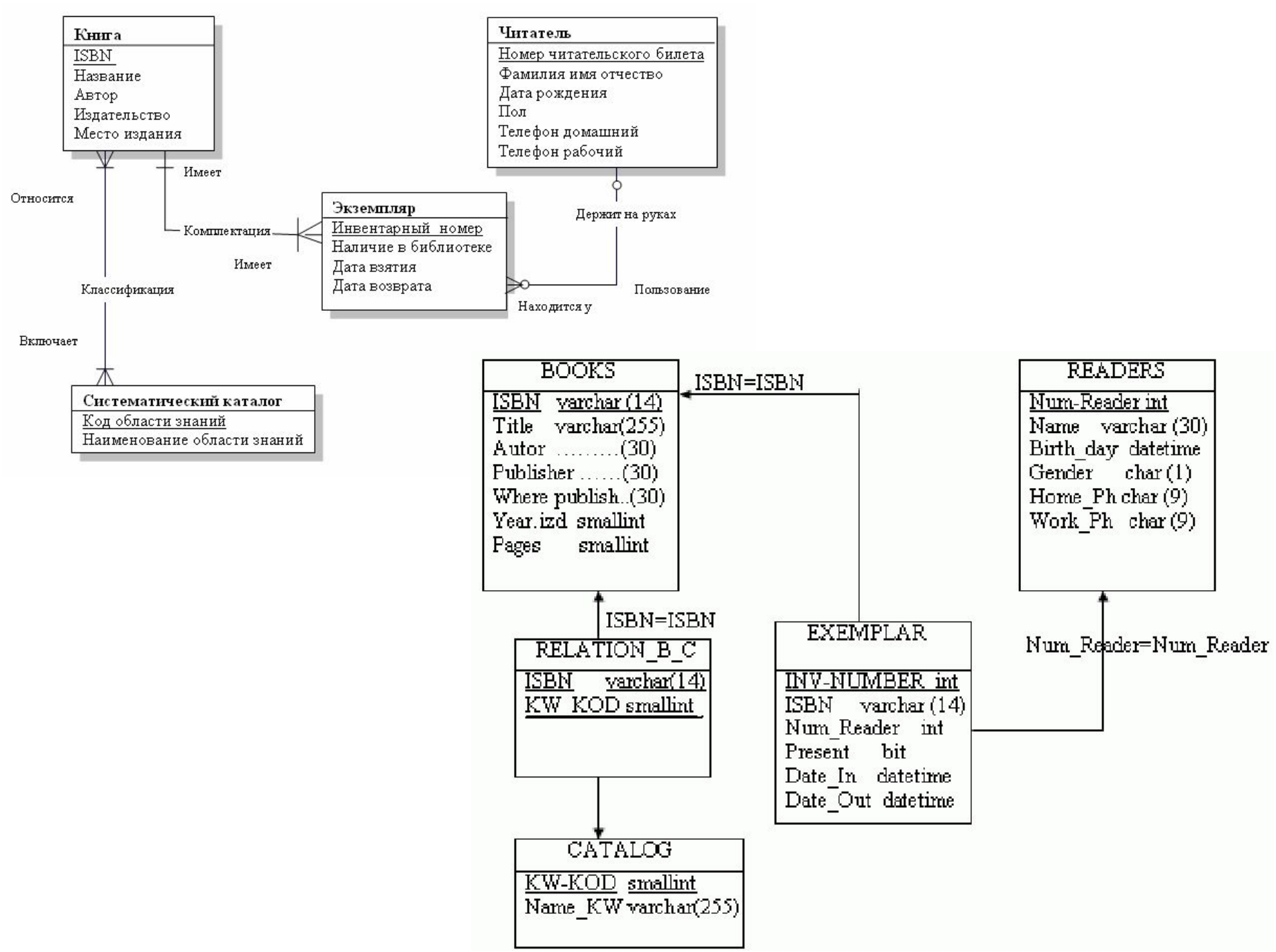
Сущность	**Описание**
Читатель	Пользователь библиотеки
Книга	Объект библиотечного дела
Экземпляр книги	Один из множества физически существующих объектов, который может быть выдан читателю
.....	.....

Таблица 2. Имена и описания некоторых атрибутов сущностей БД «Библиотека»

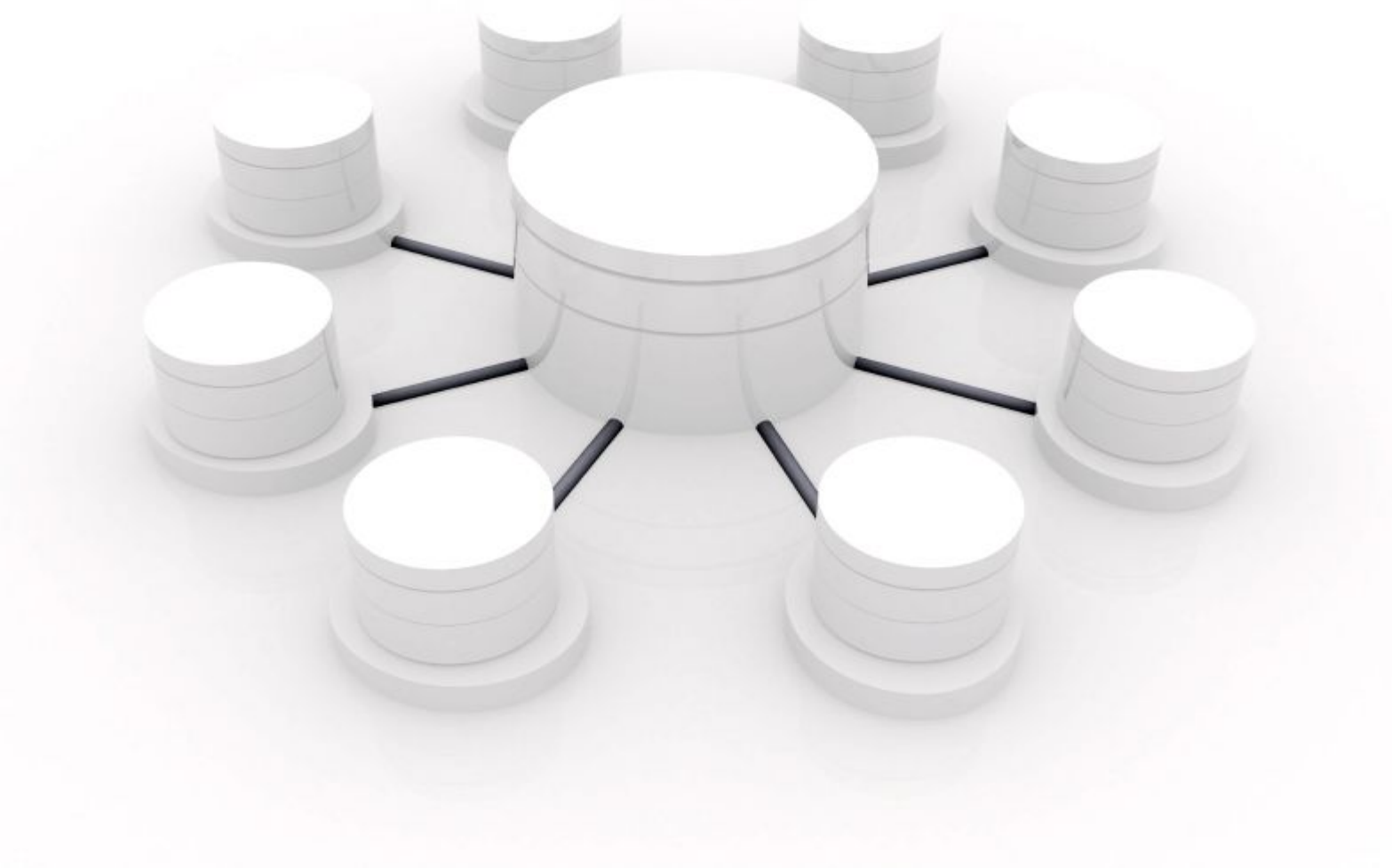
Атрибут	**Тип**	**Область значений**	**Описание**
Ф.И.О.	string	40 символов	Фамилия, имя, отчество
ISBN	string	15 символов	Уникальный библиотечный шифр
Дата_инвентар	Date	Месяц, день, год	Дата инвентаризации
.....	.....	.....	.....

Реляционная модель БД «Библиотека»

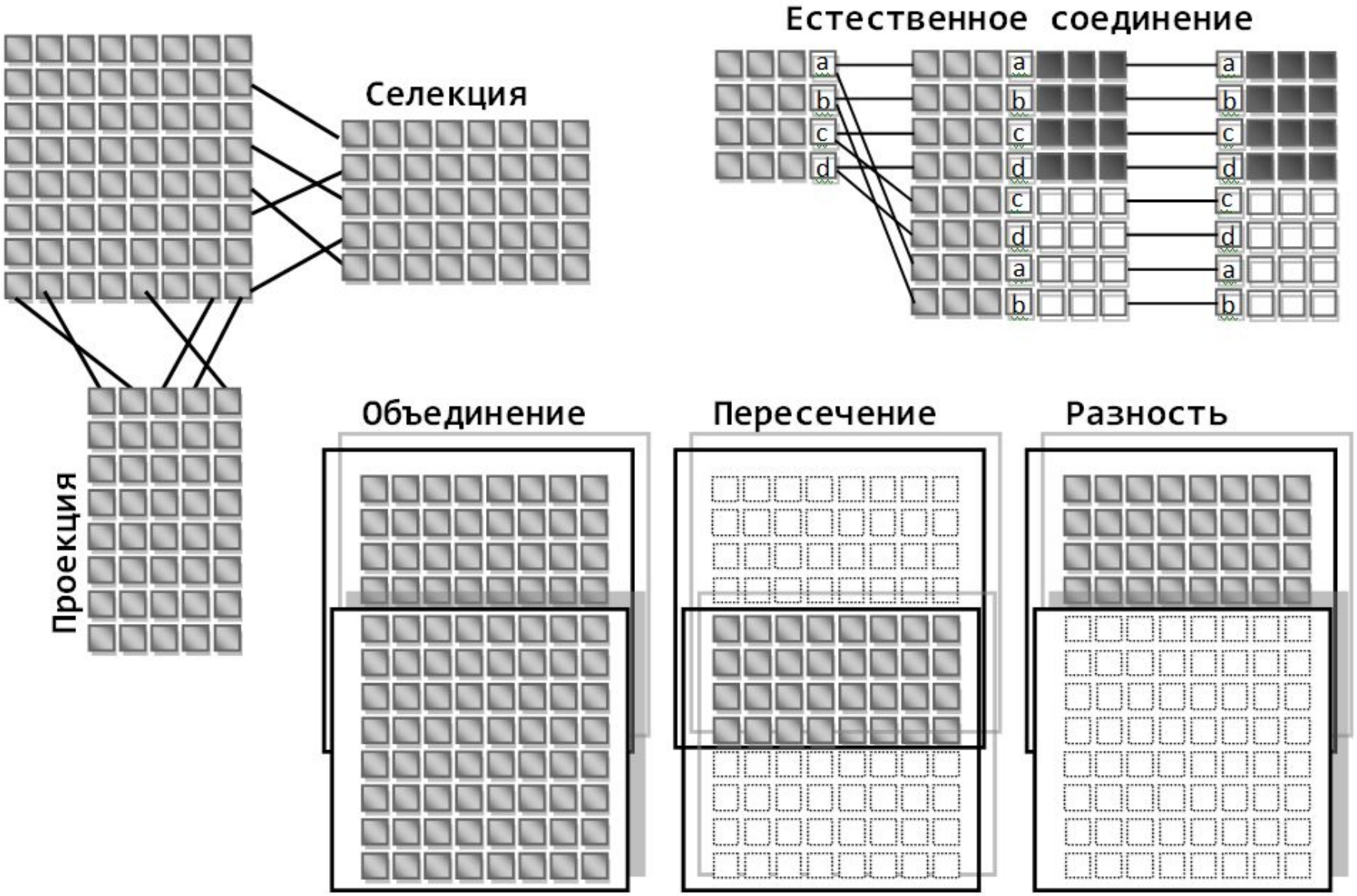



Книга
ISBN
Название
Автор
Издательство
Место издания
Читатель
Номер читательского билета
Фамилия имя отчество
Дата рождения
Пол
Телефон домашний
Телефон рабочий
Имеет
Относится
Экземпляр
Инвентарный номер
Наличие в библиотеке
Дата взятия
Дата возврата
Держит на руках
Комплектация
Имеет
Классификация
Находится у
Пользование
Включает
Систематический каталог
Код области знаний
Наименование области знаний
BOOKS
ISBN varchar (14)
Title varchar(255)
Autor .....(30)
Publisher .....(30)
Where publish..(30)
Year.izd smallint
Pages smallint
ISBN=ISBN
READERS
Num-Reader int
Name varchar (30)
Birth_day datetime
Gender char (1)
Home_Ph char (9)
Work_Ph char (9)
ISBN=ISBN
RELATION_B_C
ISBN varchar(14)
KW_KOD smallint
EXEMPLAR
INV-NUMBER int
ISBN varchar (14)
Num_Reader int
Present bit
Date_In datetime
Date_Out datetime
Num_Reader=Num_Reader
CATALOG
KW-KOD smallint
Name_KW varchar(255)

Язык запросов SQL. Команда SELECT



Реляционная алгебра



Реляционная алгебра – теоретический базис SQL

SQL- Structured Query Language — язык структурированных запросов- является комплексным языком баз данных.

- **SQL** – это гибкий язык, который можно использовать самыми разными способами.
- Он является самым распространенным инструментом, используемым для связи с реляционной базой данных.

- Этот язык не является **процедурным**, как FORTRAN, Basic, C, COBOL, Pascal и Java. Чтобы решить задачу с помощью одного из этих процедурных языков, приходится писать процедуру, которая выполняет одну за другой указанные операции, пока выполнение задачи не будет закончено. Процедура может быть линейной последовательностью или содержать ветвление, но в любом случае программист указывает порядок выполнения.
- Иными словами, SQL является **непроцедурным** языком. Чтобы с его помощью решить задачу, сообщите, **что именно** вам нужно, как если бы вы говорили с джином из лампы Аладдина. И при этом не надо говорить, **каким образом** получить для вас то, что вы хотите. Система управления базами данных (СУБД) сама решит, как лучше всего выполнить ваш запрос.

ПРИМЕР (**ЧТО** НАМ НУЖНО – запрашиваем у БД)

SELECT * FROM EMPLOYEE WHERE AGE >40 OR SALARY >60000;

Этот оператор выбирает из таблицы EMPLOYEE все строки, в которых или значение столбца AGE (возраст) больше 40 или значение в столбце SALARY (зарплата) больше 60000. SQL сам знает, каким образом надо выбирать информацию. Ядро базы данных проверяет базу и принимает для себя решение, каким образом следует выполнять запрос. Все, что от вас требуется, – указать, какие данные вам нужны.

Помните:

Запрос – это вопрос, который вы задаете базе данных. Если какие-либо ее данные удовлетворяют условиям вашего запроса, то SQL передает их вам.

- в последние годы оказывалось немалое давление, чтобы дополнить SQL некоторыми процедурными возможностями.
- Поэтому теперь в составе новой версии спецификации SQL, SQL:2003, имеются такие средства процедурного языка, как блоки BEGIN, условные операторы IF, функции и процедуры.
- Благодаря этим новым средствам, можно хранить программы на сервере с тем, чтобы их могли повторно использовать многие пользователи.

И всё же …

В современных реализациях SQL отсутствуют многие простые программные конструкции, которые являются фундаментальными для большинства других языков программирования.

В приложениях для повседневной жизни, как правило, требуются хотя бы некоторые из этих конструкций, поэтому SQL на самом деле представляет собой **подъязык** данных. Даже имея дополнения, появившиеся в SQL вместе со стандартом SQL: 1999 и дополнительные расширения, добавленные в SQL:2003, все равно для создания законченного приложения необходимо использовать вместе с SQL один из программных языков, такой, например, как C.

Формы SQL

– интерактивный (Interactive);

Интерактивный SQL применяется для непосредственной работы с базой данных с целью получения результатов для последующего использования.

Вводится оператор, он сразу же выполняется, и вы видите результат (если таковой существует). В англоязычной литературе эту форму SQL называют прямой SQL – Direct SQL

Формы SQL

- интерактивный (Interactive);
- статический (Static);

Статический SQL состоит из операторов SQL, которые жестко закодированы в приложении или программном модуле. Самым распространенным видом является *встроенный SQL* (Embedded SQL), где SQL-код включается в исходный текст программы, написанной на другом языке, например на C или Pascal.

Формы SQL

- интерактивный (Interactive);
- статический (Static);
- динамический (Dynamic).

Динамический SQL (Dynamic SQL) также является частью приложения или программного модуля, но конкретный код SQL, который будет исполняться, генерируется во время выполнения, а не вводится заранее. Для этого требуется некоторое расширение статического SQL.

КАК SQL «экономит силы» программиста

SQL устраняет много работы которую вы должны были бы сделать если бы вы использовали универсальный язык программирования, например С. Чтобы сформировать реляционную базу данных на С, вам необходимо было бы начать с самого начала. Вы должны были бы определить объект - называемый таблицей, которая могла бы расти чтобы иметь любое число строк, а затем создавать постепенно процедуры для помещения значений в нее и извлечения из них.

Если бы вы захотели найти некоторые определенные строки, вам необходимо было бы выполнить по шагам процедуру, подобную следующей :

1. Рассмотрите строку таблицы.

2. Выполните проверку - является ли эта строка одной из строк которая вам нужна.

3. Если это так, сохраните ее где-нибудь пока вся таблица не будет проверена.

4. Проверьте имеются ли другие строки в таблице.

5. Если имеются, возвратитесь на шаг 1.

6. Если строк больше нет, вывести все значения сохраненные в шаге 3.

SQL сэкономит вам все это. Команды в SQL могут работать со всеми группами таблиц как с единым объектом и могут обрабатывать любое количество информации извлеченной или полученной из их, в виде единого модуля.

Команды SQL

Язык SQL состоит из ограниченного числа команд, специально предназначенных для управления данными.

Одни из этих команд служат для определения данных, другие – для их обработки, а остальные – для администрирования данных.

Зарезервированные слова

Кроме команд, специальное значение в SQL имеют и некоторые другие слова.

Вместе с командами они зарезервированы для специального использования, поэтому эти слова нельзя применять в качестве имен переменных или любым другим способом, для которого они не предназначены.

Легко увидеть, почему таблицам, столбцам и переменным нельзя давать имена из списка зарезервированных слов. Представьте себе, какая путаница возникнет из-за такого оператора:

SELECT SELECT FROM SELECT WHERE SELECT = WHERE;

Типы данных

В разных реализациях SQL поддерживаются различные исторически сложившиеся типы данных. В спецификации SQL:2003 признаны только пять заранее определенных общих типов:

числовой;

строковый;

логический;

даты-времени;

интервальный.

Внутри каждого из этих типов может быть несколько подтипов (точный числовой; приблизительный числовой; символьный строковый; битовый строковый; строковый для больших объектов).

Кроме встроенных, заранее определенных типов также поддерживаются сконструированные и определяемые пользователем типы.

Сейчас в базах данных хранятся данные многих разных типов, в том числе графические изображения звуки и анимация.

Неопределенные значения

Если в поле базы данных находятся какие-то данные, то в этом поле имеется ***определенное значение****. А если поле не содержит никаких данных, то говорят, что у него* ***неопределенное значение****.*

Неопределенное значение (null) в числовом поле – это не одно и то же, что нуль.

А в символьном поле неопределенное значение – это не одно и то же, что пустая строка.

И нуль и пустая строка являются определенными значениями. Неопределенное же значение указывает на то, что имеющееся в поле значение ***неизвестно****.*

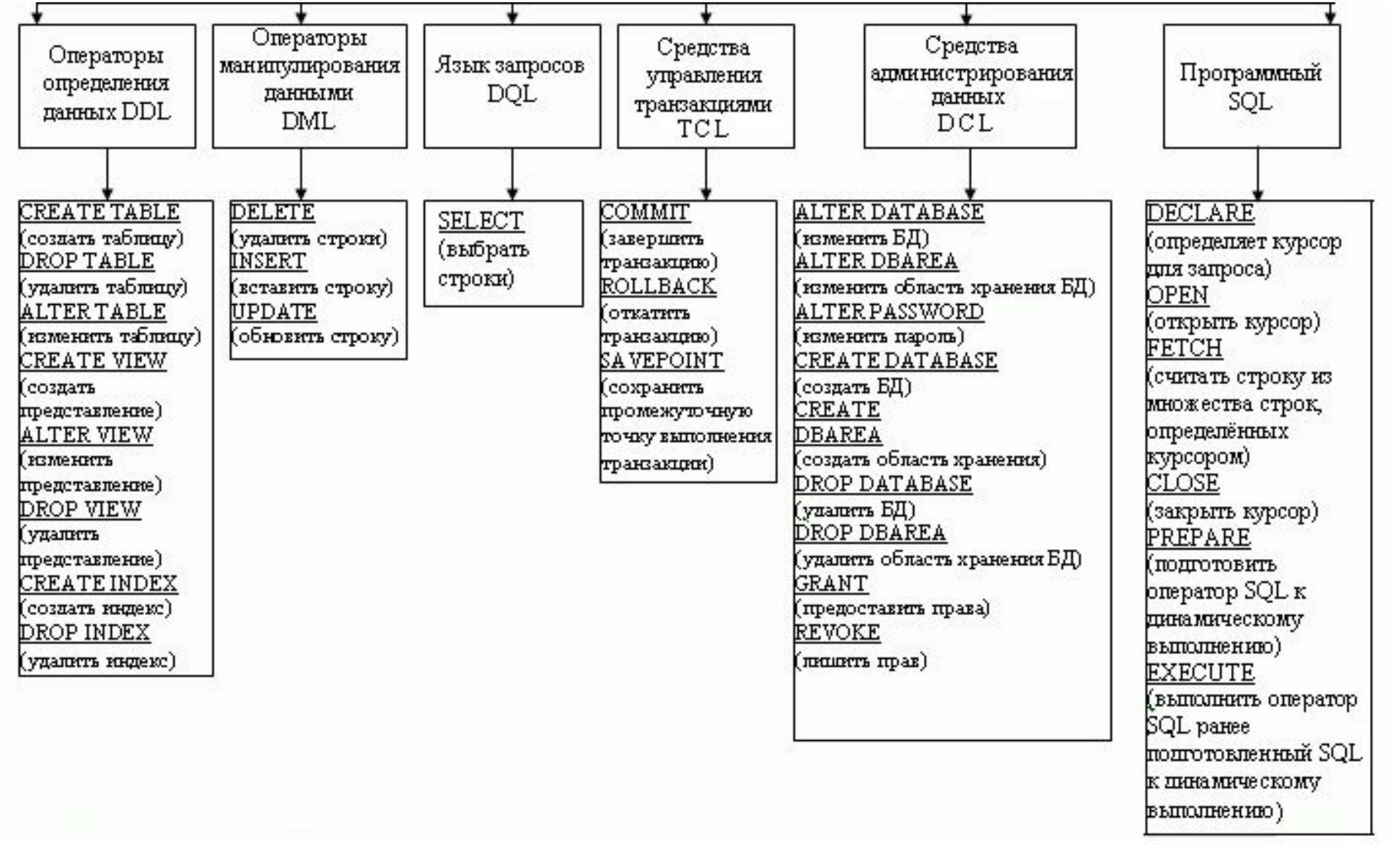
Совет:

Поле может иметь неопределенное значение по самым разным причинам. Так что не торопитесь с выводами относительно того, что означает конкретное неопределенное значение.

В следующем списке приведены некоторые из этих случаев и даны примеры каждого из них.

- **Значение существует, но вам оно пока что неизвестно**. До того, как была точно вычислена масса кварка, вы в самой верхней строке таблицы QUARM (кварк) установили в поле MASS (масса) неопределенное значение.
- **Значение пока что не существует**. В строке SQL For Dummies, 5th Edition таблицы BOOKS (книги) вы установили в поле TOTAL_SOLD (всего продано) неопределенное значение, так как первые данные о продажах за квартал еще не поступили.
- **Поле для данной строки неприменимо**. В строке C-3PO таблицы EMPLOYEE (наемный работник) вы установили в поле SEX (пол) неопределенное значение, так как C-3PO – это андроид, у которого пола нет.
- **Значение выходит за пределы установленного диапазона**. В строке Oprah Winfrey (Опра Уинфри) таблицы EMPLOYEE вы установили в поле SALARY (зарплата) неопределенное значение, так как для этого поля вы задали тип NUMERIC (8.2), а оклад, предусмотренный в контракте Опра, превышает 999999.99 доллара.

Подразделы SQL



Общий вид оператора Select:

Select [All или DISTINCT] (<список полей> или *)

From <список таблиц>

[Where <условие выборки или соединения>]

[Group by <список полей результата>]

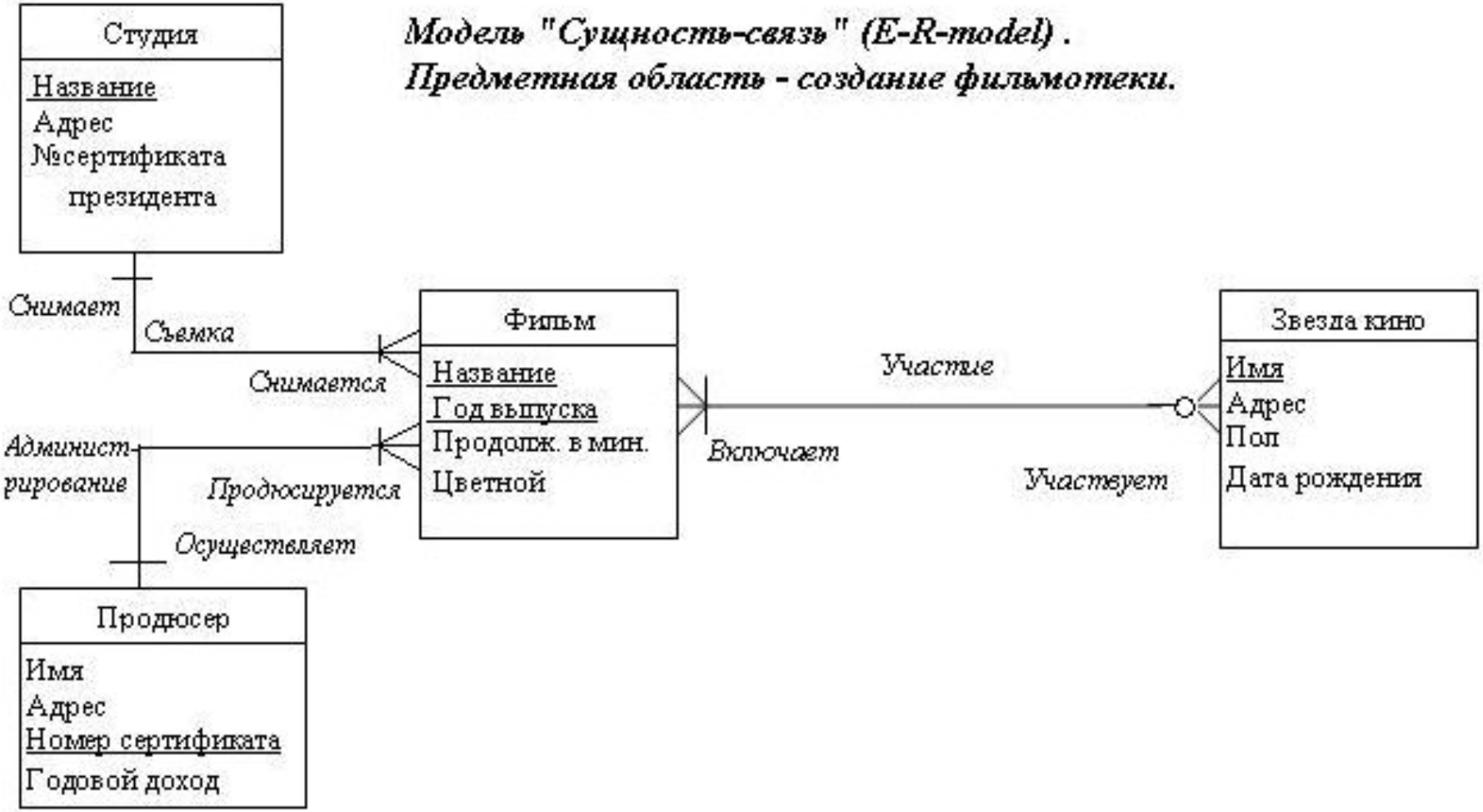
[Having <условие для группы>]

[Order by <список полей, по которым упорядочивается выбор>];

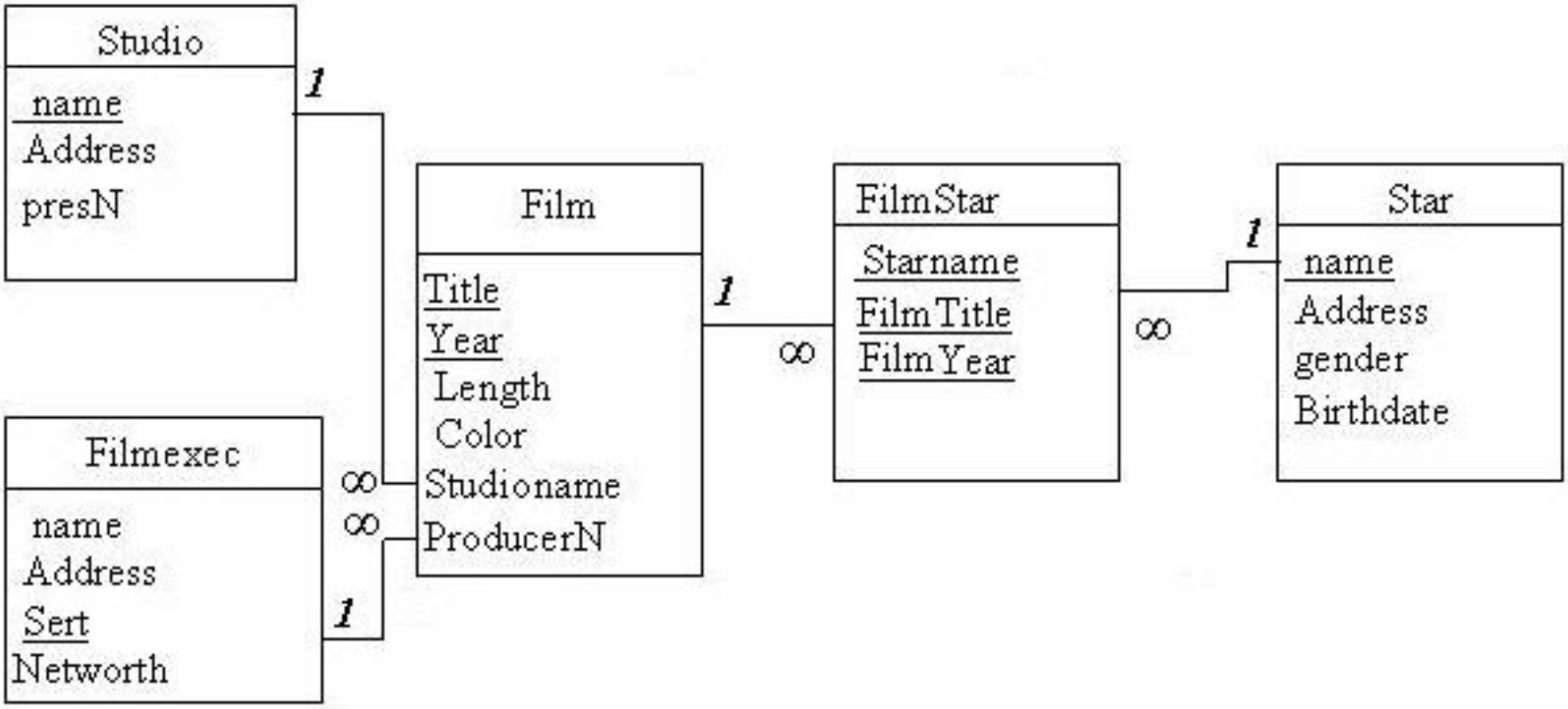
Последовательность обработки элементов оператора Select:

1.	*From*	Определяются имена используемой таблицы или нескольких таблиц
2.	*Where*	Выполняется фильтрация строк объекта в соответствии с заданными условиями
3.	*Group by*	Образуются группы строк, имеющих одно и то же значение в указанном столбце
4.	*Having*	Фильтруются группы строк объекта в соответствии с указанным условием
5.	*Select*	Устанавливается, какие столбцы должны присутствовать в выходных данных
6.	*Order by*	Определяется упорядоченность результатов выполнения оператора

Модель "Сущность-связь" (E-R-model) .
Предметная область - создание фильмотеки.



Реляционная модель БД "Фильмотека"



База данных «*Фильмотека*» состоит из отношений со следующими схемами:

Film (title, year, length, Color, studioName, producerN)
– фильмы;

FilmStar (filmTitle, filmYear, starName)
–отношение, связывающее фильмы и звезд кино;

Star (name, address, gender, birthdate)
– звезды кино;

FilmExec (name, address, sert, netWorth)
– продюсеры;

Studio (name, address, pres_N) – студии.

Примеры

Однотабличные запросы

Пример 1. Простой запрос на выборку по условию.
Найти фильмы, выпущенные студией Disney в 1990 году.

```
SELECT *
FROM Film
WHERE StudioName= 'Disney' AND year =1990;
```

Форма этого запроса

«Выбрать…из…где...»

характерна для большинства запросов SQL.

- Пункт **FROM** указывает на отношение, к которому принадлежит запрос (Film);
- Пункт **WHERE** – содержит условие, во многом похожее на условие выбора в реляционной алгебре, которому должны удовлетворять кортежи;
- Пункт **SELECT** показывает, какие атрибуты кортежей, удовлетворяющих условию, становятся частью ответа.

*Знак * в примере указывает на то, что порождается полный кортеж.*

Когда процессор запросов встречает в отношении Film кортеж

title	year	length	Color	studioName	producerN
Pretty woman	1990	119	true	Disney	12340

Он подставляет значения StudioName и year, условие в пункте WHERE становится истинным, а данный кортеж становится частью результата (12340 - вымышленный номер сертификата продюсера фильма).

Пример 2. Проекция в SQL.

Получить только название фильма и его продолжительность:

```
SELECT title, length
FROM Film;
```

Возможна перестановка порядка столбцов таблицы:

```
SELECT title, StudioName, year
FROM Film;
```

Вывод: структура информации в таблицах - это основа для активной перестройки структуры в SQL.

Пример 3. Проекция и фильтрация в SQL.

Получить только название фильма, соответствующего условию, и его продолжительность:

```
SELECT title, length
FROM Film
WHERE StudioName= 'Disney' AND year =1990;
```

Запрос будет производить вывод таблицы с двумя столбцами:

title	length
Pretty woman	119

Пример 4. Удаление избыточных данных.

Понятие отношение в SQL отличается от понятия отношение в реляционной алгебре. Об удалении дубликатов надо заботиться отдельно.

DISTINCT (ОТЛИЧИЕ) - аргумент который обеспечивает устранение повторяющихся значений из результата запроса. По умолчанию – ALL.

```
SELECT StudioName
FROM Film ;
```

или

```
SELECT ALL StudioName
FROM Film ;
```

StudioName
Disney
Paramount
Paramount
Disney
Disney
MGM

```
SELECT DISTINCT StudioName
FROM Film;
```

StudioName
Disney
Paramount
MGM

Пример 5.Переименование атрибутов.

Иногда требуется построить отношение, заголовки столбцов которого отличаются от атрибутов исходного отношения, упомянутого в пункте FROM. Тогда за именем атрибута ставится ключевое слово AS и псевдоним, представляющий имя, которое появится в результирующем отношении.

```
SELECT title AS name, length AS dlina
FROM Film
WHERE studioName ='Disney' AND year =1990;
```

В результате получается такое же множество кортежей, как и в примере 2, и столбцы в нем озаглавлены атрибутами name и dlina:

name	dlina
Pretty woman	119

В языке SQL служебное слово AS можно опустить (*SELECT title name, length dlina FROM Film* ….)

Пример 6. Использование формулы.

Получить такой же вывод, как и в примере 5, с продолжительностью фильма в часах:

```
SELECT title AS name, length/60 AS dlina_Hours
```

name	dlina_Hours
Pretty woman	1,983333

Важное замечание.

SQL нечувствителен к регистру, т.е. буква, набранная в верхнем и в нижнем регистре, считается одной и той же буквой.

`FROM` `FrOm` `from`

Имена атрибутов, отношений, псевдонимов и т.д. тоже нечувствительны к регистру.

SQL различает регистр только внутри кавычек, т.е. `'FROM'` отличается от `'from'` и, разумеется, от ключевого слова FROM.

КОНСТРУИРОВАНИЕ УСЛОВИЙ ВЫБОРА в SQL.

В выражениях условий раздела WHERE (иногда говорят WHERE-фраза) могут быть использованы следующие предикаты:

1) Предикаты сравнения { =, <>, >,<, >=,<= }.

- Сравниваемые значения могут содержать константы и атрибуты отношений, упомянутых после слова FROM.

- Перед сравнением числовых значений к ним можно применять арифметические операции +, * и т.д. Например, выражение

*(year – 1930) * (year– 1930) < 100*

истинно для годов от 1930 до 1939.

К строкам можно также применять операцию конкатенации ‖. Например,

screen‖ shot имеет значение screenshot.

2) Предикат *Between A and B* — принимает значения между A и B.

Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона.

Одновременно в стандарте задан и противоположный предикат *Not Between A and B*, который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы.

Пример 7. Сложное условие выбора

```
SELECT title
FROM Film
WHERE (length >= 100) AND
        (year between 1990 and 2000);
```

3) Предикаты сравнения с образцом

LIKE и *NOT LIKE*.

Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае. Предикат NOT LIKE имеет противоположный смысл.

По стандарту в шаблон могут быть включены специальные символы:

– символ подчеркивания ‘_’ – для обозначения любого одиночного символа (в MS Access – ‘?’);

– символ процента (%) – для обозначения любой произвольной последовательности символов (‘*’);

– остальные символы, заданные в шаблоне, обозначают самих себя.

Пример 8. Использование шаблона:

```
SELECT title
FROM Film
WHERE Studioname like 'D%';
WHERE Studioname like '%Pictures%';
WHERE Studioname like 'M__';
WHERE Studioname not like'%Star%';
```

4) Предикат вхождения в множество

IN (перечисление элементов множества)

Предикат **IN** истинен тогда, когда сравниваемое значение входит в множество заданных значений. При этом множество значений может быть задано простым перечислением или встроенным подзапросом.

Существует противоположный предикат

NOT IN,

который истинен тогда, когда сравниваемое значение не входит в заданное множество.

Пример 9. Использование предиката вхождения в множество IN:

```
SELECT title
FROM Film
WHERE year IN (1990, 2000, 2010);
```

вместо:

```
WHERE (year=1990) OR (year=2000) OR (year=2010);
```

Пример 9а. Использование предиката NOT IN:

```
SELECT title
FROM Film
WHERE Studioname NOT IN ('Мосфильм','MGM');
```

5)Предикат сравнения с неопределенным значением

IS NULL (IS NOT NULL).

Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени.

Введение Null-значений вызвало необходимость модификации классической двузначной логики и превращения ее в трехзначную.

Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	B	Not A	A AND B	A OR B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	Null	FALSE	Null	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	Null	TRUE	FALSE	Null
Null	TRUE	Null	Null	TRUE
Null	FALSE	Null	FALSE	Null
Null	Null	Null	Null	Null

Пример 10. Найдем всех звезд кино, чьи адреса неизвестны:

```
SELECT name
FROM Star
WHERE address IS NULL;
```

Пример 11. Использование NOT NULL .

Чаще применяется конструкция:

```
WHERE address IS NOT NULL;
```

Допустима также:

```
WHERE NOT address IS NULL;
```

6) Предикаты существования *EXISTS* и несуществования *NOT EXISTS*.

Эти предикаты относятся к встроенным подзапросам, и подробнее мы рассмотрим их при изучении темы

Формирование подзапросов.

Пример 12. Образцы WHERE-фразы:

- *WHERE title = ‘LONDON’;*
- *WHERE netWorth >= 100 000;*
- *WHERE year >1970 and not color;*
- *WHERE name BETWEEN 'A' AND 'B';*
- *WHERE name LIKE 'A% A% A%';*
- *WHERE title LIKE 'Star _ _ _ _'*

Общий вид оператора Select:

Select [All или DISTINCT] (<список полей> или *)

From <список таблиц>

[Where <условие выборки или соединения>]

[Group by <список полей результата>]

[Having <условие для группы>]

[Order by <список полей, по которым упорядочивается выбор>];

Обобщение данных с помощью агрегатных функций.

В SQL есть операторы, которые применяются к столбцу отношения и порождают на нем некоторый итог, или агрегацию.

SUM - сумма значений в столбце.

AVG - среднее всех значений в столбце.

MAX - наибольшее из всех выбранных значений данного поля.

MIN - наименьшее из всех выбранных значений данного поля.

COUNT – число NOT NULL значений, включая дубликаты, если они явным образом не удаляются с помощью Distinct.

Пример 13. Найти среднюю величину чистого дохода всех продюсеров фильмов:

```
SELECT AVG(netWorth)
FROM FilmExec;
```

Запрос проверяет столбец отношения FilmExec (name, address, sert, netWorth), суммирует найденные в нем значения, извлекая по одному значению из каждого кортежа (включая дубликаты), и делит полученную сумму на число кортежей. Если дубликатов нет, запрос дает ожидаемую среднюю величину чистого дохода. При наличии дубликатов чистый доход продюсера фильмов, чей кортеж входит в отношение n раз, используется при подсчете средней величины тоже n раз.

Пример 14. Подсчитать число кортежей в отношении FilmExec:

```
SELECT COUNT(*)
FROM FilmExec;
```

В таблице FilmExec определен первичный ключ name, значит, в этом отношении нет дубликатов кортежей, и результат совпадет с числом продюсеров, внесенных в БД.

Применение оператора агрегации со знаком *, т.е. к кортежу в целом, имеет смысл только для COUNT. Нельзя применять любой другой оператор агрегации более чем к одному столбцу.

COUNT( ) включает и NULL, и дубликаты

Для получения гарантии, что дублирующиеся кортежи учитываются лишь один раз, можно вычислять COUNT только по одному атрибуту и использовать в запросе ключевое слово DISTINCT:

Пример 15. Подсчитать количество студий в отношении Film:

```
SELECT COUNT(DISTINCT Studioname)
FROM Film;
```

Группирование

Иногда необходимо вычислять агрегатные функции не для всей таблицы, а для группы кортежей.

Пример 16. Вычислить общее количество времени (в минутах), в течение которого длятся фильмы, снятые на каждой студии.

Тогда кортежи отношения Film нужно сгруппировать в соответствии со студиями и вычислить сумму в столбце length каждой полученной группы. Результат может потребоваться в виде таблицы, связывающей студии с суммами продолжительности их фильмов, например:

Studio	SUM(length)
Disney	12345
MGM	54321

Общий вид оператора Select:

Select [All или DISTINCT] (<список полей> или *)

From <список таблиц>

[Where <условие выборки или соединения>]

[Group by <список полей результата>]

[Having <условие для группы>]

[Order by <список полей, по которым упорядочивается выбор>];

Для получения такого результата применяется предложение GROUP BY.

За ключевыми словами GROUP BY указывается список *группируемых* атрибутов. В простейшей ситуации в предложении FROM указано только одно отношение, и его кортежи группируются согласно их значениям в группируемых атрибутах. Любой оператор агрегации, используемый в пункте SELECT, применяется только внутри групп.

16.
```
SELECT Studioname, SUM(length)
FROM Film
GROUP BY Studioname;
```

Запрос с группировкой **без операции агрегации**:

```
SELECT StudioName
FROM Film
GROUP BY StudioName;
```

дает такой же результат, как и запрос

```
SELECT DISTINCT StudioName
FROM Film;
```

Запрос с группировкой **без указания поля группировки в предложении SELECT**:

```
SELECT SUM(length)
FROM Film
GROUP BY StudioName;
```

дает такой же результат, как и запрос

```
SELECT SUM(length)
FROM Film;
```

Пример 16а.
Группировка по нескольким полям.

```
SELECT Studioname, year, SUM(length)
FROM Film
GROUP BY Studioname, year;
```

Отбор групп по условию. Предложение HAVING.

Пример 17. Вывести суммы продолжительности всех фильмов только тех студий, которые выпустили хотя бы один фильм до 1950 года:

```
SELECT Studioname, SUM(length)
FROM Film
GROUP BY Studioname
 HAVING MIN(year)<1950;
 HAVING SUM(length)<2000;
```

Упорядочение вывода

Порождаемые запросом кортежи можно упорядочить на основе значения любого атрибута. Для получения упорядоченного вывода в запрос добавляется пункт

ORDER BY <список атрибутов> [ASC или DESC];

По умолчанию принимается порядок по возрастанию – ключевое слово **ASC** (ascending), но его можно изменить на убывание с помощью ключевого слова **DESC** (descending).

Пример 18. Вывести кортежи в алфавитном порядке названия фильмов:

```
SELECT *
FROM Film
WHERE year>=1990
ORDER BY title;
```

Пример 19.

Вывести год и количество фильмов, снятых в этом году, в порядке убывания количества:

```
SELECT year, Count(title) AS Num
FROM Film
Group by year
ORDER BY Num DESC;
```

ORDER BY **при использовании с предложением** ***GROUP BY*** **служит для упорядочения групп.**

Последовательность обработки элементов оператора Select:

1.	*From*	Определяются имена используемой таблицы или нескольких таблиц
2.	*Where*	Выполняется фильтрация строк объекта в соответствии с заданными условиями
3.	*Group by*	Образуются группы строк, имеющих одно и то же значение в указанном столбце
4.	*Having*	Фильтруются группы строк объекта в соответствии с указанным условием
5.	*Select*	Устанавливается, какие столбцы должны присутствовать в выходных данных
6.	*Order by*	Определяется упорядоченность результатов выполнения оператора

Пример 20.

Использование всех предложений оператора ***SELECT (сформулировать запрос самостоятельно):***

```
SELECT year, Count(title) AS Num
FROM Film
WHERE Not color
Group by year
Having Num>15
ORDER BY Num;
```