Системы управления базами данных

Куликова Елена Васильевна

- Проектирование БД
- Реализация в СУБД
- СУБД Access
- MS SQL Server
- SQL (язык запросов)

- Реализация (СУБД Access)
- Проектирование!!! Автоматизированное проектирование
- SQL
- MS SQL Server
- MySQL
- ГОСы!!!

Тема 1. **Основные положения теории баз данных**

1.1. Основные понятия и определения

Появление новой информационной технологии – концепции баз данных

развитие АИС разного назначения и масштаба, в первую очередь в области бизнес-приложений

База данных

- *База данных (БД)* – именованная совокупность данных, отображающая состояние объектов и их отношений в рассматриваемой предметной области.

База данных

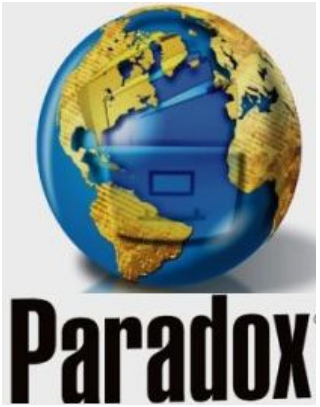
- **База данных** (БД) – структурированная совокупность данных, относящихся к определенной предметной области.

Система управления базами данных (СУБД)

- *Система управления базами данных (СУБД)* – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.
- MS Access, реляционная СУБД
- MS SQL Server
- Oracle
- MySQL (PHP+ MySQL)
- Paradox (не актуально)
- FoxPRo (не актуально)

Система управления базами данных (СУБД)

- **Система управления базами данных** (СУБД) – программа, предназначенная для работы с базами данных: определения структуры базы данных, ее наполнения, редактирования, обработки.



Система баз данных (СБД)

- *Система баз данных (СБД)* – это система специально организованных данных (баз данных), программных, технических, языковых, организационно-методических средств для централизованного накопления и коллективного многоцелевого использования данных.

Логическая концептуальная модель

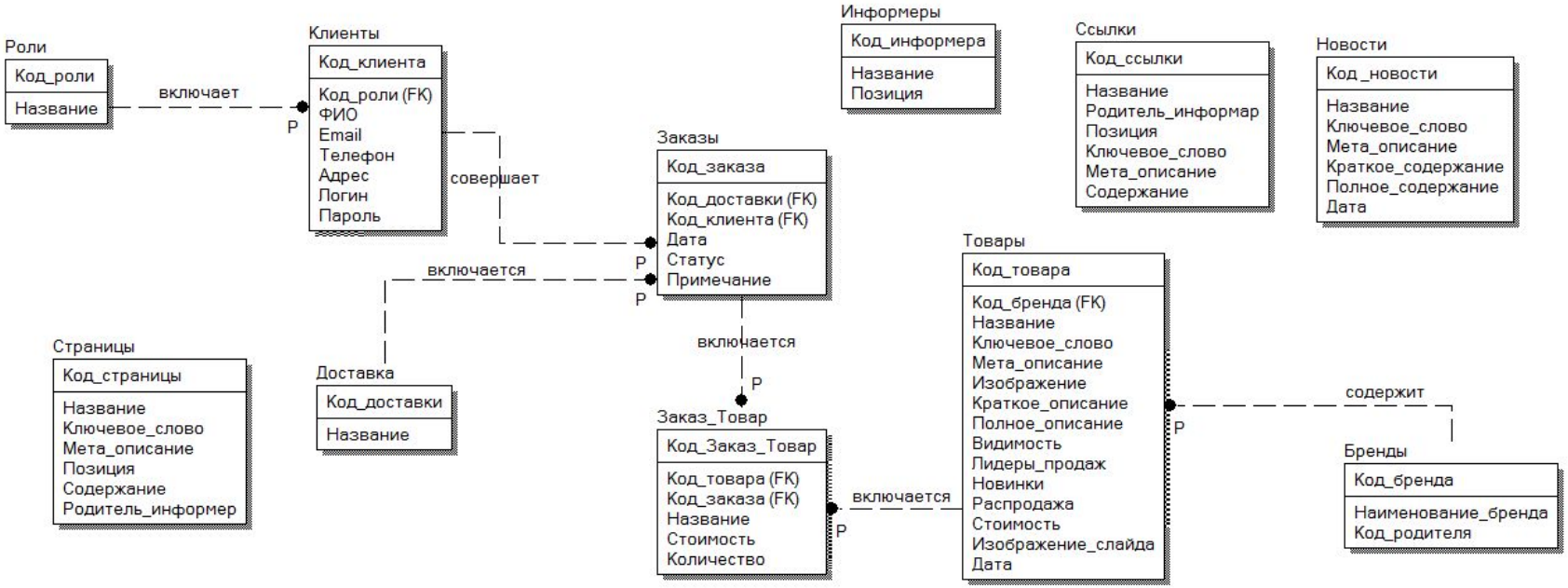


Схема данных в СУБД Access

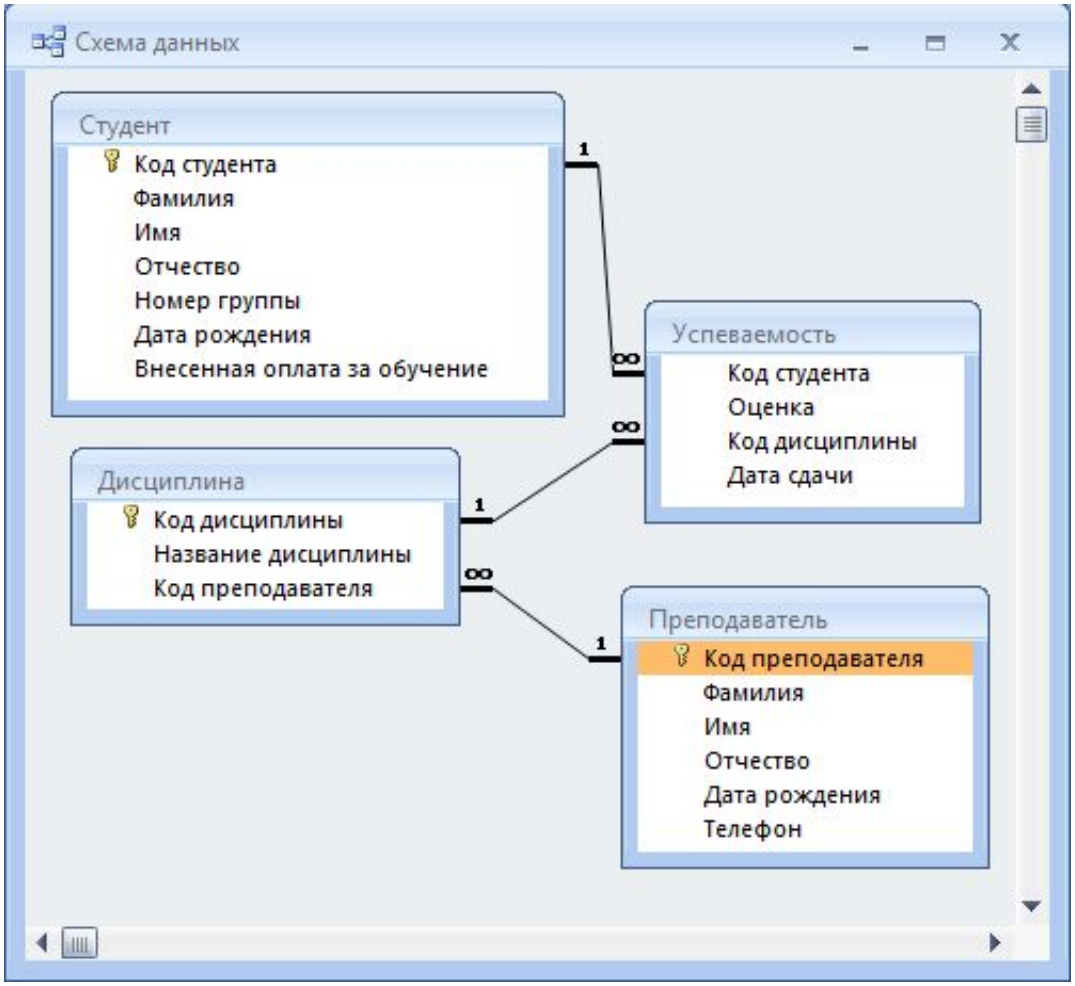
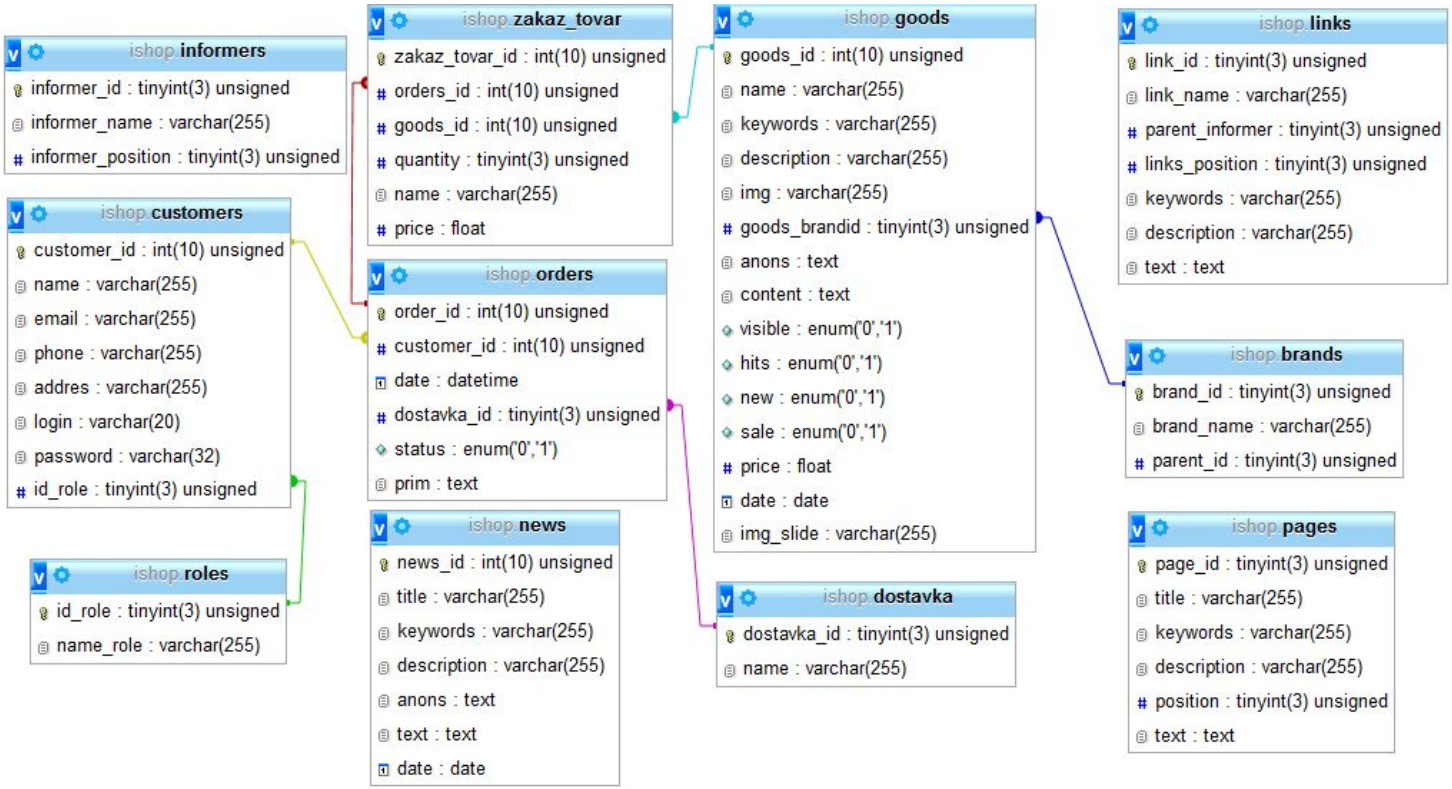


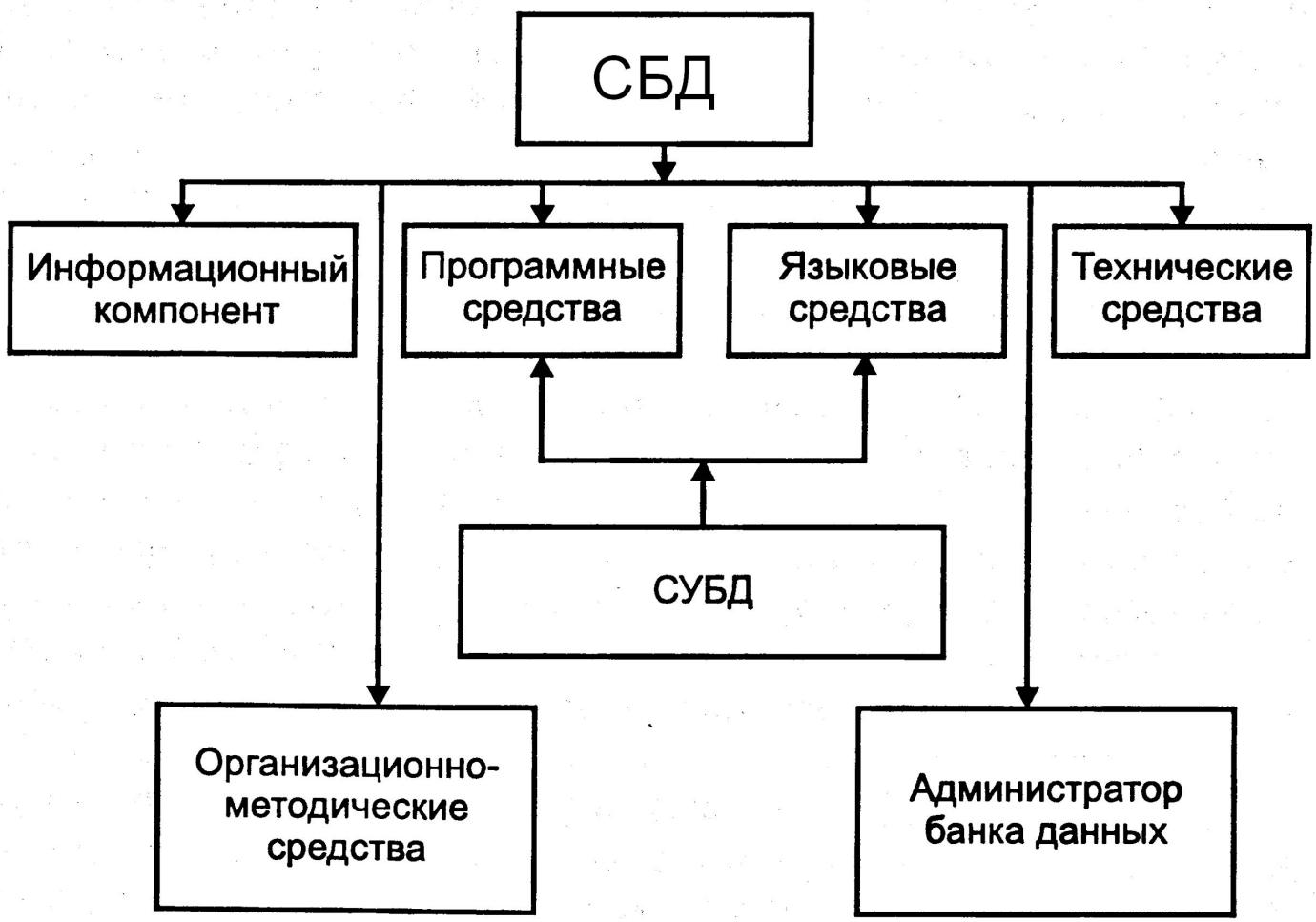
Схема данных в СУБД MySQL



1.2. Основные требования к СБД

- адекватность отображения предметной области (полнота, целостность, непротиворечивость и актуальность данных);
- возможность взаимодействия пользователей разных категорий, обеспечение высокой эффективности доступа;
- дружественность интерфейса;
- обеспечение секретности и конфиденциальности;
- обеспечение взаимной независимости программ и данных;
- обеспечение надежности – защита данных от случайного и преднамеренного разрушения, возможность восстановления данных в случае сбоев в системе;
- распределенная обработка данных и обеспечение эффективного доступа пользователей к данным в любой точке сети.

1.3. Компоненты системы баз данных



Информационный компонент: информационная база

- - это данные, отражающие состояние предметной области и используемые ИС

Информационная база включает:

- собственно данные;
- метаданные (описания этих данных).

Языковые средства

Язык описания данных (ЯОД)

- Входит в состав СУБД
- Нужен для выражения обобщенного взгляда на данные
- Позволяет определять схемы БД, характеристики хранимых данных, параметры хранения их в памяти
- Может включать средства поддержки целостности, ограничения доступа, секретности.

Одна БД на ЯОД разных СУБД может описываться по-разному.

Язык манипулирования данными (ЯМД)

- *Язык манипулирования данными (ЯМД)* включает средства запросов к БД и поддержания БД (добавление, удаление, обновление данных, создание и уничтожение БД, обеспечение запросов к справочнику БД).

ЯМД разделяются на:

- процедурные;
- непроцедурные (декларативные).

Отличия

При пользовании процедурными языками надо указать, какие действия и над какими объектами необходимо выполнить, чтобы получить результат.

В непроцедурных языках указывается, что надо получить в ответе, а не как этого достичь.

Процедурные языки

могут различаться по *основным информационным единицам*, которыми они манипулируют. Это могут быть:

- языки, ориентированные на обработку данных по отдельной записи;
- языки, ориентированные на операции над множеством записей.

Примеры непроцедурных языков

языки, основанные на реляционном исчислении:

- язык запросов SQL
- табличный язык QBE

Процедурный язык программирования

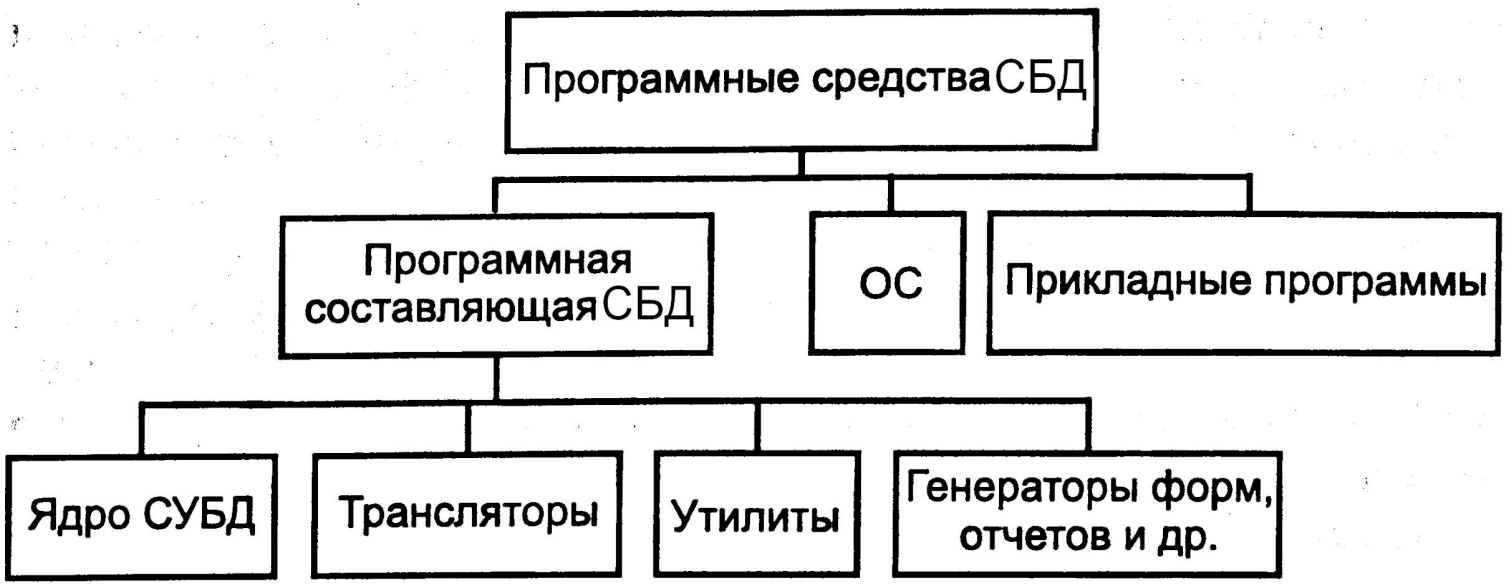
- язык xBase (для dBase и FoxPro)

Языковые средства

SQL (Structured Query Language)

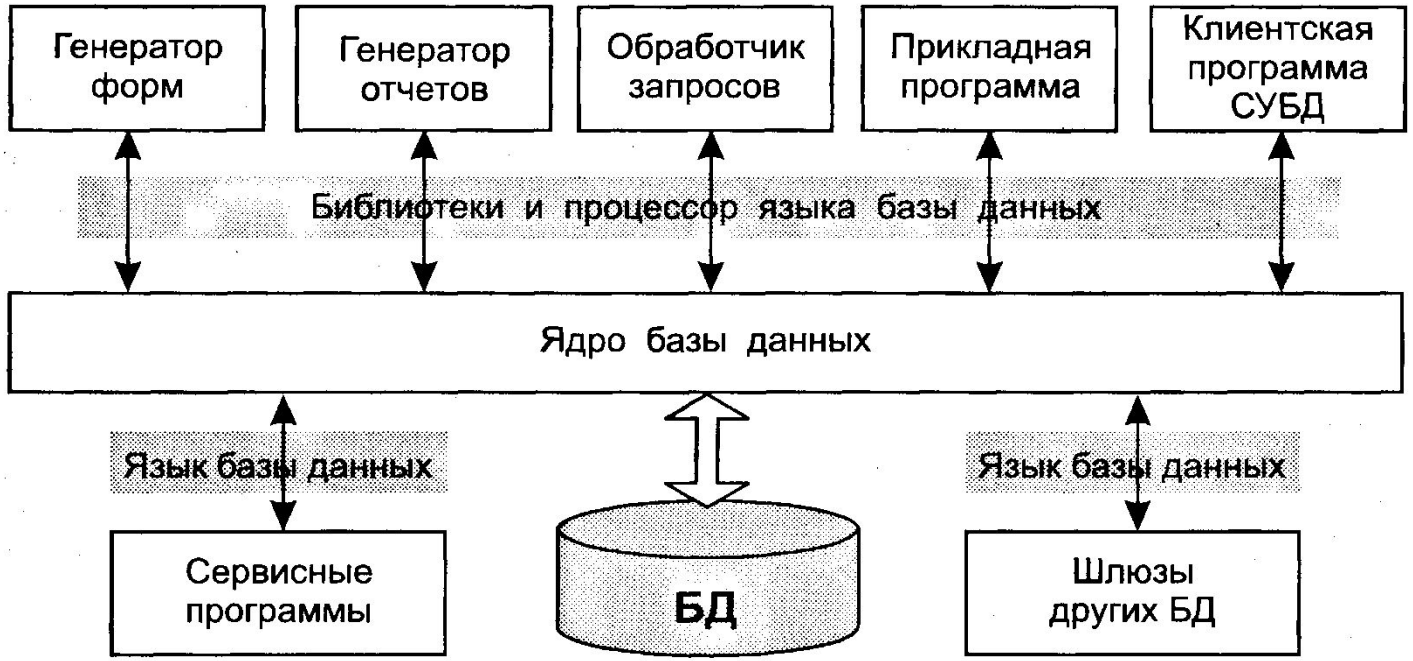
- Наиболее распространенным языком является SQL, предоставляющий средства обработки запросов и функции по созданию, обновлению и управлению доступом.
- SQL соединяет в себе ЯОД и ЯМД.
- SQL не является полноценным языком программирования.
- Для доступа к БД из прикладных программ SQL-выражения встраиваются в конструкции базового языка.

Программные средства СБД



Программная составляющая СБД

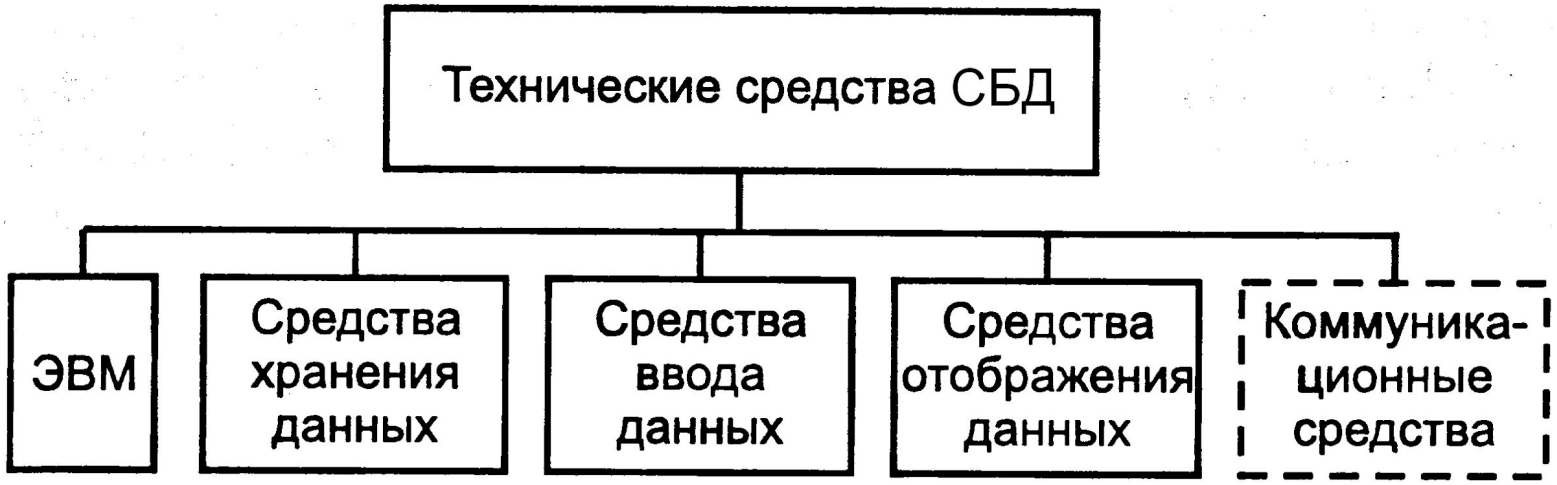
- осуществляет обработку данных и взаимодействие с операционной системой (ОС) и прикладными программами



Компоненты в составе комплекса

- *ядро,* обеспечивающее управление данными во внешней и оперативной памяти, а также протоколирование изменений;
- *процессор языка БД,* обеспечивающий обработку и оптимизацию запросов на выборку и изменение данных;
- *подсистему (библиотеку) поддержки программных вызовов,* которая обслуживает прикладные программы управления данными, взаимодействующие с СУБД через средства пользовательского интерфейса;
- *сервисные программы* (системные и внешние утилиты), обеспечивающие настройку СУБД, восстановление после сбоев и другое обслуживание.

Технические средства СБД



1.4. Информационная модель данных

- Информационная модель - это представление объектов и отношений, ограничений, правил и операций, призванное указать семантику данных для выбранного домена (проблемной области).
- *Как правило, она определяет отношения между классами объектов, но может также включать отношения между конкретными объектами.*

- Одним из ключевых моментов создания ИС с целью автоматизации информационных процессов организации является всестороннее изучение объектов автоматизации, их свойств, взаимоотношений между этими объектами и *представление* полученной информации в виде информационной модели данных.

Тема 2. *Жизненный цикл БД*

- *Жизненный цикл БД* – это развитие системы базы данных во времени

Разные степени детализации ЖЦ. Вариант 1-ый

1 проектирование

- **техническое проектирование**
- **рабочее проектирование**

2 эксплуатация

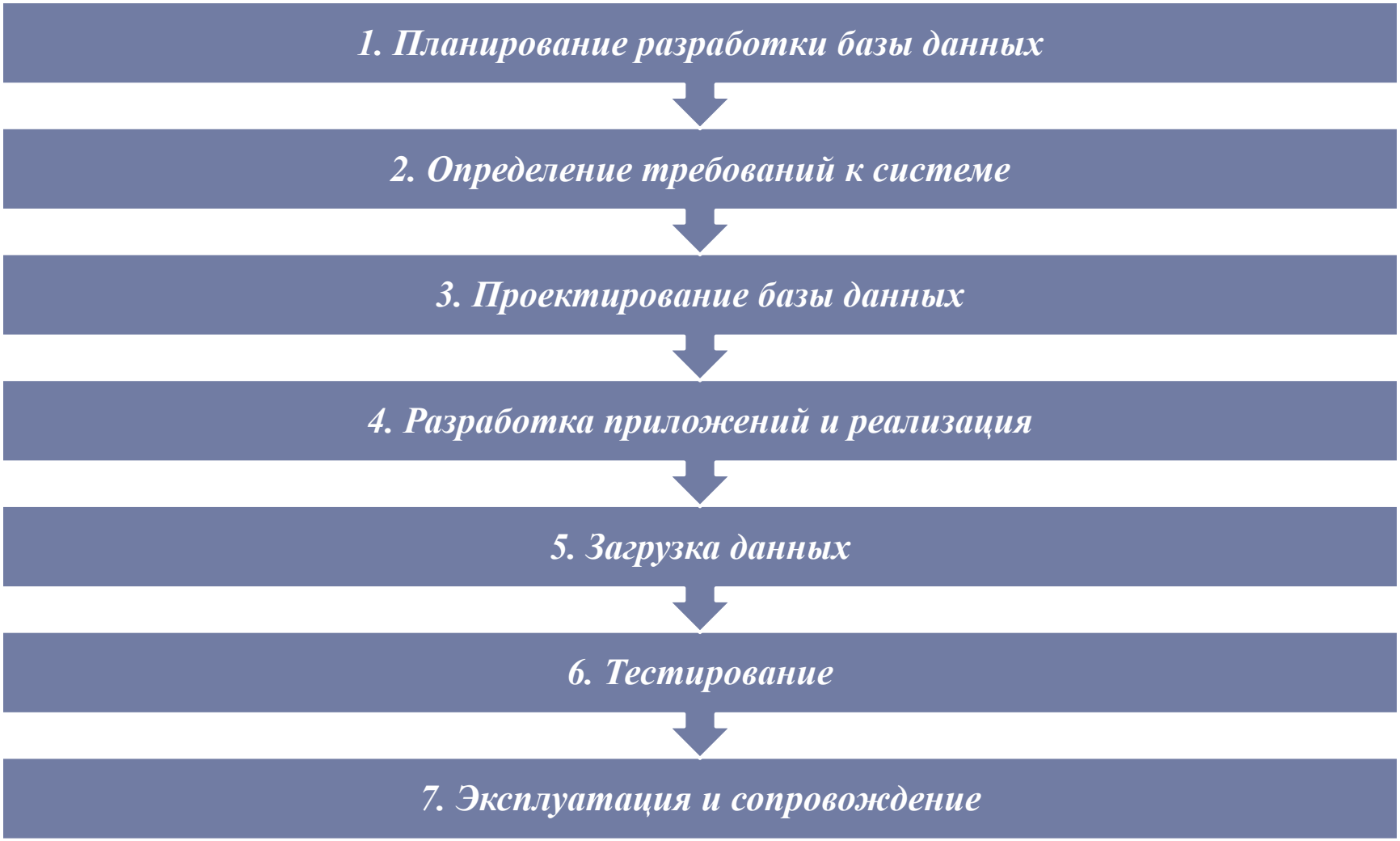
- ***организация сбора информации***
- ***заполнение БД собранной информацией***
- ***сдача БД в эксплуатацию***
  - *опытная эксплуатация*
  - *промышленная эксплуатация*
- ***дальнейшее развитие БД***
  - *реорганизация*
  - *реструктуризация*

3 вывод из эксплуатации

Разные степени детализации ЖЦ. Вариант 2-ой

1. *Проектирование БД*
2. *Проектирование приложений*
3. *Реализация БД*
4. *Разработка специальных средств администрирования*
5. *Эксплуатация БД*

Разные степени детализации ЖЦ. Вариант 3-ий



1. Планирование разработки базы данных

2. Определение требований к системе

определение требований к информационному наполнению;

определение требований к оборудованию;

определение требований к программному обеспечению;

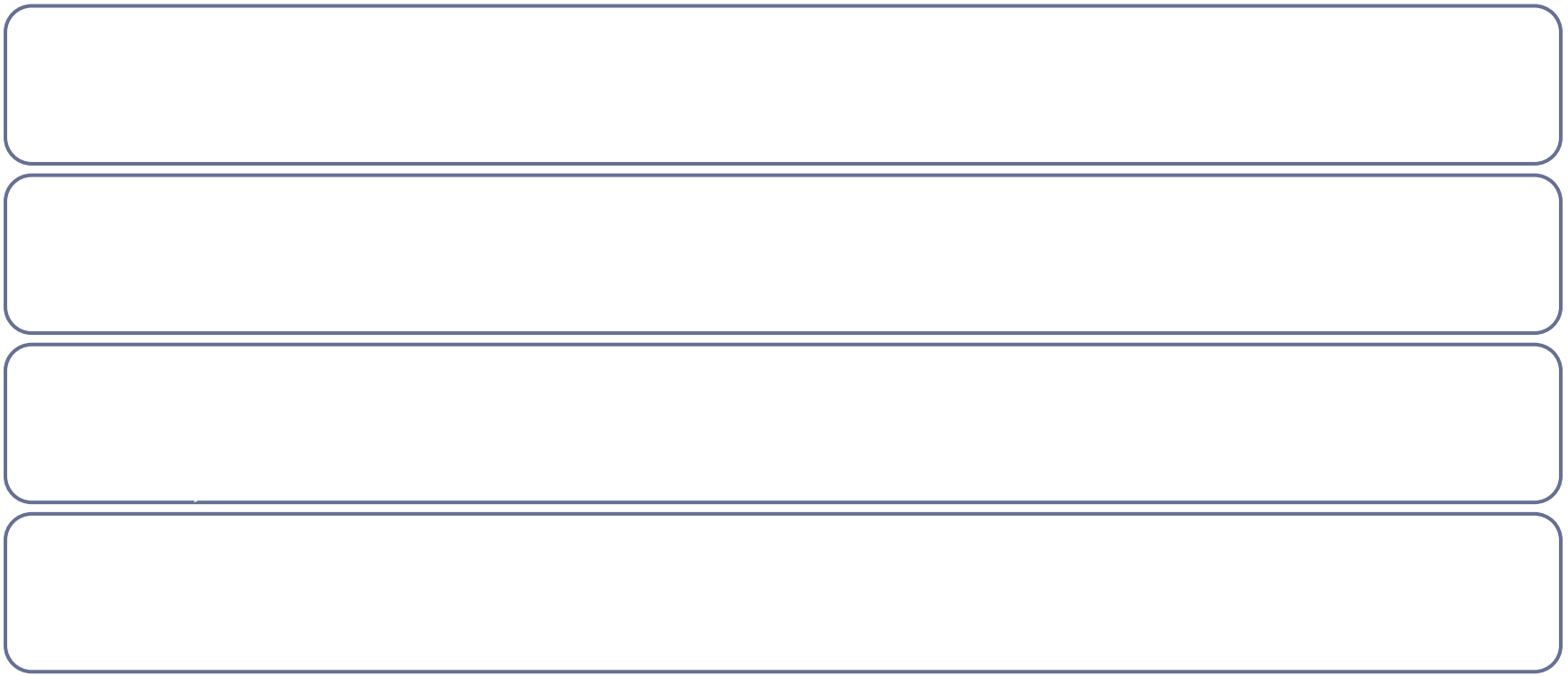
определение требований к организационно-методическому обеспечению;

определение требований к функциональности;

определение требований к интерфейсу и др.

3. Проектирование базы данных.

При проектировании базы данных должны быть решены следующие **задачи**:



Логическое (даталогическое) проектирование

- Логическое (даталогическое) проектирование представляет собой необходимый этап при создании БД.
- Основной задачей логического проектирования является разработка логической модели.

Логическое (даталогическое) проектирование, этапы

1. Определение сущностей;
2. Определение описательных атрибутов сущностей;
3. Определение первичных и альтернативных ключей;
4. Определение связей между сущностями и их характеристик;
5. приведение модели к требуемому уровню нормальной формы;
6. графическое представление логической модели (модели «сущность-связь») *является результатом этапа.*

Физическое проектирование

- Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным.
- Физическая модель данных, в отличие от логической, зависит от конкретной СУБД, фактически являясь отображением системного каталога.
- В физической модели содержится информация обо всех объектах базы данных: определяются типы данных, характеристики полей, индексы для таблиц.

Физическое проектирование

- Выполнить переименования атрибутов и сущностей: на английском языке, исключая пробелы.
  - Такое именование является необходимым для исключения разного рода ошибок при создании запросов, макросов, процедур и написании программ.
- Определить таблицы различного назначения: справочные, таблицы-связки, основные.
- Определить дополнительные свойства полей.
- Преобразовать (при необходимости) связи между таблицами.

Таблица соответствий

Логическое проектирование	Физическое проектирование
Сущность	Таблица
Экземпляр сущности	Запись
Атрибут сущности	Поля таблицы
Идентификатор, первичный ключ, главный ключ	Идентификатор, первичный ключ, главный ключ, поле главного ключа
Вторичный ключ, мигрирующий атрибут, внешний ключ	Вторичный ключ, поле внешнего ключа, внешний ключ
Предварительный тип данных (числовой, текстовый,….)	Точный тип данных (текстовый, memo, целое, длинное целое, …….)

4. Разработка приложений и реализация

проектирование транзакций

проектирование пользовательского интерфейса, объектов для работы с БД

создание средств обеспечения целостности

защита данных

Транзакция

- *Транзакция* - это последовательность операций над базой данных, рассматриваемых СУБД как единое целое.
- *Транзакция* – выполнение в качестве атомарного (неделимого) действия одной и более операций над БД, не приводящее к нарушению целостности БД.

Пример. Выполнение группы запросов

- *Если транзакция успешно выполняется, то СУБД фиксирует изменение базы данных, произведенные ею, во внешней памяти базы данных.*

5. Загрузка данных

- наполнение базы данных

6. Тестирование

- обнаружение имеющихся ошибок, показатели надежности и качества созданного программного обеспечения

7. Эксплуатация и сопровождение

- ввод в эксплуатацию
- наблюдение за созданной системой
- поддержка ее нормального функционирования
- создание дополнительных программных компонент
- модернизация самой базы данных

Тема 3. Этап проектирования базы данных

3.1 Уровни моделирования предметной области

Уровни моделирования:

- информационно-логический (инфологический, или концептуальный);
- даталогический;
- физический.

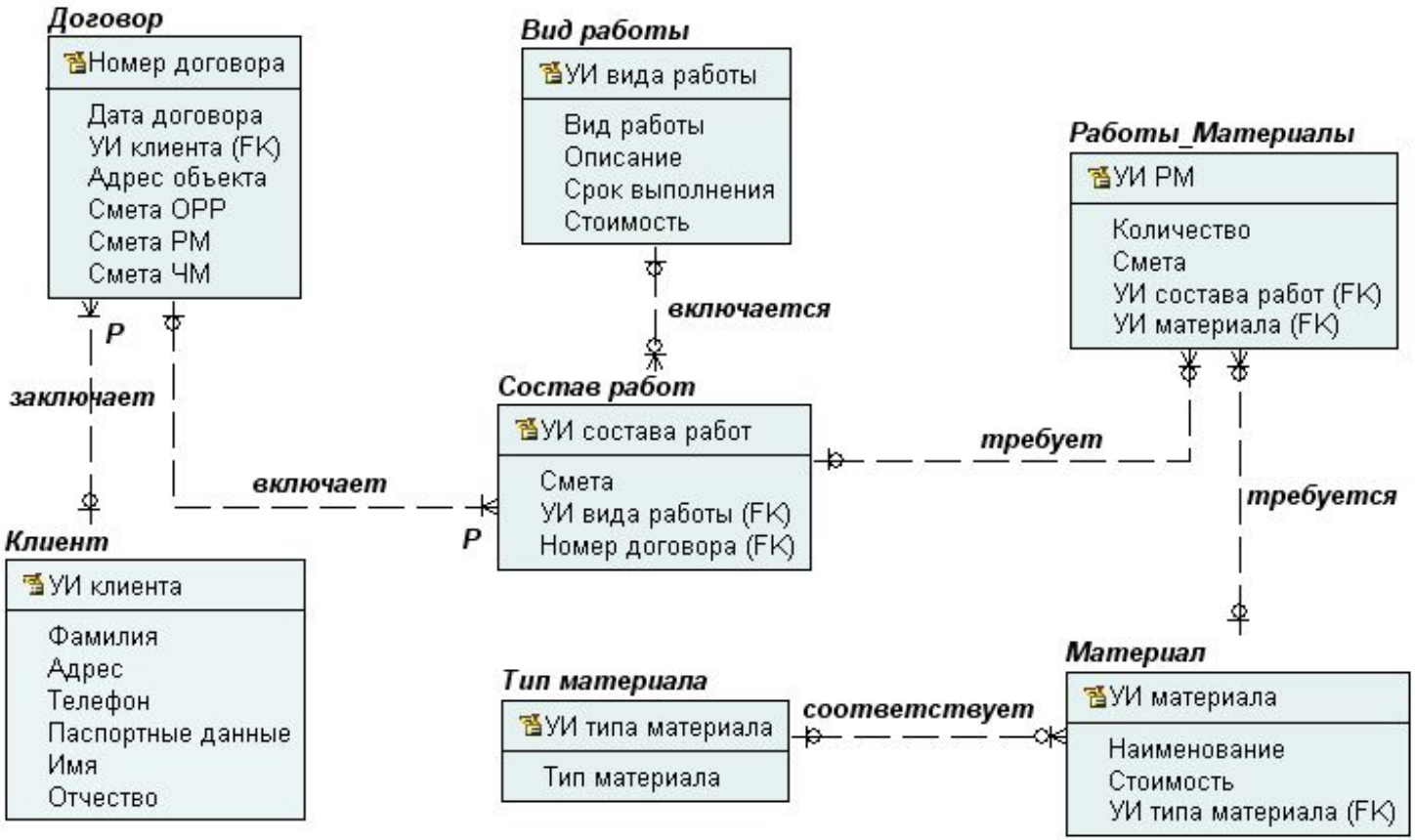
Модели базы данных:

- информационно-логическая (инфологическая, или концептуальная);
- даталогическая;
- физическая.

3.2. Информационно-логическая модель базы данных. Методология информационного моделирования IDEF1X

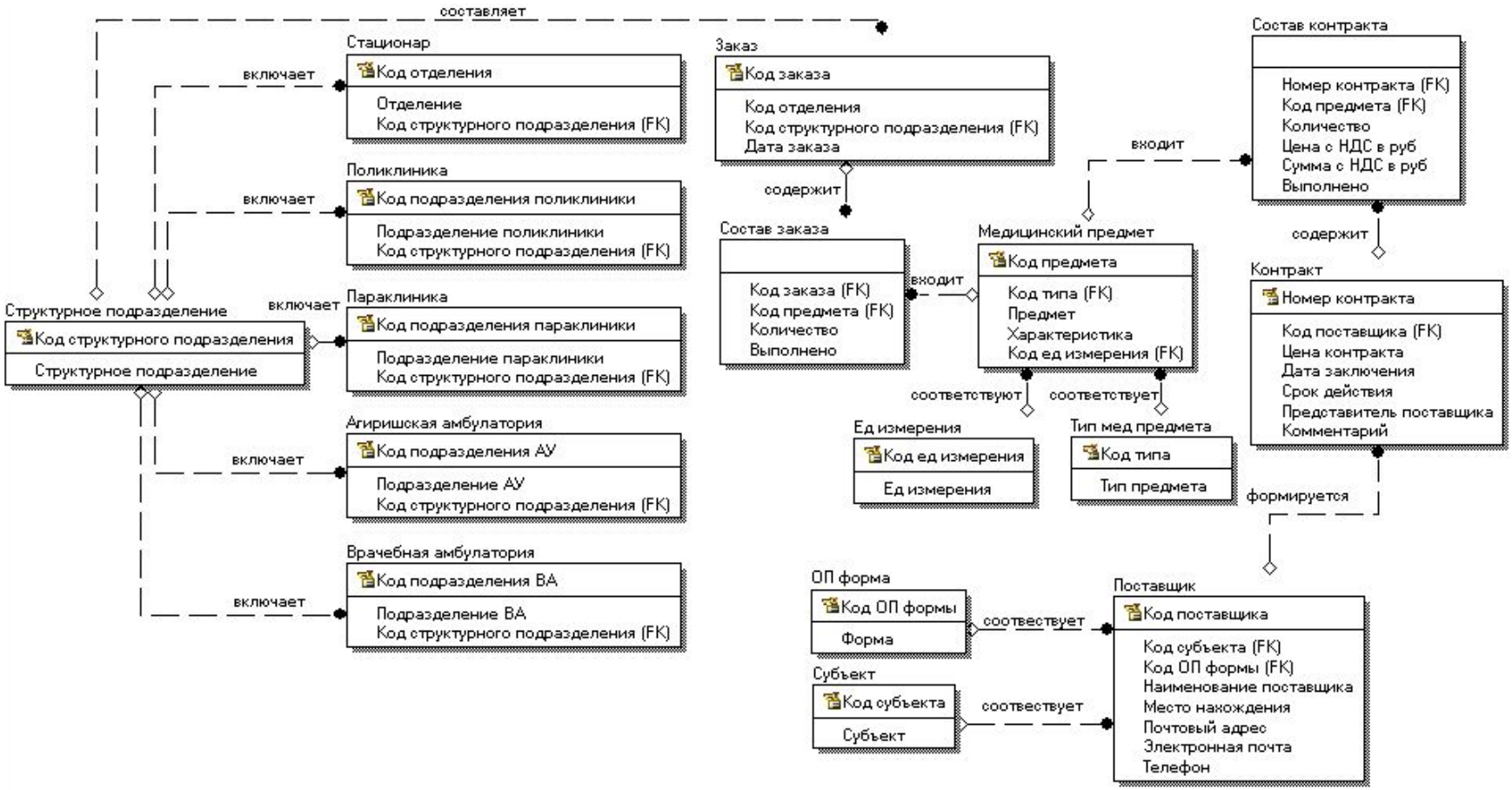
Информационно-логическая модель базы данных (созданная в case-средстве Erwin)

предметная область: Учет заказов и работ в строительной фирме по ремонту квартир



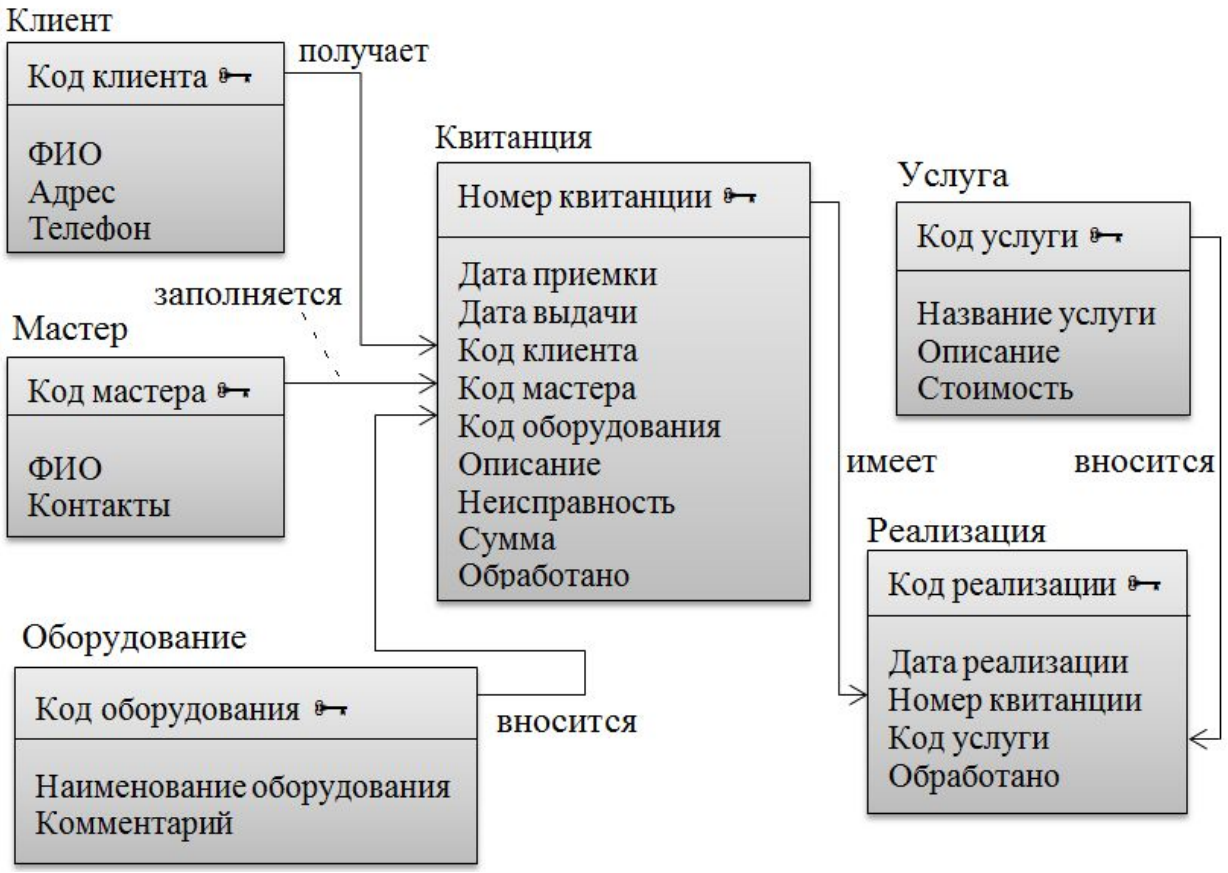
Информационно-логическая модель базы данных
(созданная в case-средстве Erwin)

предметная область: автоматизация функций сотрудников, отдела по организации и проведению закупок учреждения здравоохранения



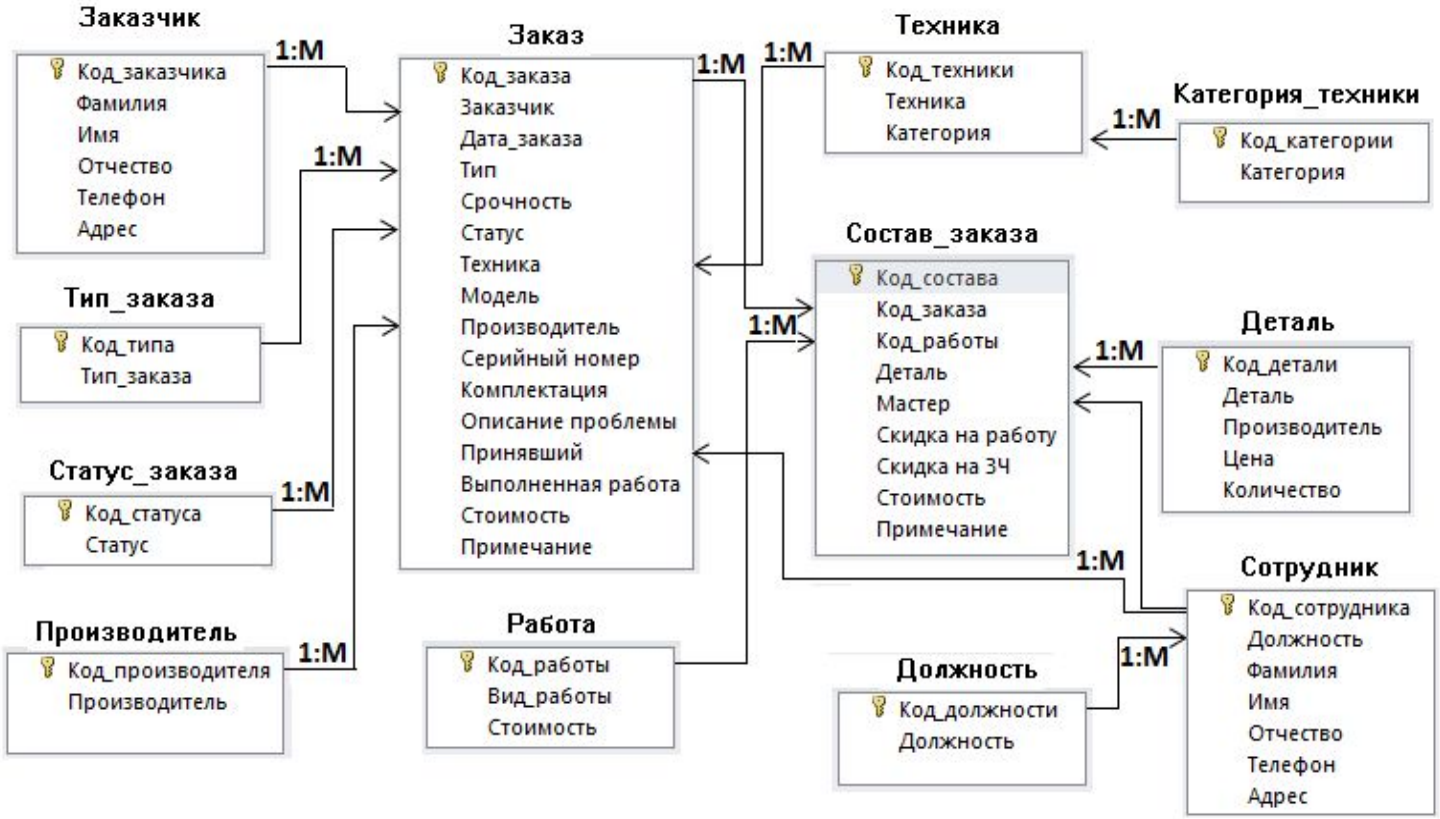
Информационно-логическая модель базы данных (созданная без применения case-средств)

предметная область: сервисный центр



Информационно-логическая модель базы данных (созданная без применения case-средств)

предметная область: организация по ремонту бытовой техники



Что такое IDEF1X?

Методология IDEF1X

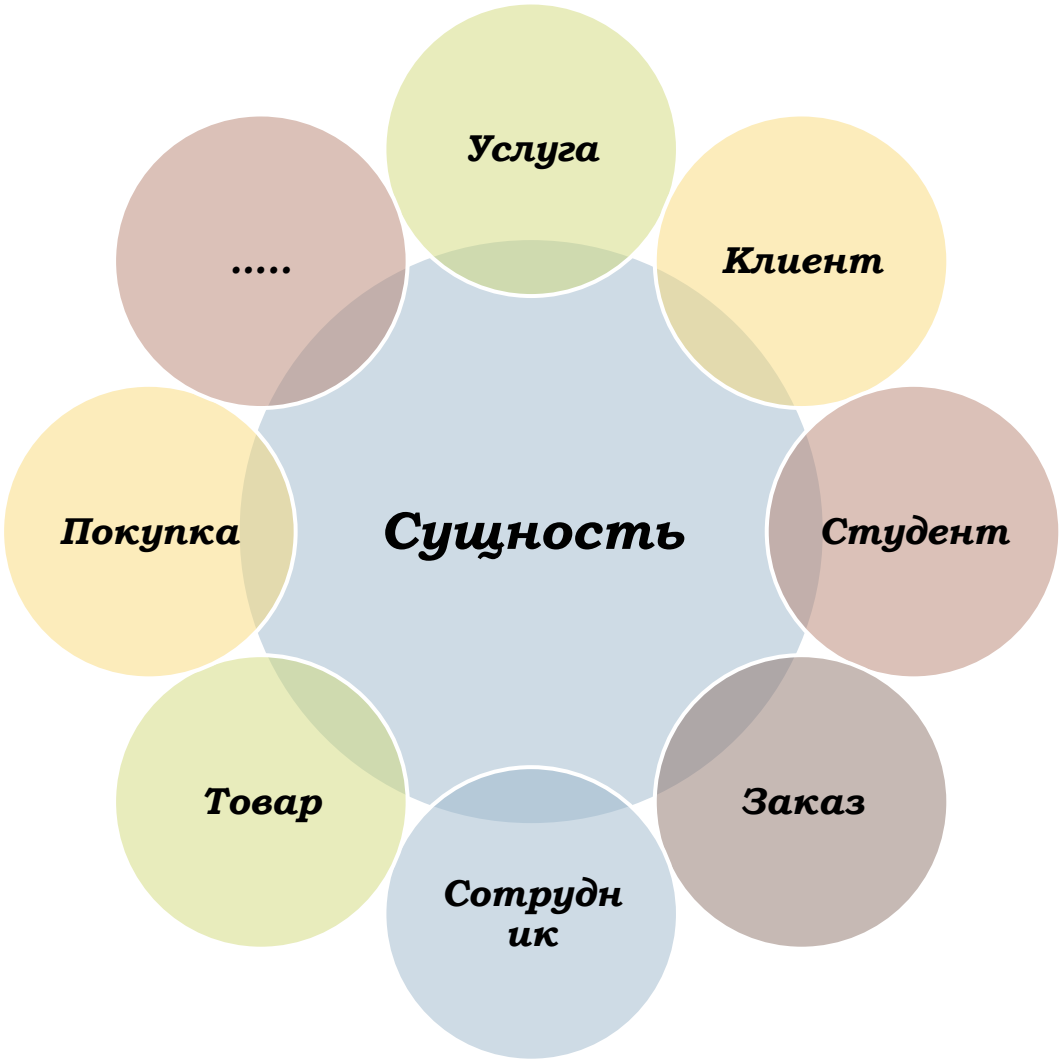
Диаграмма «сущность-связь» ***Entity-Relationship Diagram***

Теоретической базой построения информационной модели является теория баз данных типа «сущность-связь».

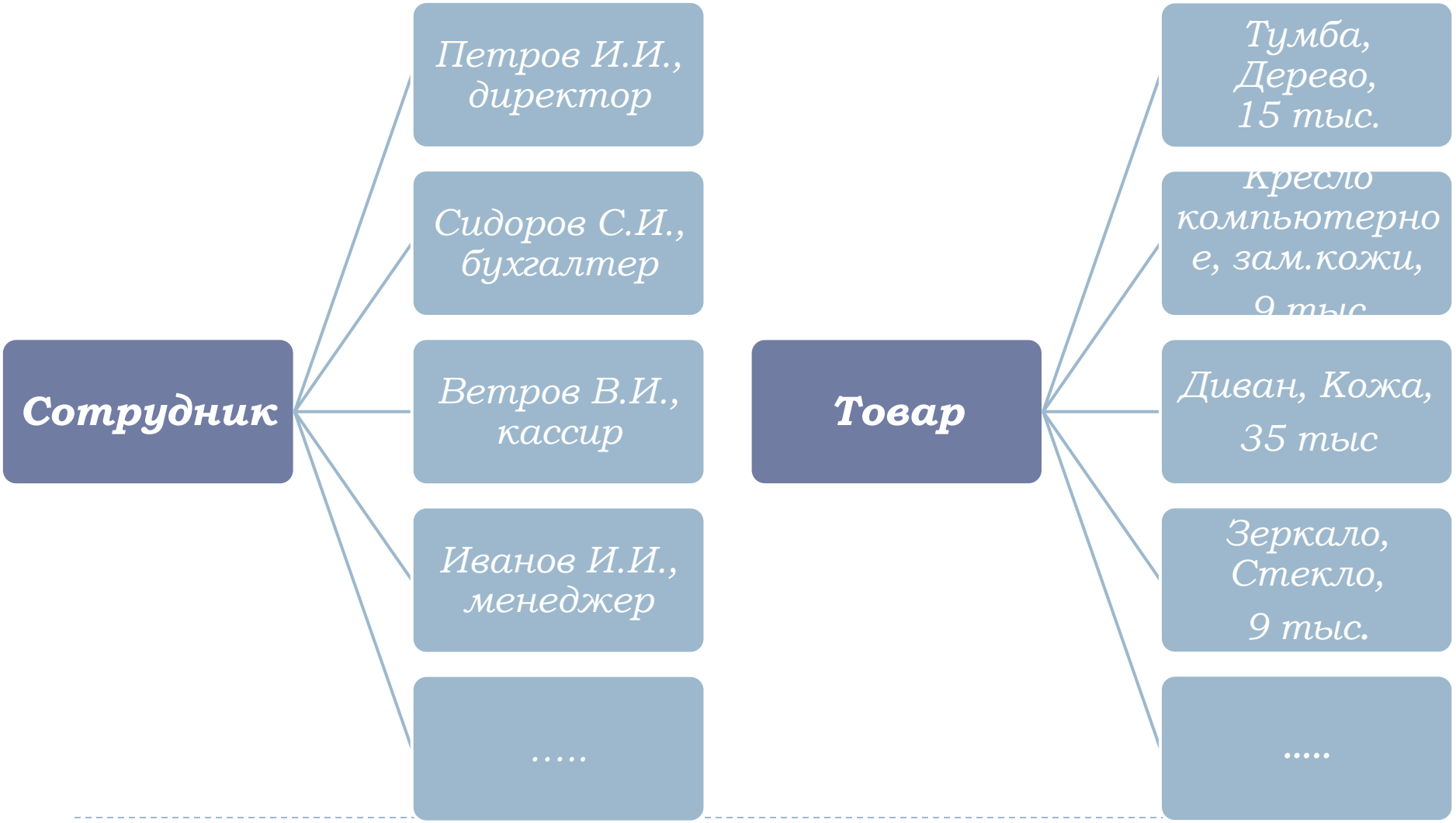
Основные понятия:

- Сущность
- Экземпляр сущности
- Атрибут
- Ключ
- Отношение

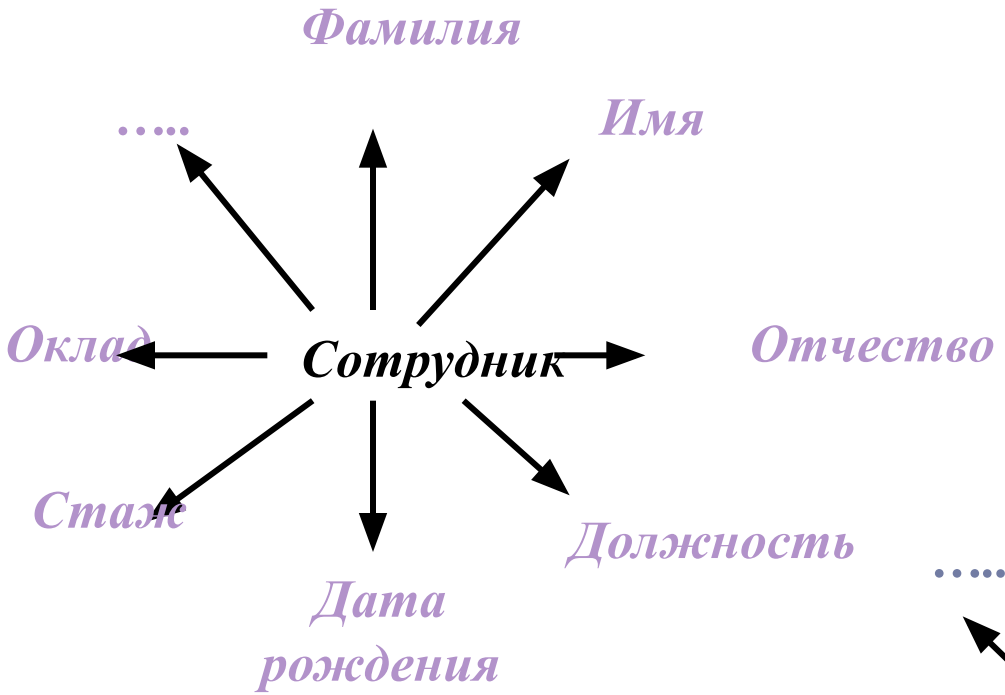
Сущности. Сущность-связь

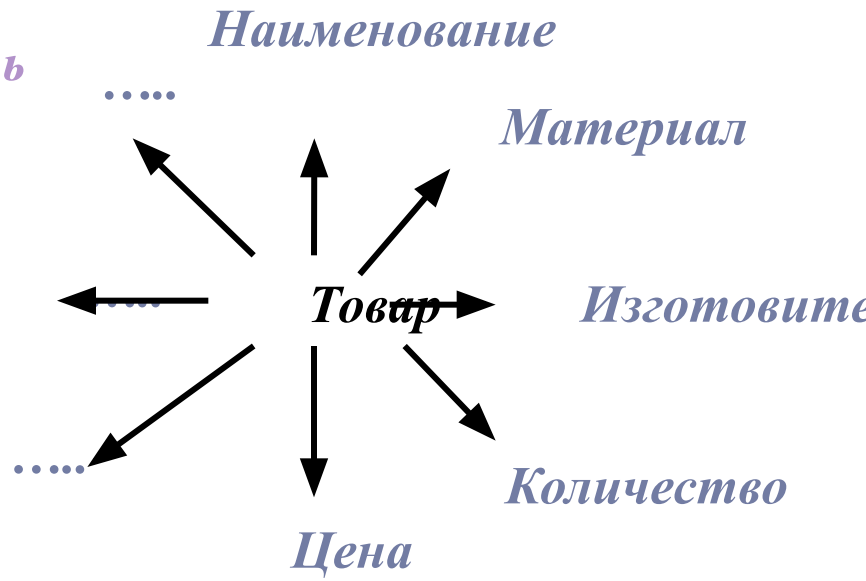


Экземпляры сущностей



Атрибуты


Фамилия
…..
Имя
Оклад
Сотрудник
Отчество
Стаж
Должность
Дата рождения


Наименование
…..
Материал
….
Товар
Изготовите
…..
Количество
Цена

Отношения

- Заказ включает Товар
- Товар входит в Заказ
- Покупатель осуществляет Покупку
- Продавец оформляет Покупку

Правила определения сущности

1. Сущность должна иметь уникальное имя и именоваться существительным в единственном числе.

Пример: *Студент, Кредитная карта, Договор,…*

2. Сущность обладает одним или несколькими атрибутами, которые ей либо принадлежат, либо наследуются через отношения.
3. Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый образец сущности (экземпляр) и называются ключом (составным ключом).

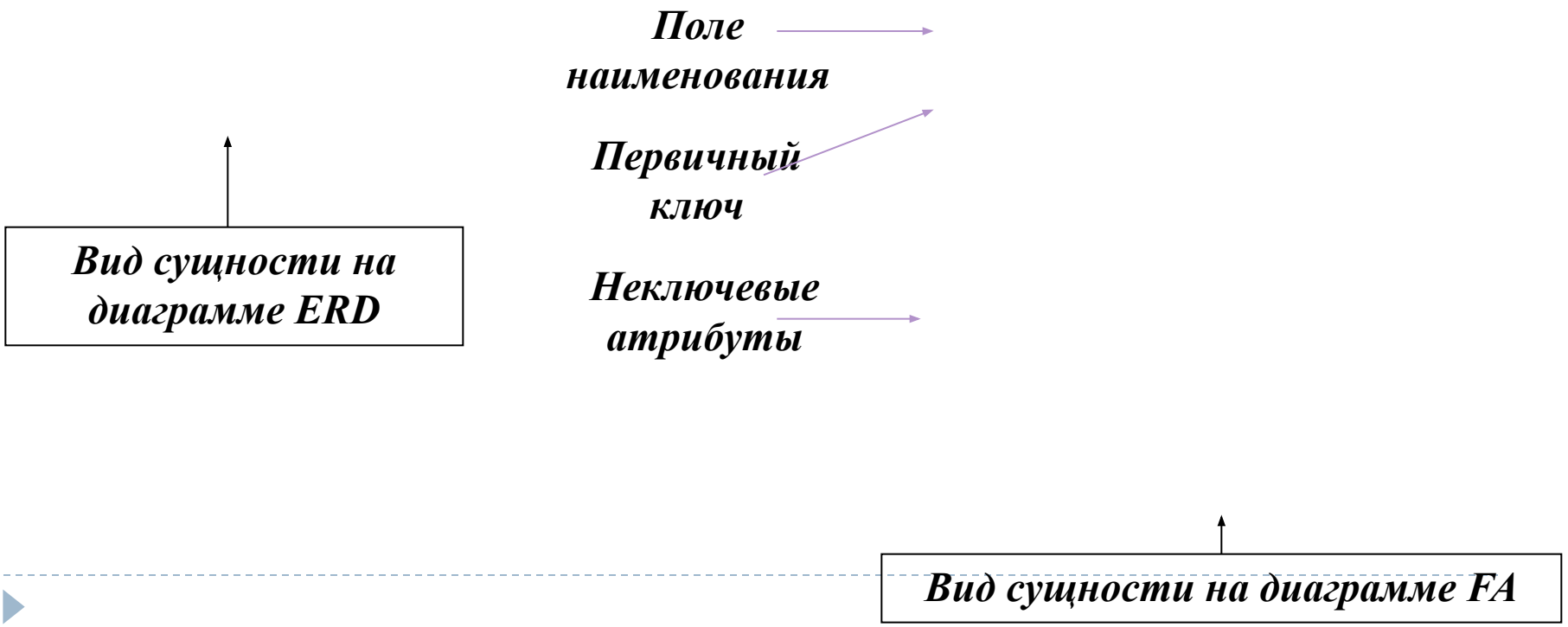
Правила определения сущности

4. Каждая сущность может обладать любым количеством отношений с другими сущностями.
5. Если внешний ключ целиком используется в составе первичного ключа, то сущность является зависимой от идентификатора.
6. В нотации IDEF1X сущность изображается в виде прямоугольника, в зависимости от уровня представления данных могут быть некоторые различия

Графическое представление сущности

Различают следующие уровни представления сущности:
1)

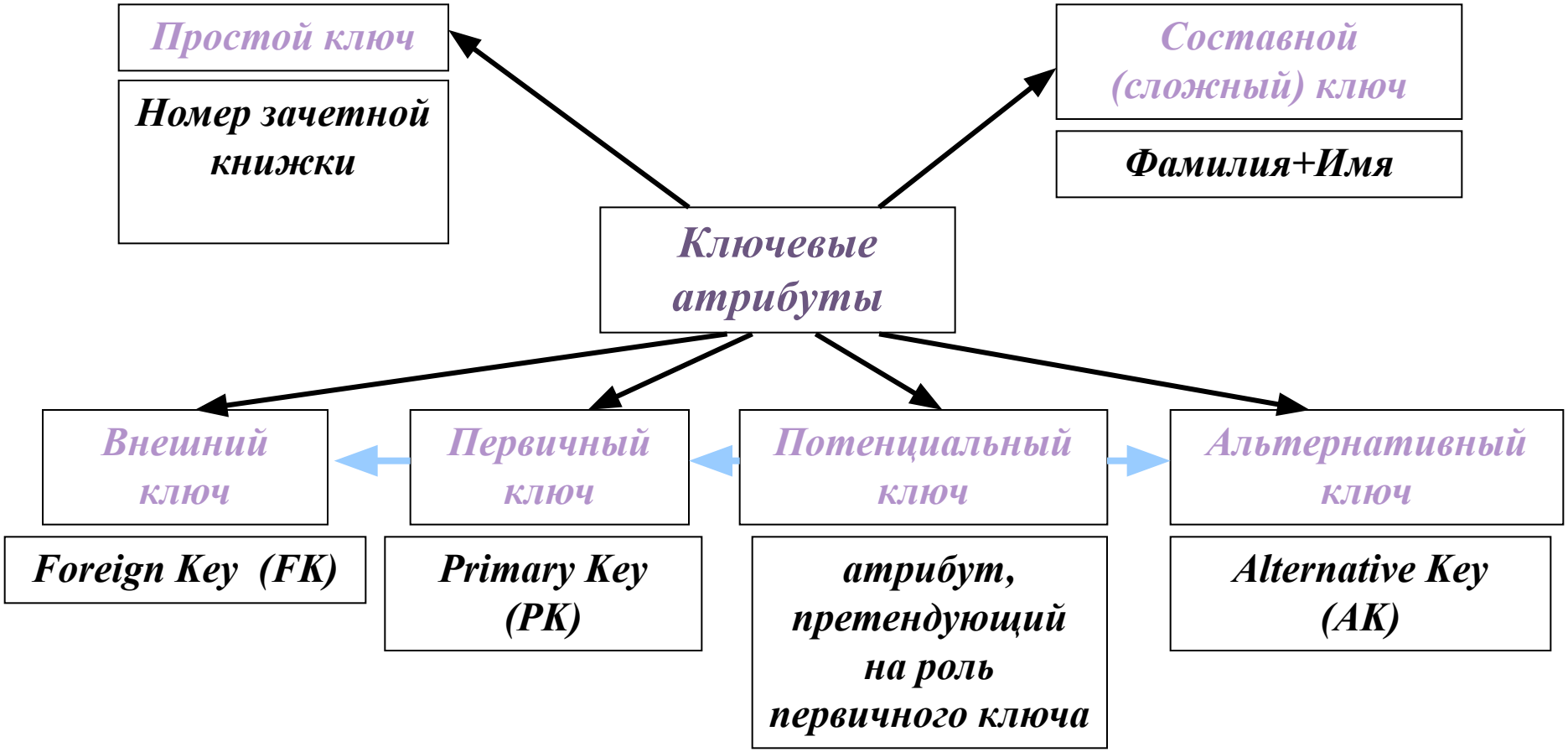
2) модель данных, основанная на ключах (*KB*),
3) полная атрибутивная модель (*FA*)



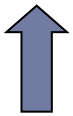
Правила определения атрибутов

1. Каждый атрибут каждой сущности обладает уникальным именем.
2. Сущность может обладать любым количеством атрибутов.
3. Различают собственные и наследуемые атрибуты. Собственные атрибуты являются уникальными в рамках модели. Наследуемые передаются от сущности-родителя при определении идентифицирующей связи.

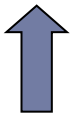
Ключевые атрибуты



Примеры ключевых атрибутов

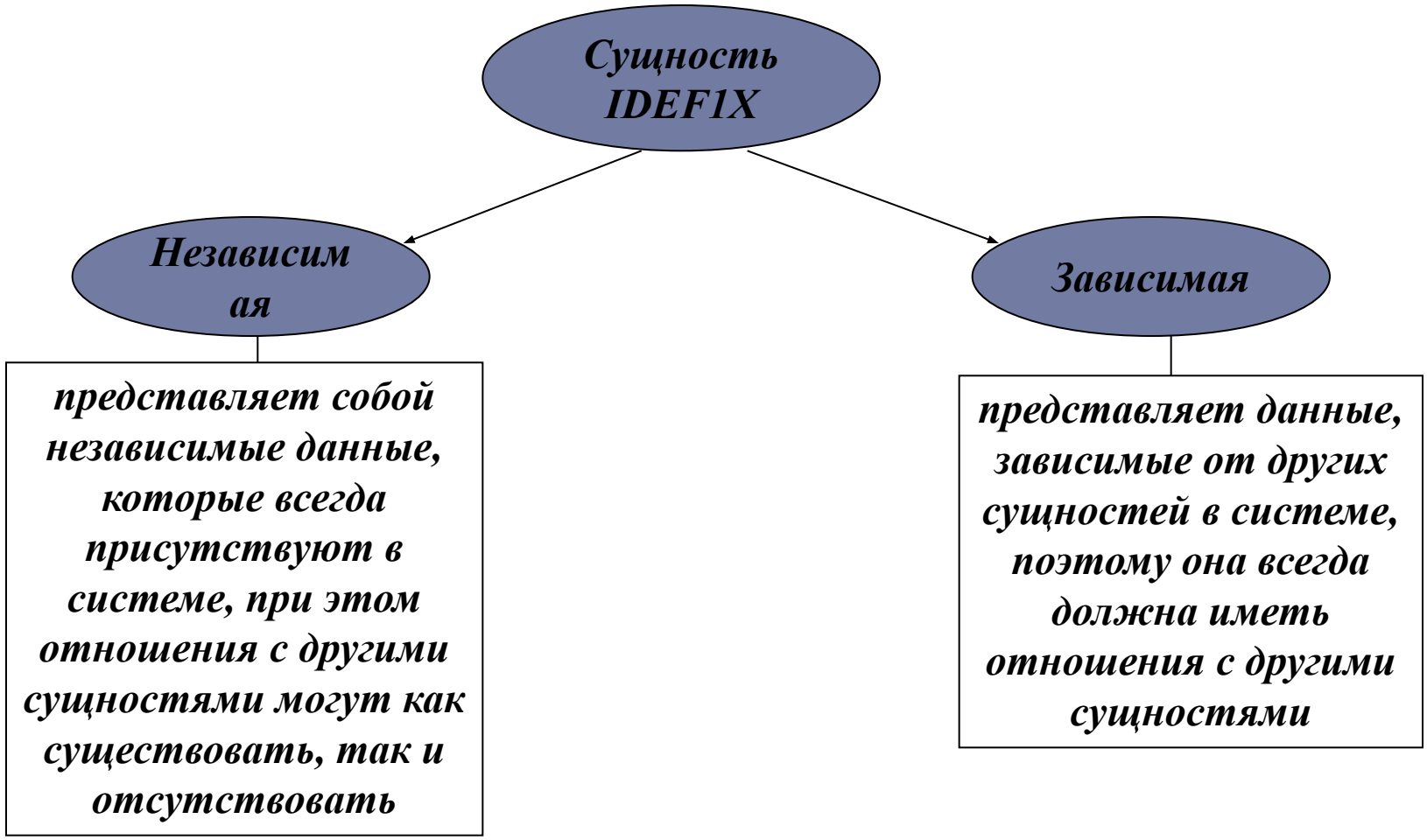


№_зачетнойКнижки – первичный простой ключ



ФИО+дата_рождения – первичный составной ключ;

Типы сущностей в IDEF1X



Типы сущностей в IDEF1X

Рис. Независимые от идентификации сущности

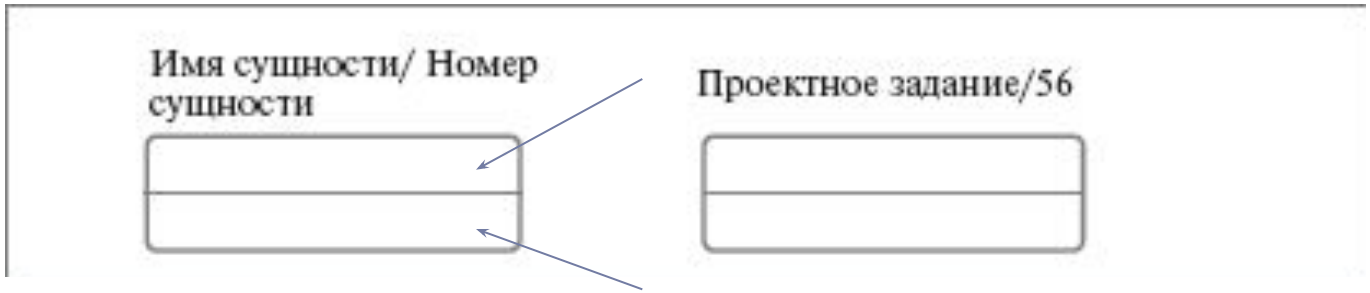


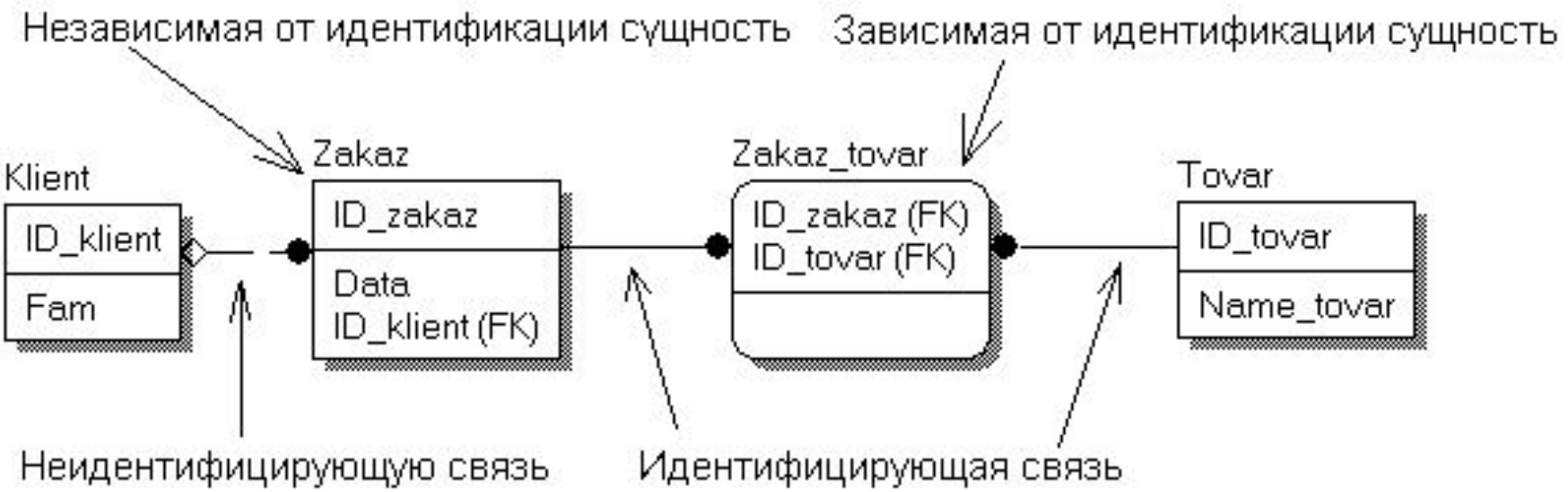
Рис. Зависимые от идентификации сущности

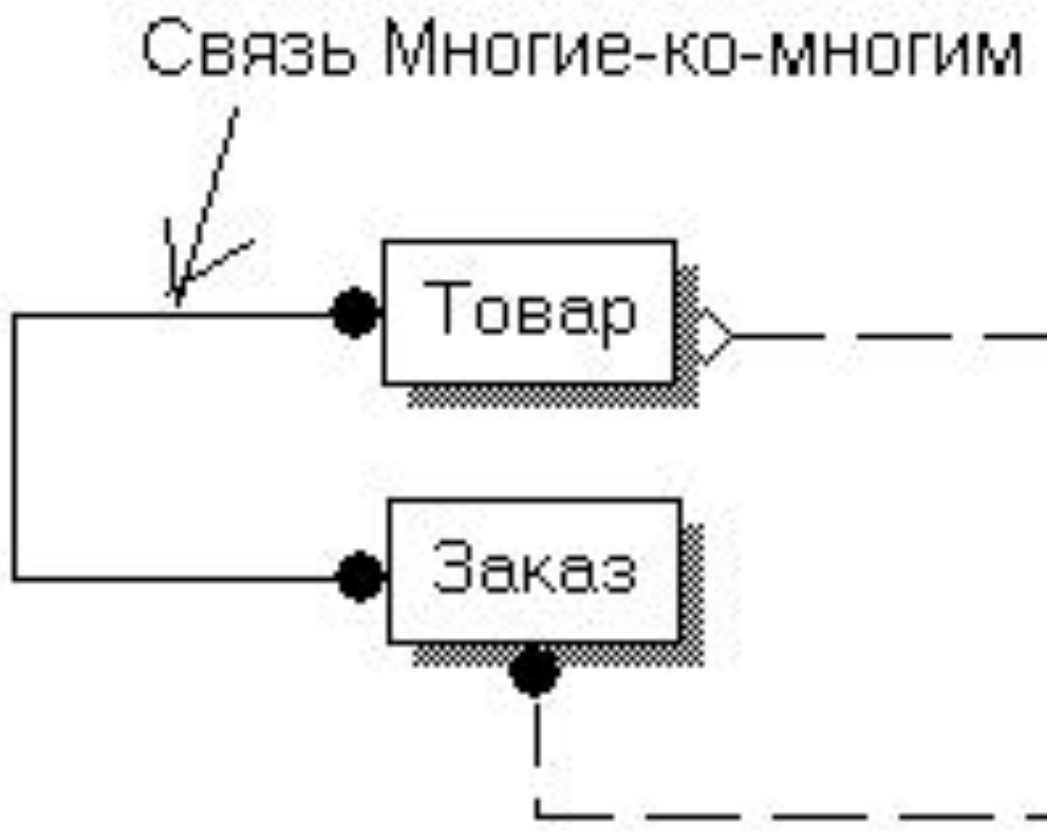
Виды отношений

а) идентифицирующая связь
Сущность А1 однозначно определяет сущность А2. Ее первичный ключ наследуется в качестве первичного ключа сущностью А2 (внешний ключ)

б) неидентифицирующая связь
Сущность А1 связана с сущностью А2, но однозначно не определяет ее. Первичный ключ сущности А1 наследуется в качестве неключевого атрибута сущности А2

в) связь «многие-ко-многим» (неспецифическая). Сущности А1 и А2 имеют формальную связь, но наследования атрибутов не происходит.


Независимая от идентификации сущность
Зависимая от идентификации сущность
Klient
ID_klient
Fam
Zakaz
ID_zakaz
Data
ID_klient (FK)
Zakaz_tovar
ID_zakaz (FK)
ID_tovar (FK)
Tovar
ID_tovar
Name_tovar
Неидентифицирующую связь
Идентифицирующая связь


Связь Многие-ко-многим
Товар
Заказ

4 типа мощности связей

а) общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности

б) когда одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней (0 исключается).

4 типа мощности связей

в) когда одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности.

г) когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

Контрольная работа по темам 1,2,3

Тема 4. Модели данных

Как данные хранить?

Как эффективно манипулировать данными?

Возникновение термина

- *Понятие модели данных предложено в 1969 г. **Эдгаром Коддом («Тед» Кодд)** для описания реляционного подхода к организации БД (см. тему «Реляционные базы данных»).*
- *Понятие модели данных оказалось удобным и для реляционно-независимого представления и сопоставления других подходов.*

Понятие модели данных

- В классической теории баз данных, **модель данных** есть формальная теория представления и обработки данных в системе управления базами данных.

Модели данных

- ***Иерархическая***
- ***Сетевая***
- ***Реляционная***
- ***Объектно-ориентированная***
- ***Документ-ориентированная***
- ***Хранилища «ключ-значение»***
- ***Графовая***
- ***Колоночная***
- ***….***

Рейтинг СУБД

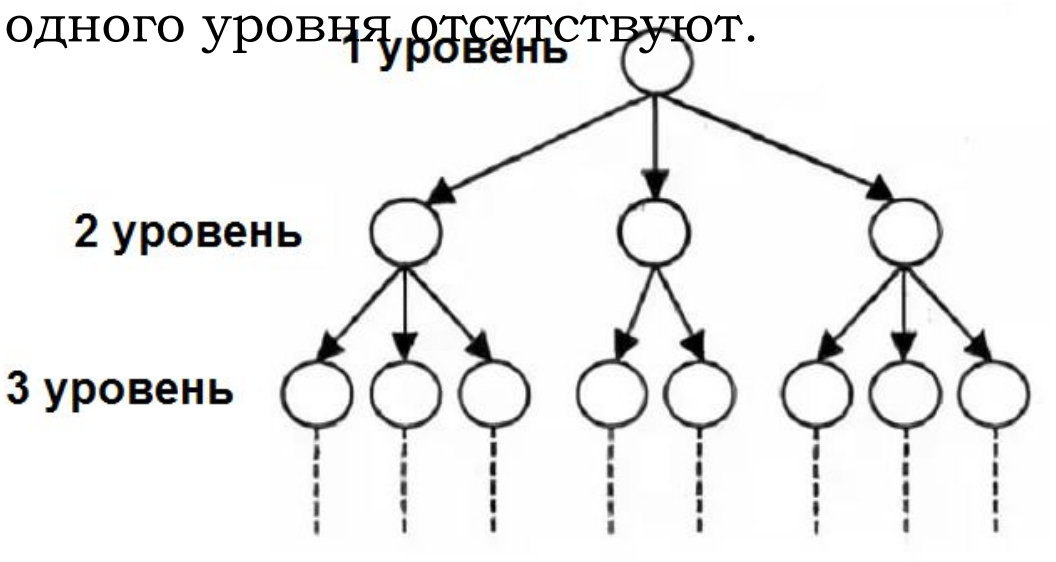
Место	СУБД	Модель данных
1	MySQL	реляционная
2	PostgreSQL	реляционная
3	MS SQL Server	реляционная
4	MongoDB	документ-ориентированная
5	SQLite	реляционная
6	Oracle Database	реляционная
7	Firebird	реляционная
8	CouchDB	документ-ориентированная
	DB2	реляционная
9	MariaDB	реляционная
10	RavenDB	документ-ориентированная
	Redis	хранилище ключ-значение
	SAP ASE	реляционная

Иерархическая модель данных

Иерархическая модель данных основана на графическом способе связей данных, и схема взаимосвязей объектов имеет вид перевернутого дерева.

- Доступ к информации возможен только по вертикальной схеме, начиная с корня дерева.
- Поиск данных происходит по одной из ветвей дерева.
- Связи между вершинами одного уровня отсутствуют.

- К основным понятиям модели относятся: уровень, элемент (узел), связь.
- Узел — это совокупность атрибутов данных, описывающих некоторый объект.
- На схеме иерархического дерева узлы представляются вершинами графа.

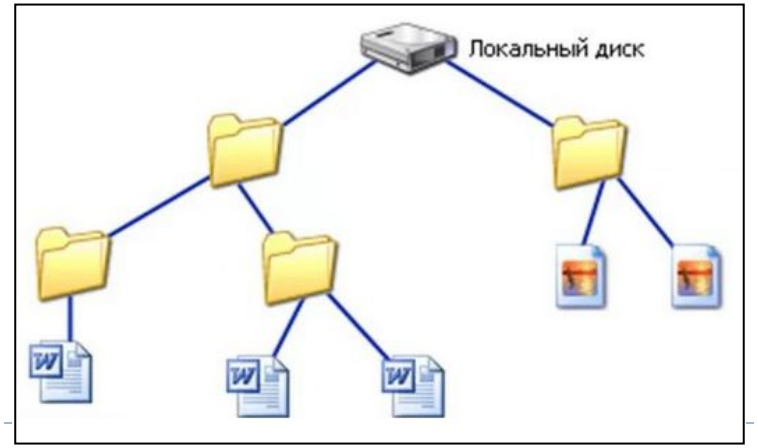


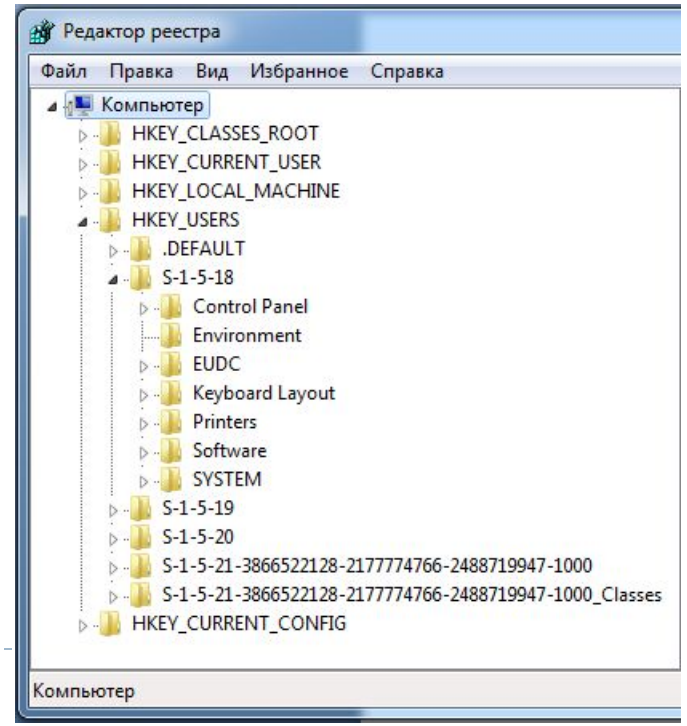
Иерархическая модель данных, примеры

1. Базы данных с иерархической моделью *одни из самых старых*, и стали первыми системами управления базами данных для мейнфреймов:

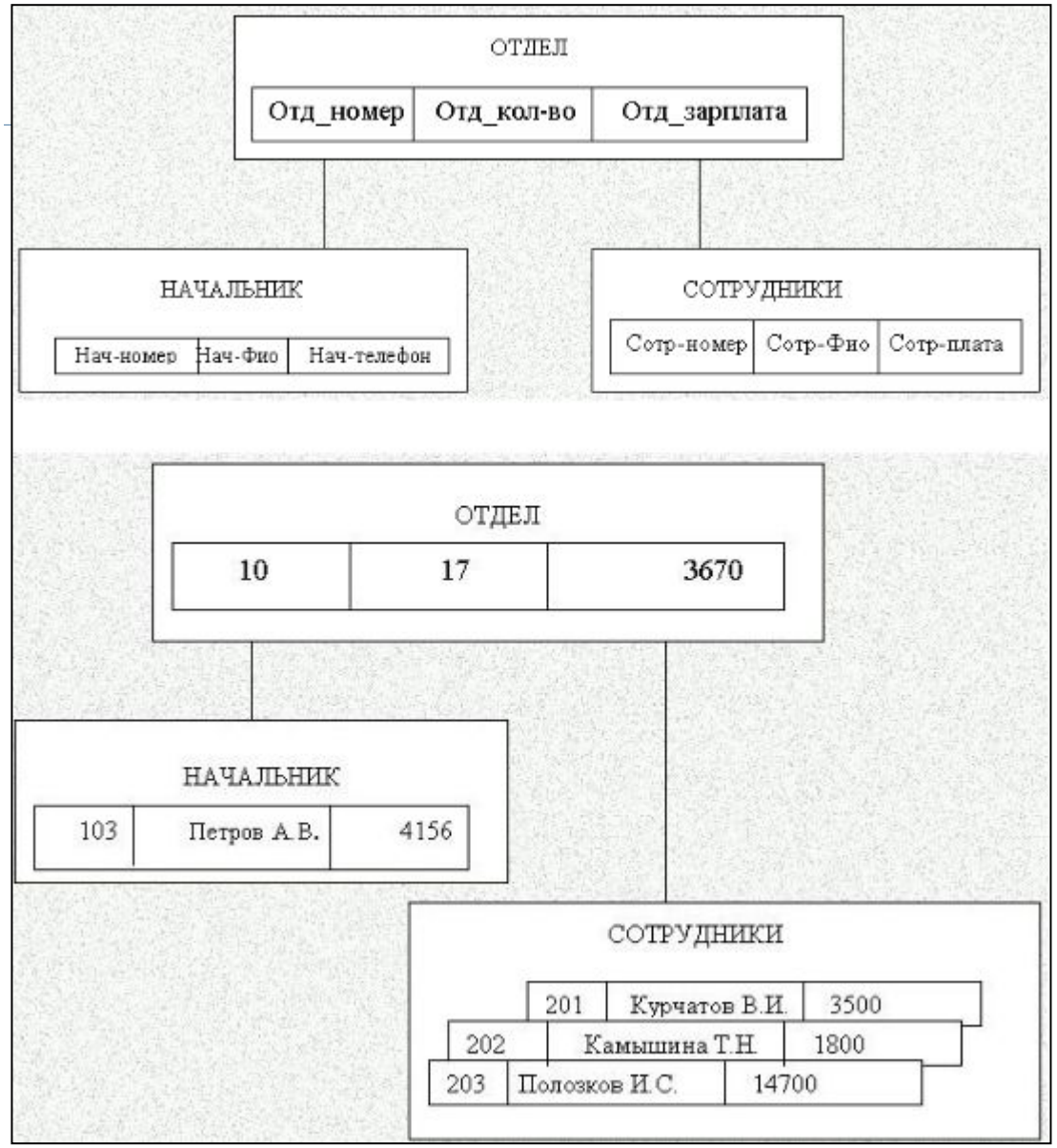
- **Information Management System** (IMS) фирмы IBM. Первая версия появилась в 1968 г.

2. Файловая структура
3. Реестр Windows



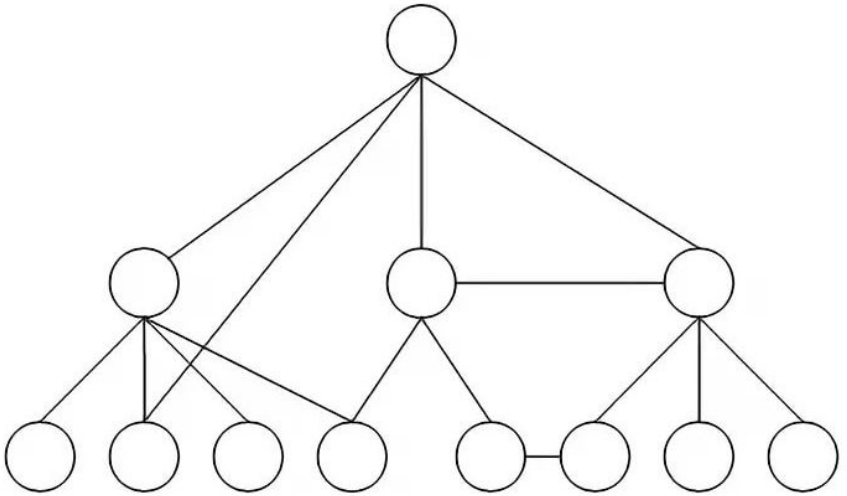


Иерархическая модель данных, примеры



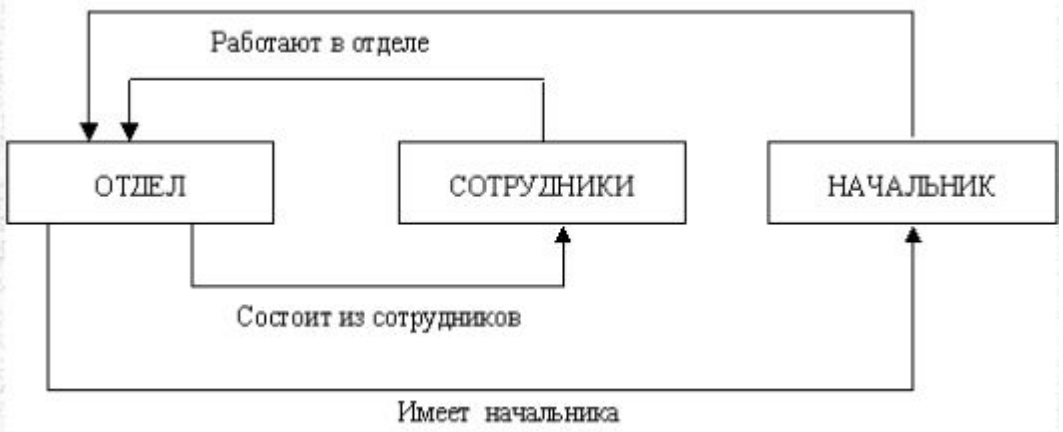
Сетевая модель данных

- Сетевой подход к организации данных является расширением иерархического подхода. В иерархических структурах запись-потомок должна иметь в точности одного предка, в сетевой структуре данных у потомка может иметься любое число предков (т.е. в них имеются указатели в обоих направлениях).
- К основным понятиям модели относятся: уровень, элемент (узел), связь.
- Узел — это совокупность атрибутов данных, описывающих некоторый объект.
- В сетевой структуре каждый элемент может быть связан с любым другим элементом.



Сетевая модель данных, примеры

- **IDS (Integrated Data Store)** компании General lectric — самая первая сетевая СУБД, разработанная Чарльзом Бахманом в 1960 г.



Реляционная модель данных

- Рассмотрено в теме «Реляционные базы данных»

Документ-ориентированная модель данных

- Документ-ориентированная модель специально предназначена для хранения иерархических структур данных – документов.
- Документ – набор атрибутов (ключ и соответствующее ему значение).
  - Документ может быть вложен в документ.
- Представление данных – JSON или XML формат.

Документ-ориентированная модель данных, примеры

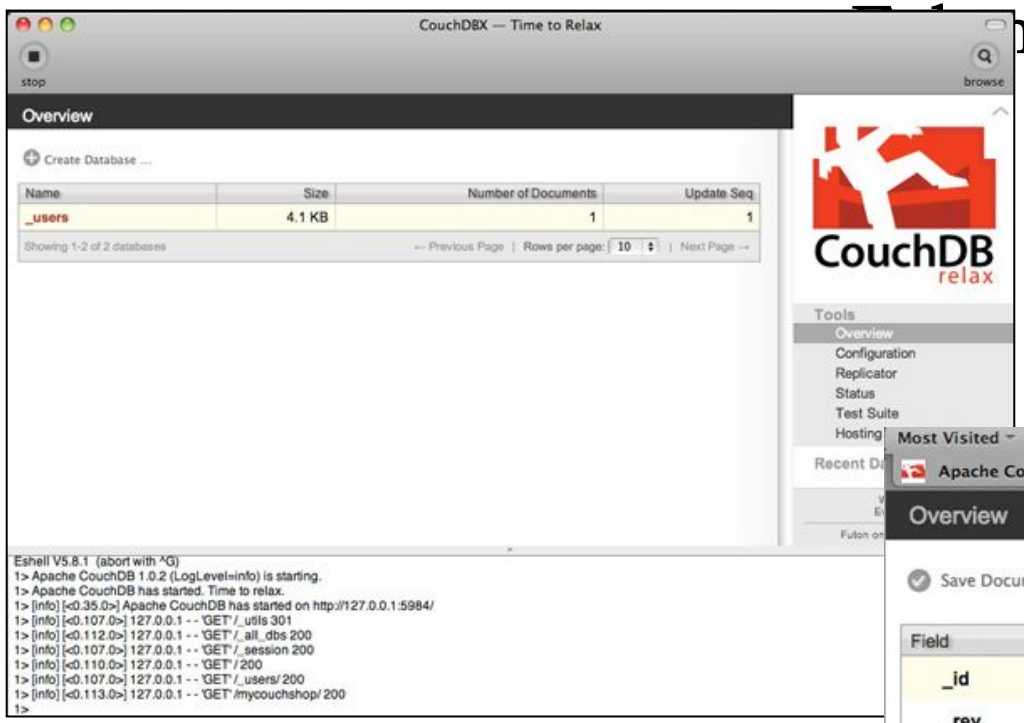
- **Документ-ориентированные** базы данных применяются в системах управления содержимым, издательском деле, документальном поиске и т.п.

Примеры СУБД:

- **CouchDB**
- Couchbase
- MarkLogic
- MongoDB
- eXist
- Berkeley DB

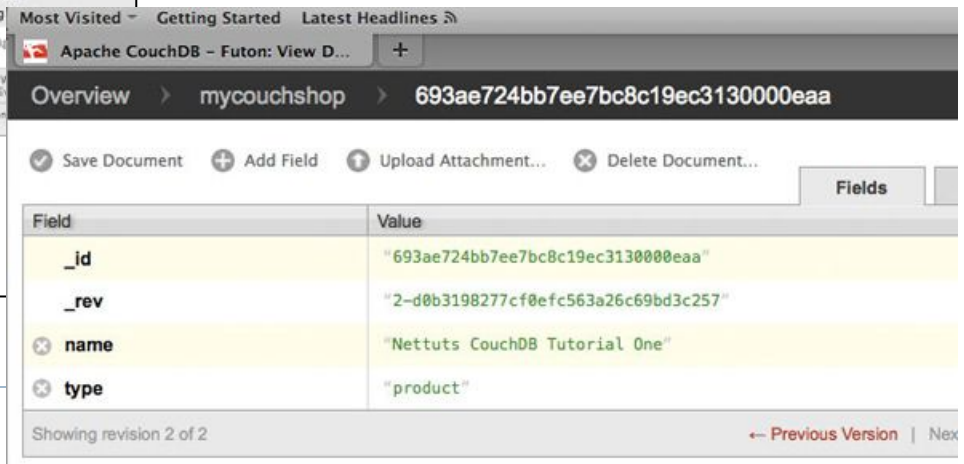
CouchDB

- **CouchDB** (база данных, которая ориентируется на хранение данных в формате JSON, написанная на языке ng)



***JSON** (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript.*

Если нужно с сервера взять объект с данными и передать его клиенту, то в качестве промежуточного формата – для передачи по сети, почти всегда используют именно его.

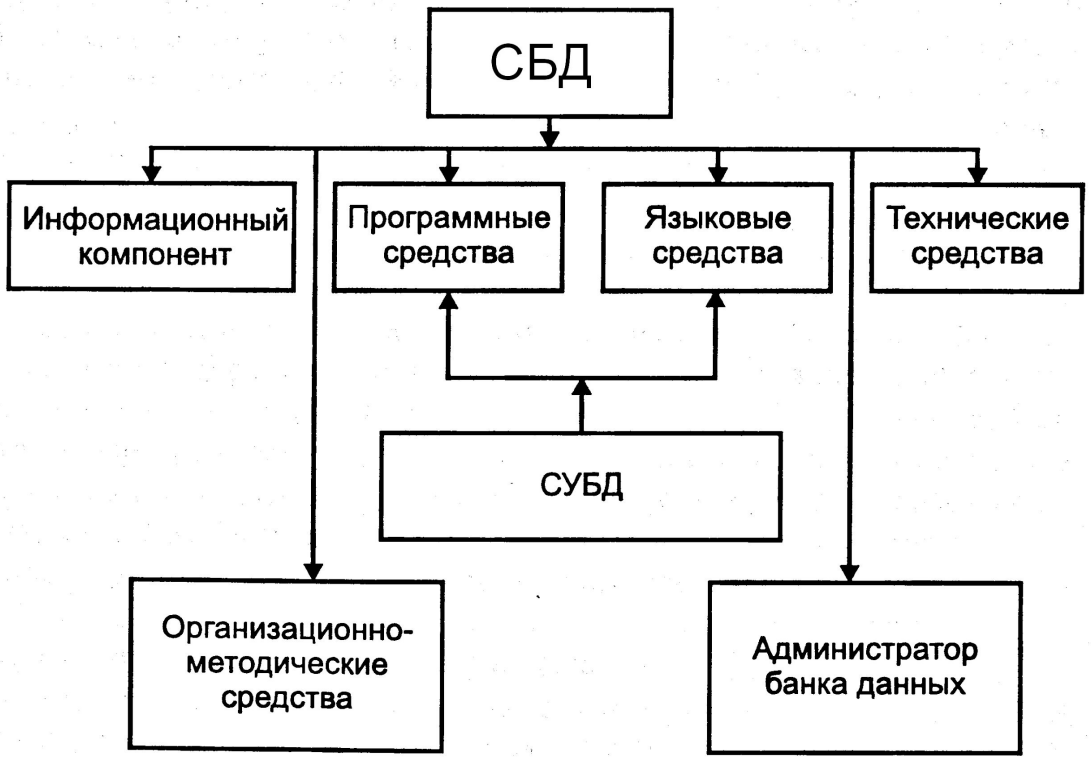


Задание.

- Объектно-ориентированная модель: определение, применение, примеры
- Хранилища «ключ-значение»: определение, применение, примеры

Тема 5. Системы управления базами данных

- *Система управления базами данных (СУБД)* – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользо



5.1. Функциональные возможности системы управления базами данных

Определение данных (описание структуры баз данных), поддержка словаря данных

- Описание наименований и типов полей
- Формат полей
- Критерии проверки вводимых данных
- Задание связей между таблицами

СЛОВАРЬ ДАННЫХ

Часть СУБД, определяющая структуру пользовательских данных и то, как они могут использоваться.

Подсистема словаря следит за определением всех элементов данных базы.

Словарь отслеживает отношения, существующие между различными группами данных.

Словарь поддерживает индексы, служащие для быстрой сортировки и обращения к данным.

Словарь отслеживает установки формата вывода данных.

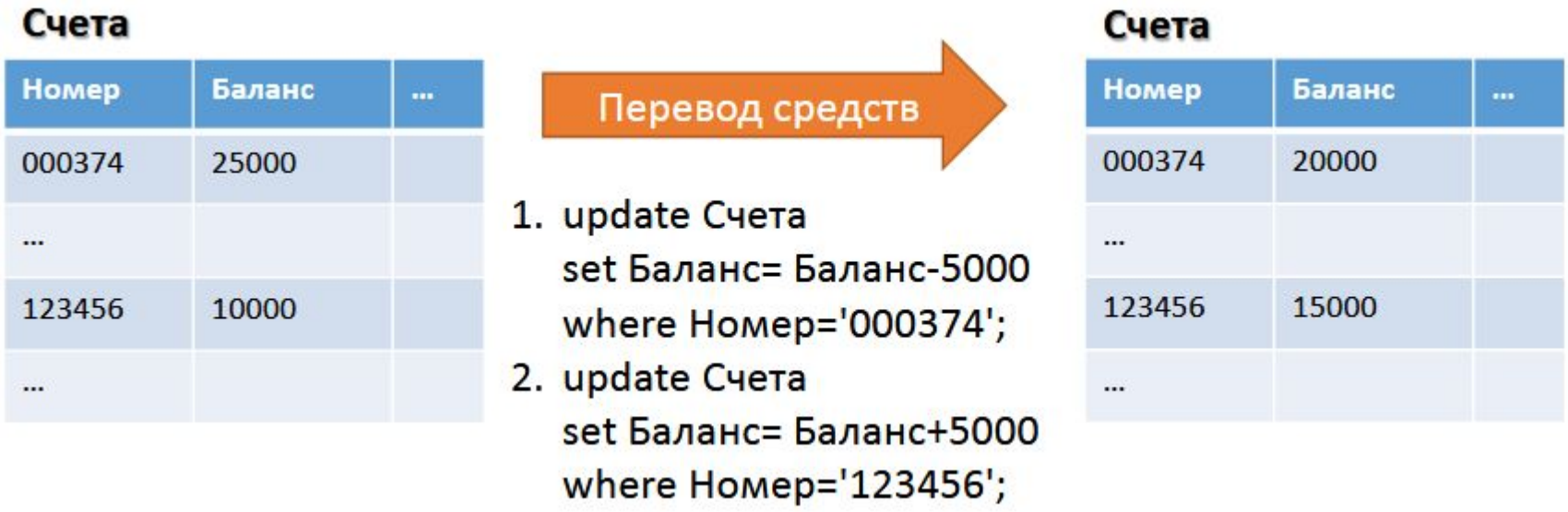
Обработка данных, поддержка транзакций

- добавление в таблицу одной или нескольких записей;
- удаление из таблицы одной или нескольких записей;
- обновление значений некоторых полей в одной или нескольких записях;
- поиск одной или нескольких записей, удов

Для выполнения этих операций используется механизм запросов. Язык структурированных запросов (SQL – Structured Query Language)

Транзакция

- Транзакция – последовательность операций над базой данных, рассматриваемых СУБД как единое целое.
- Транзакции необходимы для поддержания логической непротиворечивости информации, хранящейся в базе данных



Управление данными

- Защита от несанкционированного доступа
- Поддержка многопользовательского режима работы с данными
- Обеспечение целостности и согласованности данных
- Резервное копирование данных и восстановление данных после сбоев

Примеры ограничений целостности

- Возраст может принимать значения 16..65
- ФИО не может быть пустым
- Пол может принимать значения 'М' или 'Ж‘
- Удаление записи из таблицы Отделы должно повлечь удаление связанных записей из таблицы Сотрудники
- Нельзя принять в отдел нового сотрудника, если средний возраст сотрудников этого отдела будет превышать 45 лет

Резервное копирование и восстановление данных

- СУБД поддерживает **журнал транзакций** – файл, в котором регистрируются изменения, вносимые транзакциями в базу данных.
  - Запись об изменениях производится до фактического выполнения этих изменений (принцип WAL, Write Ahead Log).
- Используя журнал транзакций, СУБД восстанавливает базу данных после программных сбоев.
- СУБД поддерживает резервное копирование базы данных и журнала транзакций для восстановления данных после аппаратных сбоев.

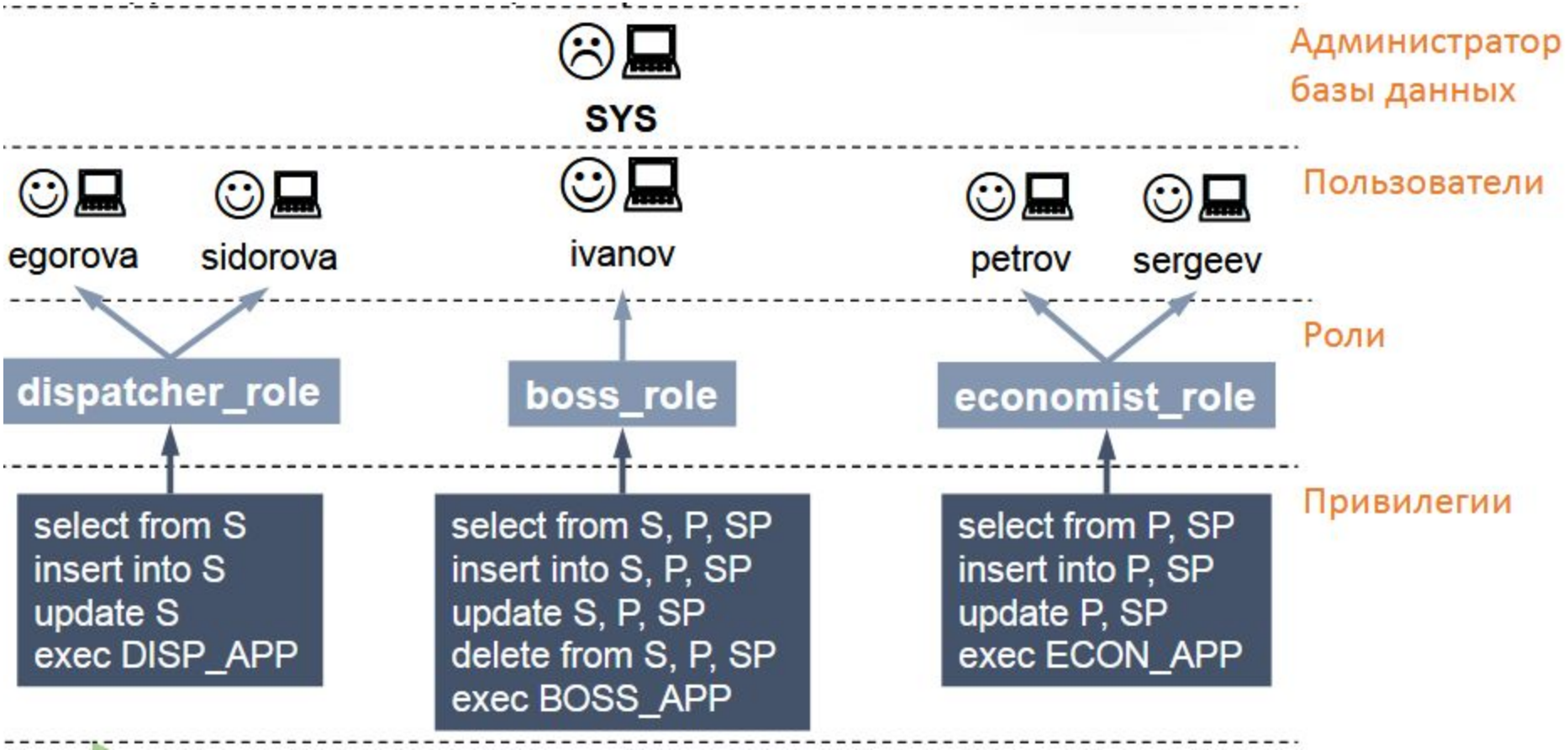
Пример. Выборка из журнала транзакций операций на вставку строк

```
SELECT [Current LSN],
       Operation,
       Context,
       [Transaction ID],
       [Begin time]
       FROM sys.fn_dblog
   (NULL, NULL)
  WHERE operation IN
   ('LOP_INSERT_ROWS');
```

	Current LSN	Operation	Context	Transaction ID	Begin time
1	00000074:0000019e:0002	LOP_INSERT_ROWS	LCX_CLUSTERED	0000:0000922d	NULL
2	00000074:0000019e:0003	LOP_INSERT_ROWS	LCX_INDEX_LEAF	0000:0000922d	NULL
3	00000074:0000019e:0004	LOP_INSERT_ROWS	LCX_INDEX_LEAF	0000:0000922d	NULL
4	00000074:0000019e:0005	LOP_INSERT_ROWS	LCX_INDEX_LEAF	0000:0000922d	NULL
5	00000074:0000019e:0008	LOP_INSERT_ROWS	LCX_CLUSTERED	0000:0000922d	NULL
6	00000074:0000019e:0009	LOP_INSERT_ROWS	LCX_INDEX_LEAF	0000:0000922d	NULL
7	00000074:0000019e:000a	LOP_INSERT_ROWS	LCX_CLUSTERED	0000:0000922d	NULL

Безопасность данных

- СУБД обеспечивает безопасность базы данных – защищает данные от несанкционированных пользователей.



5.2. Свойства СУБД

Универсальность.	• ***СУБД должна обладать мощными средствами поддержки концептуальной модели данных для отображения пользовательских логических представлений***
Совместимость	• ***СУБД должна сохранять работоспособность при развитии программного и аппаратного обеспечения.***
Непротиворечивость данных	• ***В отличие от файловых систем база данных должна представлять собой единую совокупность интегрированных данных***
Защита данных	• ***СУБД должна обеспечивать защиту от несанкционированного доступа***
Целостность данных.	• ***СУБД должна предотвращать нарушения базы данных пользователями.***
Независимость данных	• ***Различают логическую и физическую независимость данных. Под независимостью понимают возможность изменения логического (физического) представления БД, не меняя ее физического (логического) представления.***

Тема 6. Реляционные базы данных и СУБД

Должны знать:

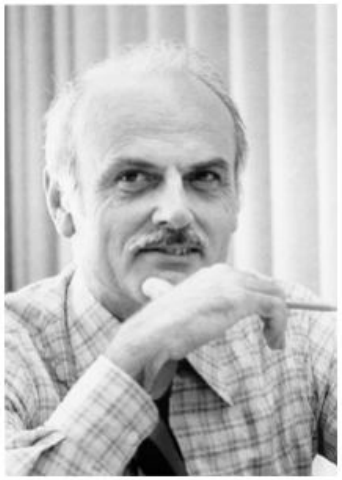
- Реляционная БД
- Реляционная СУБД
- Реляционная модель данных
- Эдгар Кодд
- 12 правил Кодда
- Принципы реляционной модели
- Отношение
- Скаляр
- Замыкание
- Кортеж
- Атрибут
- Сущность
- Домен
- Связь

6.1. Понятие реляционной БД и СУБД

- **Реляционная база данных** — база данных, основанная на реляционной модели данных. .
- **Реляционная СУБД**– это **СУБД**, предназначенная для работы с реляционными базами данных.

6.2. Реляционная модель данных

- Реляционная модель данных основывается на математических принципах, вытекающих непосредственно из теории множеств и логики предикатов.
  - Эти принципы **впервые были применены в области моделирования данных в конце 1960-х гг**. доктором ***Е.Ф. Коддом***, в то время работавшим в IBM, а **впервые опубликованы - в 1970 г.**



Эдгар Франк Кодд
1923-2003

12 правил Кодда, которым должна соответствовать настоящая реляционная база данных

(13 правил, т.к. исчисление начинается с 0)

0. Реляционная СУБД должна быть способна полностью управлять базой данных через ее реляционные возможности.
1. **Информационное правило** – вся информация в реляционной БД (включая имена таблиц и столбцов) должна определяться строго как значения в таблицах.
2. **Гарантированный доступ** – любое значение в реляционной БД должно быть гарантированно доступно для использования через комбинацию имени таблицы, значения первичного ключа и имени столбца
3. **Поддержка пустых значений** (null value) – СУБД должна уметь работать с пустыми значениями (неизвестными или неиспользованными значениями), в отличие от значений по умолчанию и независимо для любых *доменов*.

12 правил Кодда (13 правил, т.к. исчисление начинается с 0)

4. **Онлайновый реляционный каталог** – описание БД и ее содержания должны быть представлены на логическом уровне как таблицы, к которым можно применять запросы, используя язык базы данных.
5. **Исчерпывающий язык управления данными** – по крайней мере, один из поддерживаемых языков должен иметь четко определенный синтаксис и быть всеобъемлющим. Он должен поддерживать описание структуры данных и манипулирование ими, правила целостности, авторизацию и транзакции.
6. **Правило обновления представлений** (views) – все представления, теоретически обновляемые, могут быть обновлены через систему.
7. **Вставка, обновление и удаление** – СУБД поддерживает не только запрос на отбор данных, но и вставку, обновление и удаление

12 правил Кодда (13 правил, т.к. исчисление начинается с 0)

8. **Физическая независимость данных** – на программы-приложения и специальные программы логически не влияют изменения физических методов доступа к данным и структур хранилищ данных.
9. **Логическая независимость данных** – на программы-приложения и специальные программы логически не влияют, в пределах разумного, изменения структур таблиц.
10. **Независимость целостности** – язык БД должен быть способен определять правила целостности. Они должны сохраняться в онлайновом справочнике, и не должно существовать способа их обойти.
11. **Независимость распределения** – на программы-приложения и специальные программы логически не влияет, первый раз используются данные или повторно.
12. **Неподрывность** – невозможность обойти правила целостности, определенные через язык базы данных, использованием языков низкого уровня

Основные принципы реляционной модели

С точки зрения **теории реляционных БД**, основные принципы реляционной модели на концептуальном уровне:

1. Все данные представляются в виде упорядоченной структуры, определенной в виде строк и столбцов и называемой ***отношением***;
2. Все значения являются **скалярами**. Это означает, что для любой строки и столбца любого отношения существует одно и только одно значение;
3. Все операции выполняются над целым отношением, и результатом их выполнения также является целое отношение. Этот принцип называется **замыканием**

Термин «отношение» (relation)

- Ввел термин: Эдгар Кодд
- Почему не таблица или др.?
  - *Кодд* выбрал термин "отношение" (relation), потому что, по его мнению, этот термин однозначен (в то время как, например, термин "таблица" имеет множество различных видов - таблица в тексте, электронная таблица и пр.).

Термин «отношение» (relation)

- Неверное мнение: *реляционная модель* названа так потому, что она определяет связи между таблицами.
- Верное утверждение: *реляционная модель* названа так потому, что название этой модели происходит от отношений – реляций (таблиц базы данных), лежащих в ее основе.

Термин «отношение» (relation)

Термины «кортеж», «атрибут»

- Каждая строка, содержащая данные, называется ***кортежем***, каждый столбец отношения называется ***атрибутом***
  - (на уровне практической работы с современными реляционными БД используются термины "запись" и "поле").

Термины «кортеж», «атрибут»

Термины «кортеж», «атрибут»

Элементы описания реляционной модели

- Сущности (См. тему «Проектирование БД»)
- Атрибуты (См. тему «Проектирование БД»)
- Домены
- Связи (См. тему «Проектирование БД»)

Спецификация атрибута

- Спецификация *атрибута* состоит из:
  - его названия
  - указания типа данных
  - множества значений (или домена), которые может принимать данный *атрибут*.

Термин «домен»

- ***Домен*** - это набор всех допустимых значений, которые может содержать *атрибут («вид» данных).*
- ***Домен*** – именованное множество скалярных значений одного типа.
- Скалярное (атомарное) значение не имеет внутренней структуры.

Пример.

- ФИО: строка[30] – скаляр
- ФИО: { строка[10], строка[10], строка[10] } – не скаляр

Термин «домен»

- **домен («вид» данных)≠ тип данных**
- **Тип данных** – это **физическое понятие** (которое реализовано средствами конкретной СУБД), а **домен** – логическое.

Пример.

- *Атрибут*: имя
- *Тип данных*: текстовый с определенной длиной
- *Домен*: определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, такие строки не могут начинаться с мягкого знака).

Структура реляционных данных

Термин РМД	Англ. термин	Неформальный термин
Отношение	Relation	Таблица
Кортеж	Tuple	Запись таблицы
Атрибут	Attribute	Поле таблицы
Домен	Domain	Тип данных у значений в столбце таблицы
Первичный ключ	Primary key	уникальный идентификатор записи

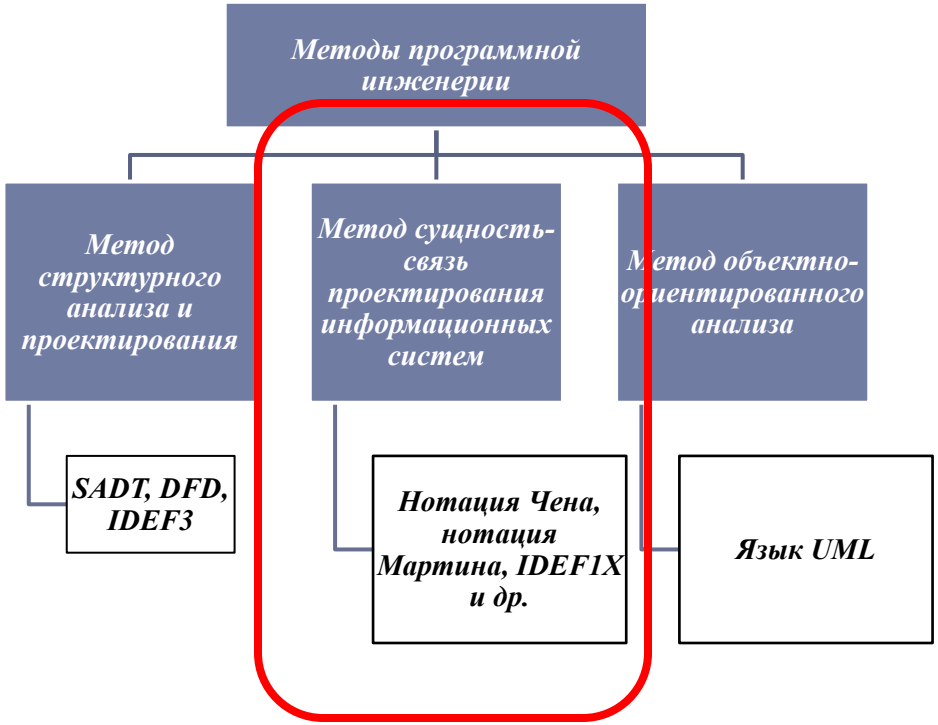
Записи, поля, индексация

- Каждая строка реляционной БД называется **записью**, каждый столбец – **полем.**
- **Поля** – это различные характеристики (атрибуты) объекта. Значения полей в одной строчке относятся к одному объекту.
- Записи в таблицах не повторяются. Их уникальность обеспечивается **первичным ключом**, содержащим набор полей, однозначно определяющих запись.
- Для быстрого поиска информации в базе данных создаются индексы по одному или нескольким полям таблицы. Значения индексов хранятся в упорядоченном виде и содержат ссылки на записи таблицы.

Реляционная модель данных

- Модель "сущность-связь" (EntityRelationship Model, ER-model) – один из наиболее известных и получивших широкое распространение методов семантического моделирования.

- ***Примечание.*** *Ранее рассмотрели на примере метода IDEF1X*



6.3. Нотации ER-диаграмм

Классическая нотация Питера Чена.

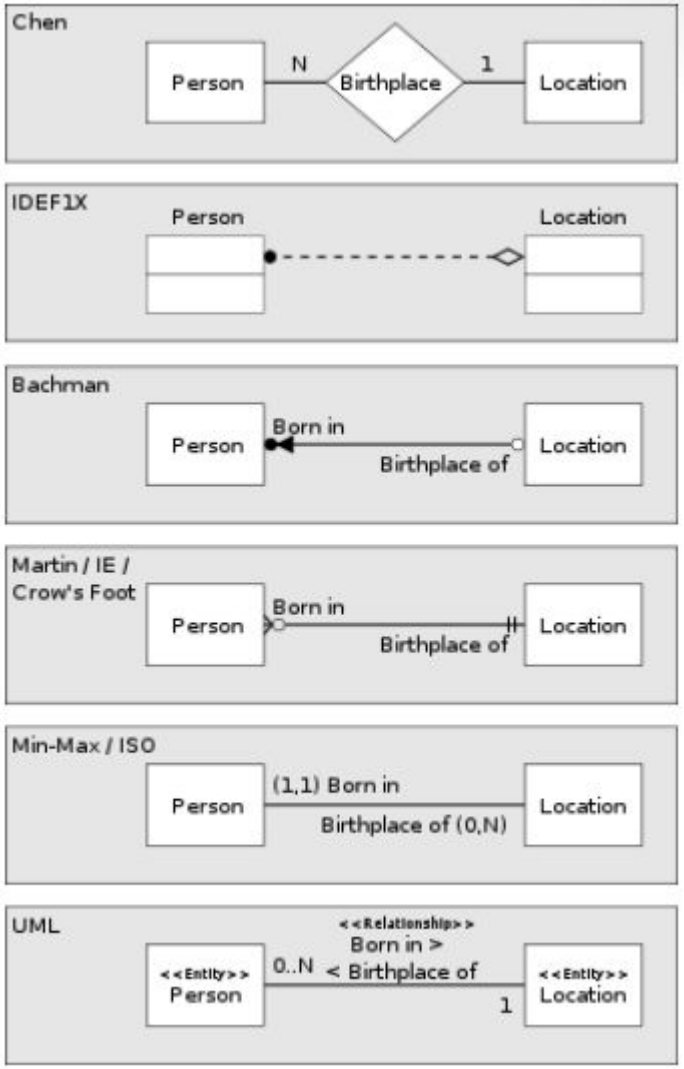
Нотация IDEFIX (Integration Definition for Information Modeling).

Нотация Ч. Бахмана.

Нотация Дж. Мартина ("вороньи лапки").

Нотация Ж.-Р. Абриаля (мин- макс).

Диаграммы классов UML.



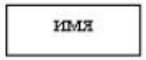
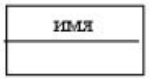
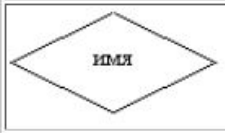
Реляционная модель данных (метод Питера Чена)

Элементы ER-модели (аналогично IDEF1X):

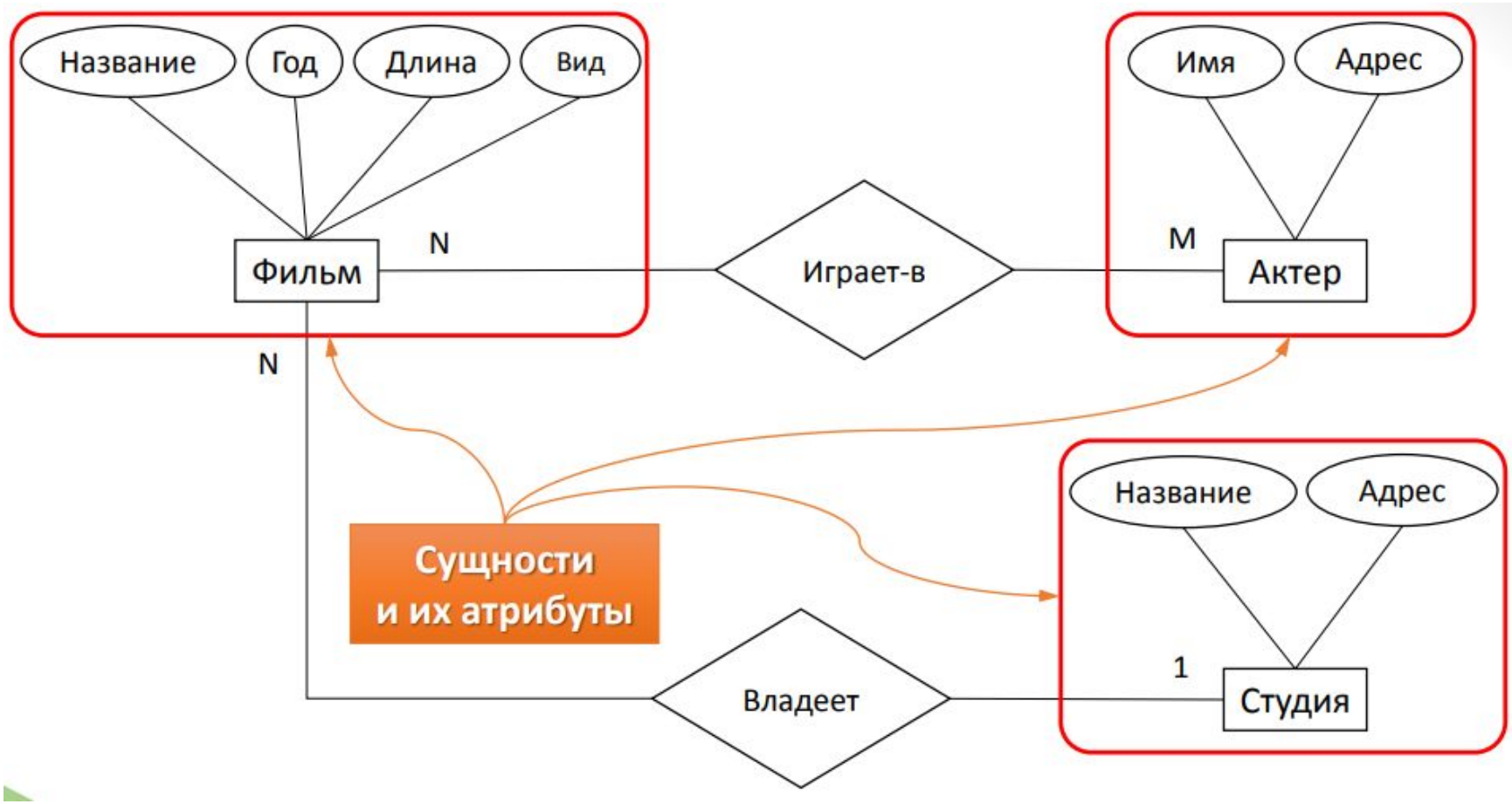
- **Сущность (Entity)** – реальный или абстрактный объект, имеющий существенное значение для предметной области.
- **Сущность** – любой различимый объект, информацию о котором необходимо хранить в базе данных.
- **Атрибут** – поименованная характеристика сущности.
- **Связь (relationship)** – это ассоциация, устанавливаемая между сущностями.
- **Мощность связи** – число экземпляров сущности, участвующих в связи.

Питер Чен

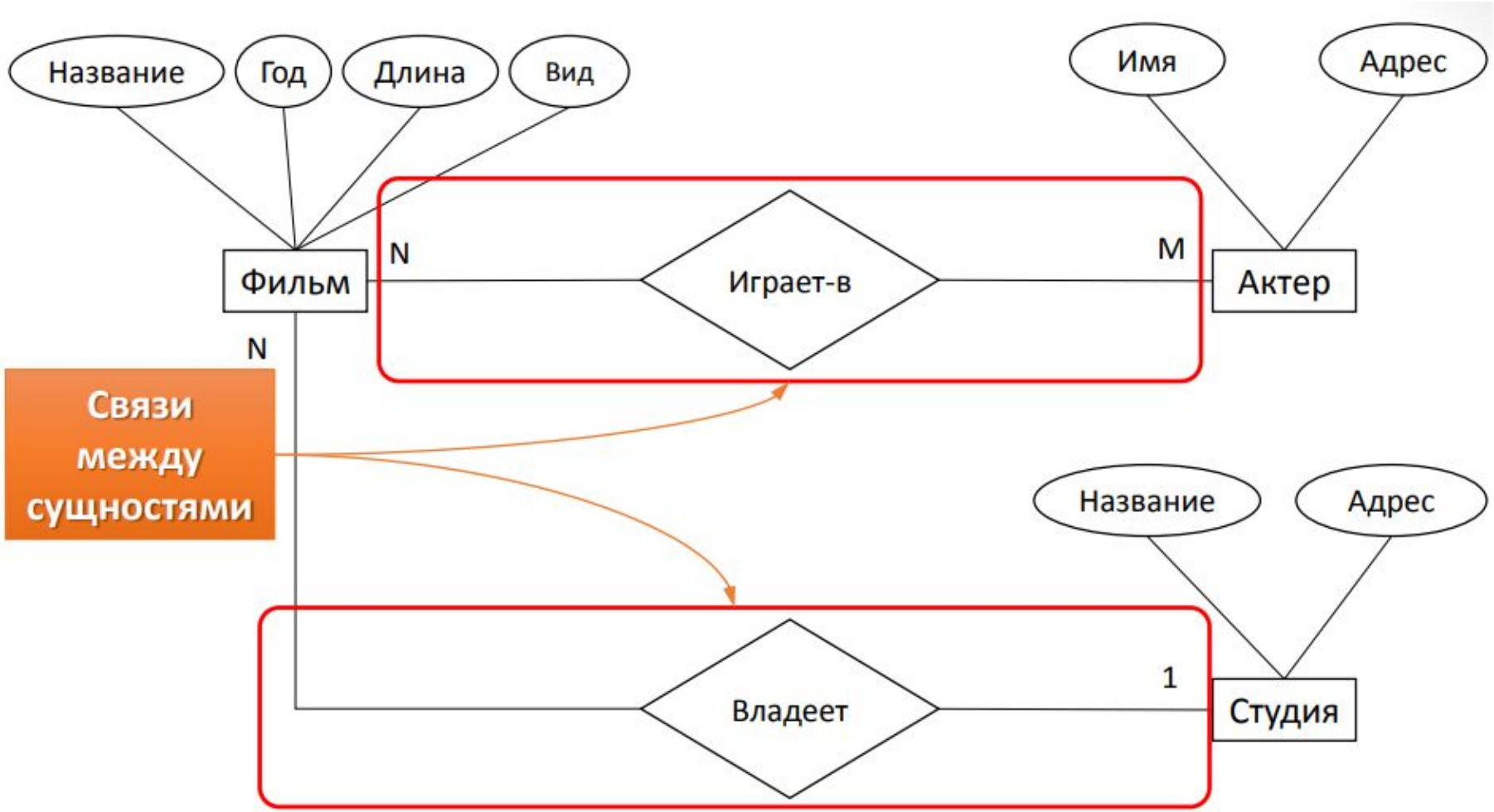
Обозначения: нотация Чена

Элемент диаграммы	Обозначает
	независимая сущность
	зависимая сущность
	родительская сущность в иерархической связи
	связь
	идентифицирующая связь
	атрибут
	первичный ключ
	внешний ключ (понятие внешнего ключа вводится в реляционной модели данных)
	многозначный атрибут
	получаемый (наследуемый) атрибут в иерархических связях

Сущности и их атрибуты: нотация Чена

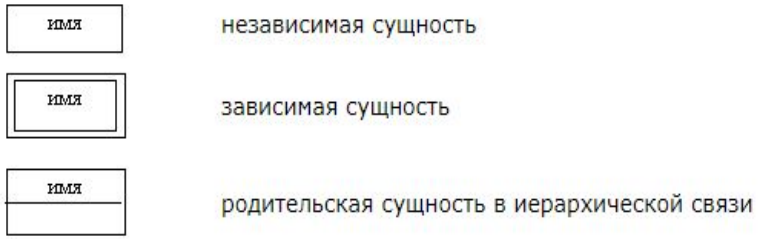


Связи между сущностями: нотация Чена



Обозначения: нотация Мартина

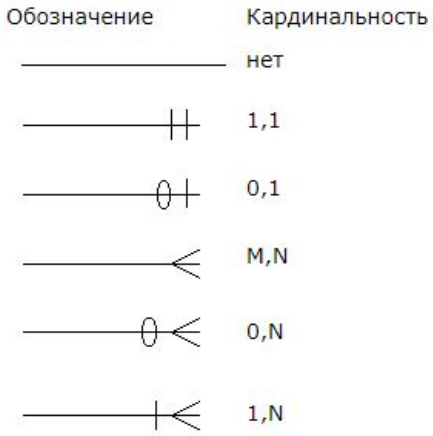
Обозначения сущностей:



Список атрибутов приводится внутри прямоугольника, обозначающего сущность.

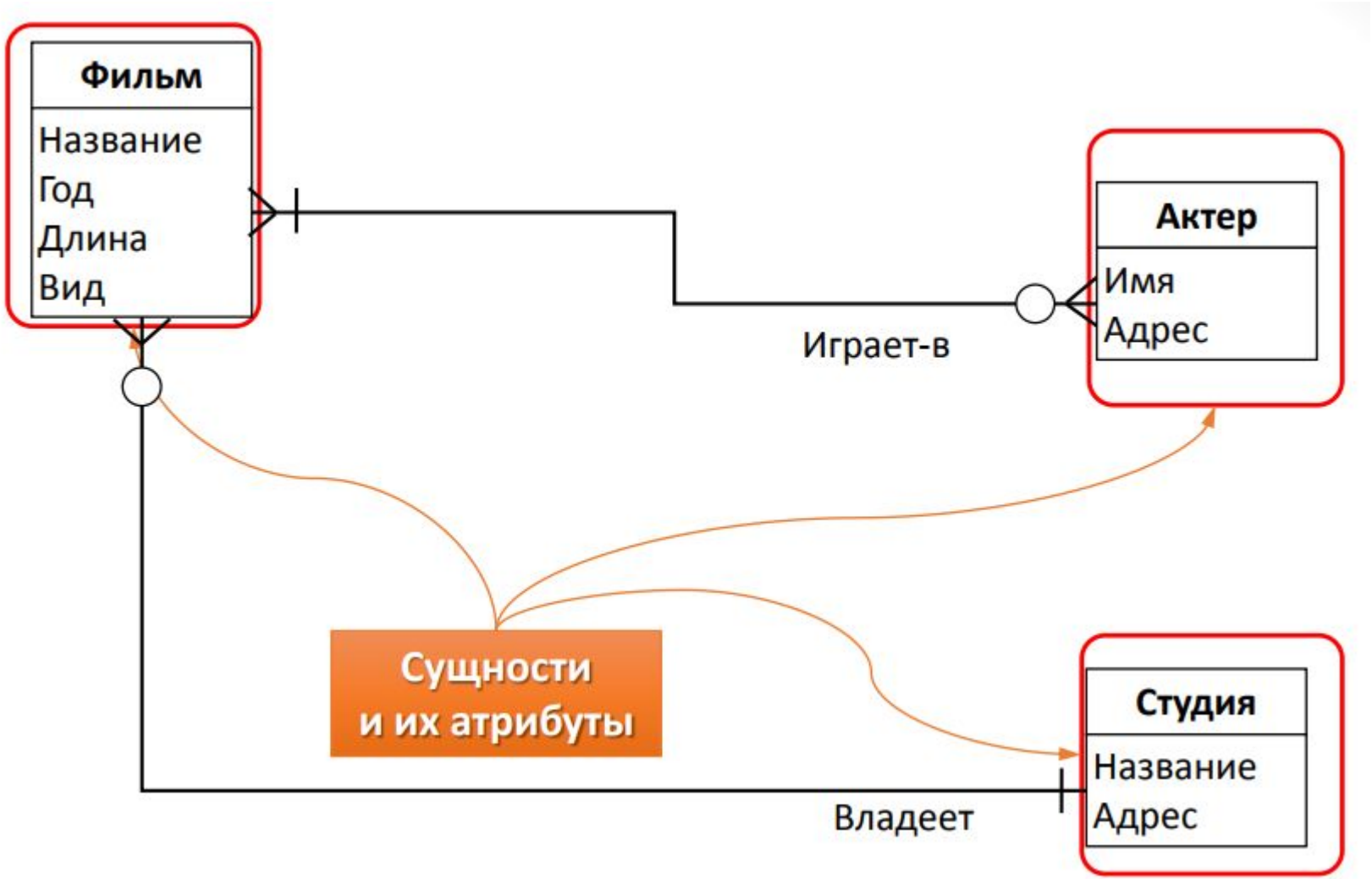
Ключевые атрибуты подчеркиваются.

- Связи изображаются линиями, соединяющими сущности, вид линии в месте соединения с сущностью определяет кардинальность связи:

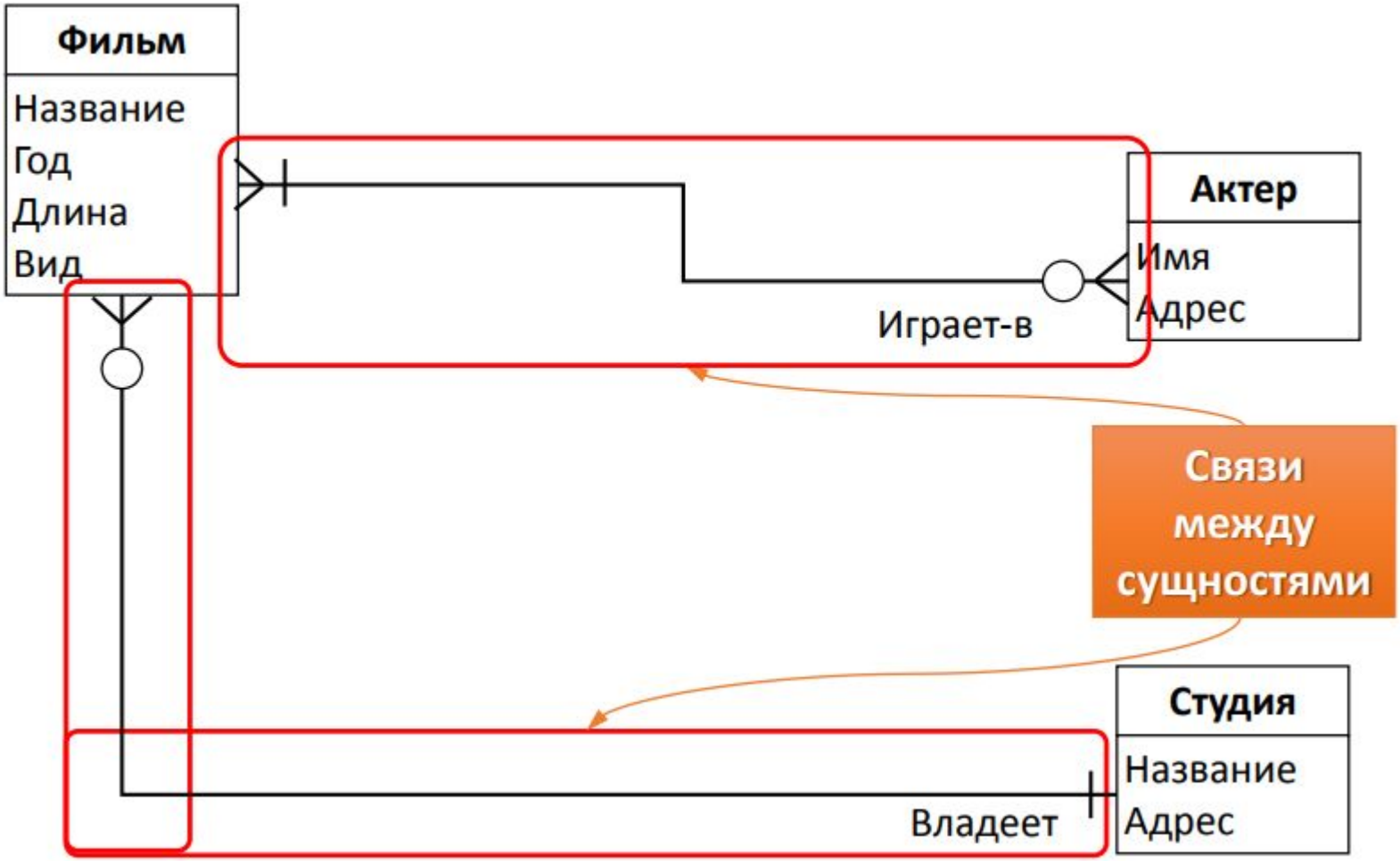


- Имя связи указывается на линии ее обозначающей

Сущности и их атрибуты: нотация Мартина



Связи между сущностями: нотация Мартина



Задание. Построить р**еляционную модель данных (методом Питера Чена)**

Сущности	*Связи*
• ***Шахматисты играют партии в рамках турниров, проводимых организаторами.*** • ***Шахматист – ФИО, пол, возраст.*** • ***Партия – игравший белыми, игравший черными, результат игры.*** • ***Турнир – название, сроки.*** • ***Организатор – название, адрес.***	• ***В турнире участвуют два или более шахматистов.*** • ***Шахматист может участвовать в нескольких турнирах.*** • ***У турнира может быть много организаторов.*** • ***Организатор может организовать много турниров.***

Тема 7. Нормализация баз данных

Понятие нормализации

- **Нормализация** - это формальный метод анализа *отношений* на основе их первичного ключа и существующих связей.
  - Задача нормализации - это замена одной схемы (или совокупности *отношений*) БД другой схемой, в которой *отношения* имеют более простую и регулярную структуру.

Назначение нормализации баз данных

Проектирование схемы БД должно учитывать:

- минимизацию дублирования данных
- упрощение и ускорение процедур обработки и обновления данных

Для решения подобных проблем проводится **нормализация отношений**

Аномалии

- **Аномалией** называется такая ситуация в таблице БД, которая приводит к противоречию в БД либо существенно усложняет обработку БД.

- *Причиной аномалии является излишнее дублирование данных в таблице.*

Денормализация?

- В технологии работы с хранилищами данных может использоваться обратный прием - **денормализация отношений** с целью увеличения скорости выполнения запросов к очень большим объемам архивных данных.

http://foreva.susu.ru/courses/db/lecture2.pdf