Введение в моделирование данных, базы данных и SQL

Лекции 1-2

Контактная информация

- **Анатолий Николаевич Бездушный <anb@ccas.ru>**
  - Антон Михайлович Меденников **<meden@ccas.ru>**
  - Кирилл Борисович Теймуразов **<kbt@ccas.ru>**
- адрес:
  - ВЦ РАН, м Ленинский проспект, Вавилова 42, ком. **164** тф. 135-54-71 (*4220)
- «сайтик» с материалами лекций
  - http://bdis.umeta.ru/db

Цели курса

- **Познакомить с понятиями**, возможностями и средствами
  - Реляционных баз данных (БД/DB - РБД/RDB)
  - Систем управления реляционными базами данных (СУБД/DBMS - РСУБД/RDBMS)
  - Моделирования данных
  - Проектирования БД
  - Языка структурированных запросов (SQL)
- Дать вам возможность научиться **проектировать и использовать** БД
- Научить осознавать смысл/назначение этапов **проектирования** БД
- Дать вам возможность освоить **правила работы** с БД средствами языка SQL (в среде MS SQL Server 2005)

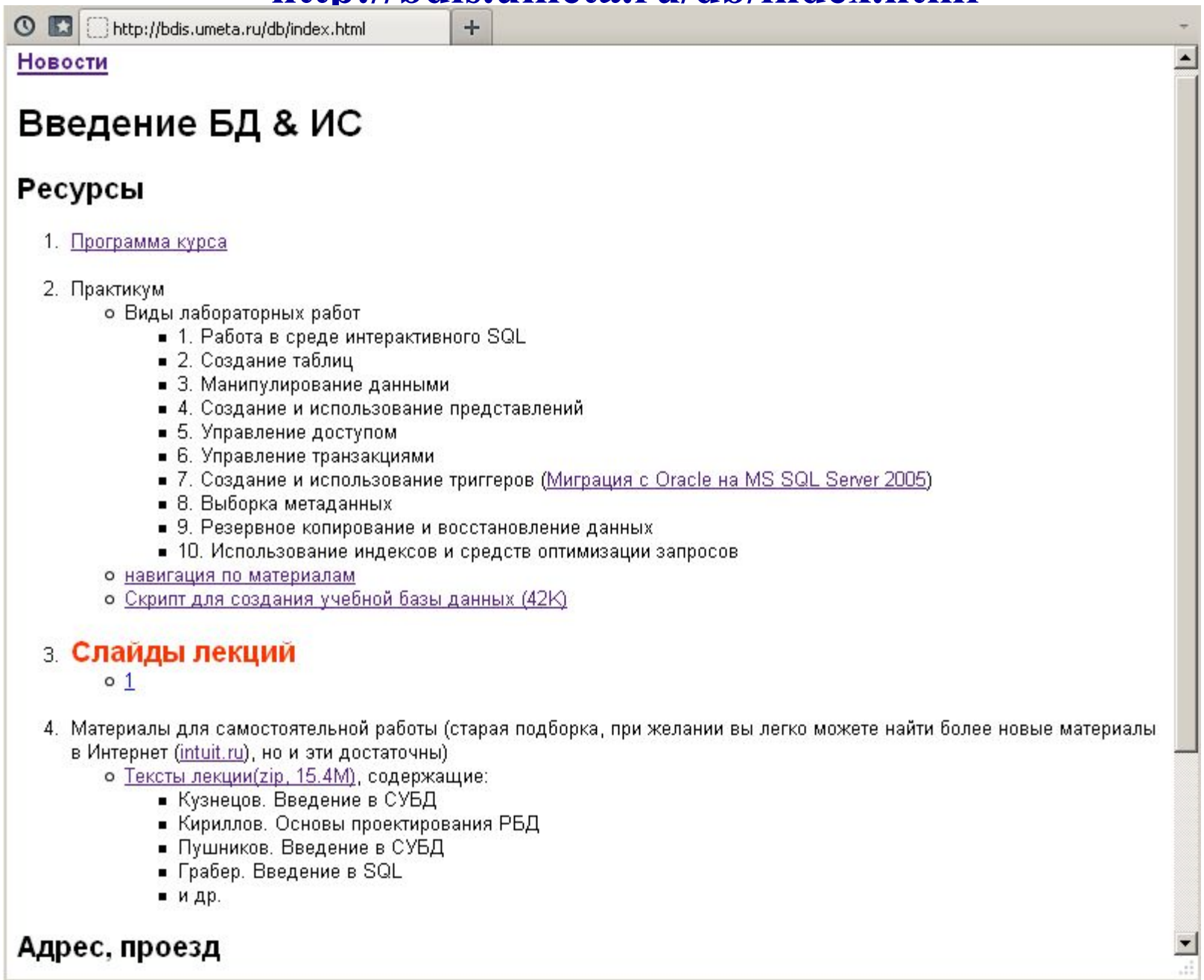
Структура курса

№ п/п	Тема	Объем (час.)	
		лекц.	лаб.
1.	*Основные концепции баз данных*	2	0
2.	*Модели данных* (обзор и сравнение)	1	0
3.	*Реляционная модель данных.*	2	2
4.	*Язык запросов Structured Query Language (SQL)*	10	18
5.	*Введение в проектирование реляционных баз данных*	6	2
6.	*Восстановление данных*	2	2
7.	*Совместное использование базы данных*	2	2
8.	*Безопасность данных.*	1	2
9.	*Целостность данных*	2	2
10.	*Обработка запросов*	2	2
11.	*Реляционная алгебра*	1	0
12.	*Реляционное исчисление*	1	0
	ИТОГО	32	32

Рекомендуемая литература

- **Гарсиа-Молина Г.,Ульман Д.,Уидом Д. Системы баз данных Полный курс**. // М.: Ид. «Вильямс», 2002. - 1088 с.
- **Дейт К. Введение в системы баз данных. // К.: "Диалектика",** 1998. 784 с.
- Грабер М. Введение в SQL. // М.:Изд-во “Лори”,1996
- Грабер М. Справочное руководство по SQL. - М.: Лори, 1997. - 291 с.
- **http://citforum.ru/database/edu.shtml** http://citforum.ru/database/ http://citforum.ru/database/sql.shtml
  - Кузнецов С.Д. **Базы данных. Вводный курс**
  - Кузнецов С.Д. Основы современных баз данных
  - Пушников А.Ю. Введение в системы управления базами данных
  - …
- **http://www.intuit.ru** (catalog/database/gentheory/ (catalog/database/gentheory/, /catalog/database/sql/ (catalog/database/gentheory/, /catalog/database/sql/, catalog/database/serversdb/)

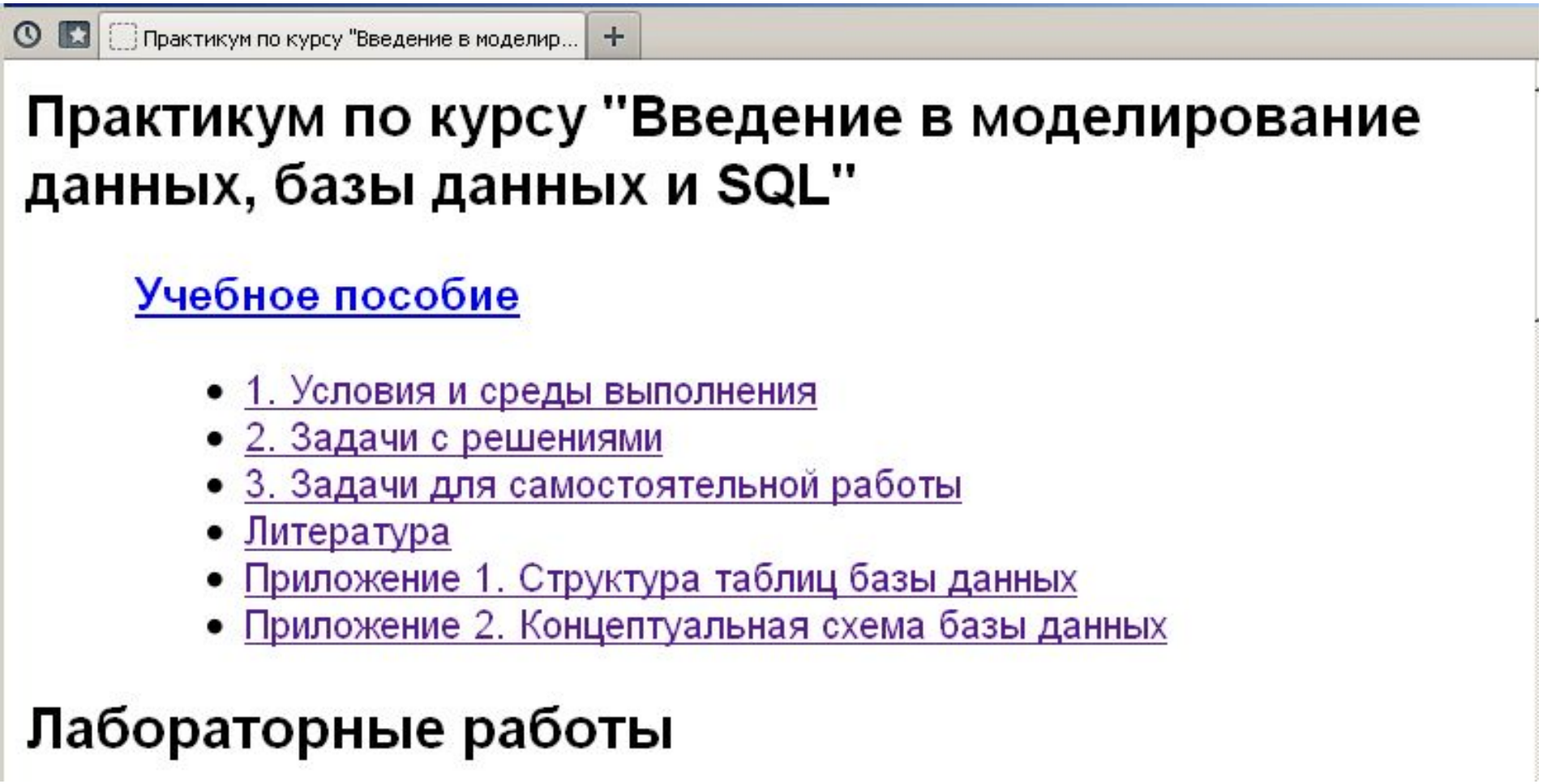
«Сайтик» с материалами
http://bdis.umeta.ru/db/index.html



Лабораторные работы

- целью - применение знаний теории для разработки собственной системы баз данных,
- познакомить с промышленными корпоративными СУБД,
- основное внимание **проектированию и работе с SQL**,
- выполнение **работ с демонстрационным заданием**,
- проектирование, построение и использование БД индивидуального задания,
- в ходе каждой лабораторной работы в начале на демонстрационном задании осваивается тема работы, затем решаются соответствующие задачи индивидуального задания,
- среда СУБД **MS SQL Server 2005**,
- имеется справочно - методическое пособие к практикуму.

Пособие - 1


Практикум по курсу "Введение в моделир...
Практикум по курсу "Введение в моделирование данных, базы данных и SQL"
Учебное пособие
1. Условия и среды выполнения
2. Задачи с решениями
3. Задачи для самостоятельной работы
Литература
Приложение 1. Структура таблиц базы данных
Приложение 2. Концептуальная схема базы данных
Лабораторные работы

Пособие-2

Практикум по курсу "Введение в моделир...

Лабораторные работы

Лабораторные работы

- 1. Работа в среде интерактивного SQL
- 2. Проектирование схемы
- 3. Создание таблиц
- 4. Манипулирование данными
- 5. Создание и использование представлений
- 6. Управление транзакциями
- 7. Управление доступом
- 8. Выборка метаданных
- 9. Создание и использование триггеров
- 10. Использование индексов и средств оптимизации запросов

Пример выполнения

- 1. Работа в среде интерактивного SQL
- 2. Проектирование и создание базы данных
- 3. Манипулирование данными
- 4. Создание и использование представлений
- 5. Управление доступом
- 6. Управление транзакциями
- 7. Создание и использование триггеров
- 8. Выборка метаданных

Пособие-3

Справочные данные

База данных "King Corporation"

- Концептуальная схема
- Структура таблиц

Интерактивные среды

Oracle v. 8

- Oracle SQL*Plus - среда интерактивного SQL
- Oracle Navigator - среда администрирования (Personal Edition)

MS SQL Server 2005

- SQL Server Management Studio: выполнение запросов
- SQL Server Management Studio: средства администрирования
- Средства оптимизации запросов, предоставляемые средой Microsoft SQL Server

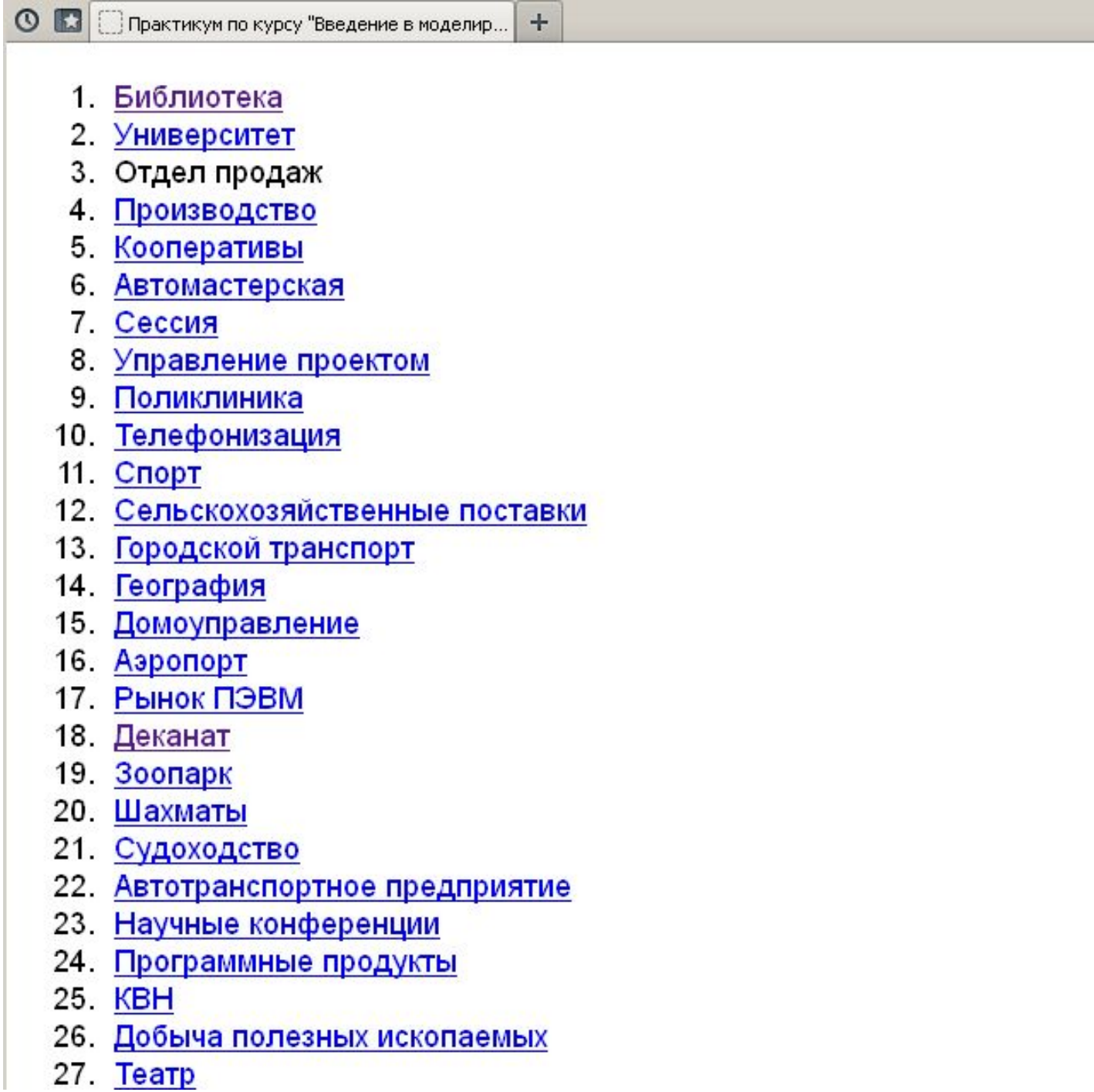
DB2 v. 7

- IBM DB2 Command Center - среда интерактивного SQL
- IBM DB2 Control Center - среда администрирования

SQL

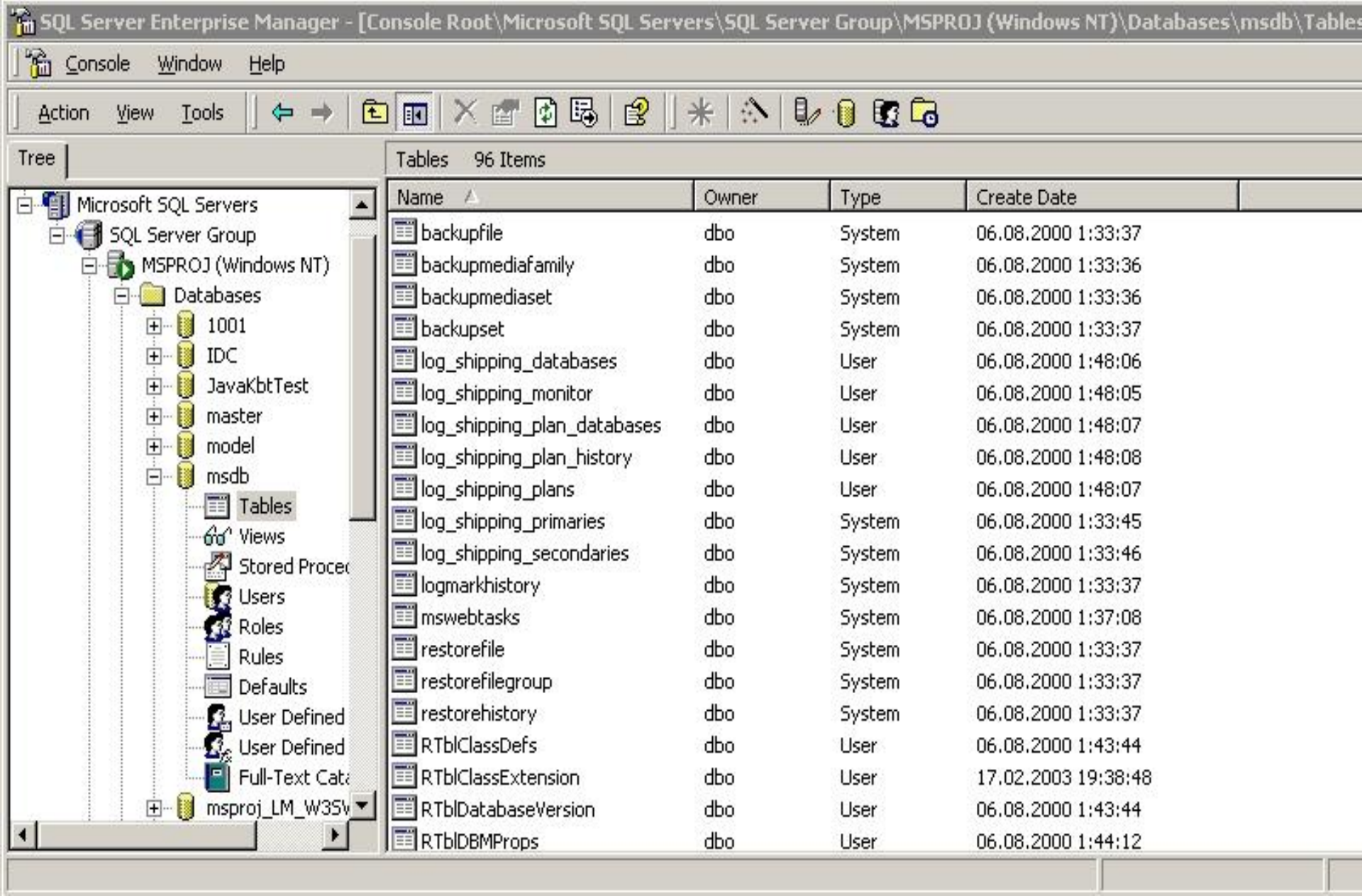
- Синтаксис запроса
- Дата и время
- Создание таблиц
- Операторы манипулирования данными
- Представления
- Управление доступом
- Управление транзакциями
- Триггеры
- Системный каталог и словарь данных
- Использование индексов

Пособие-4

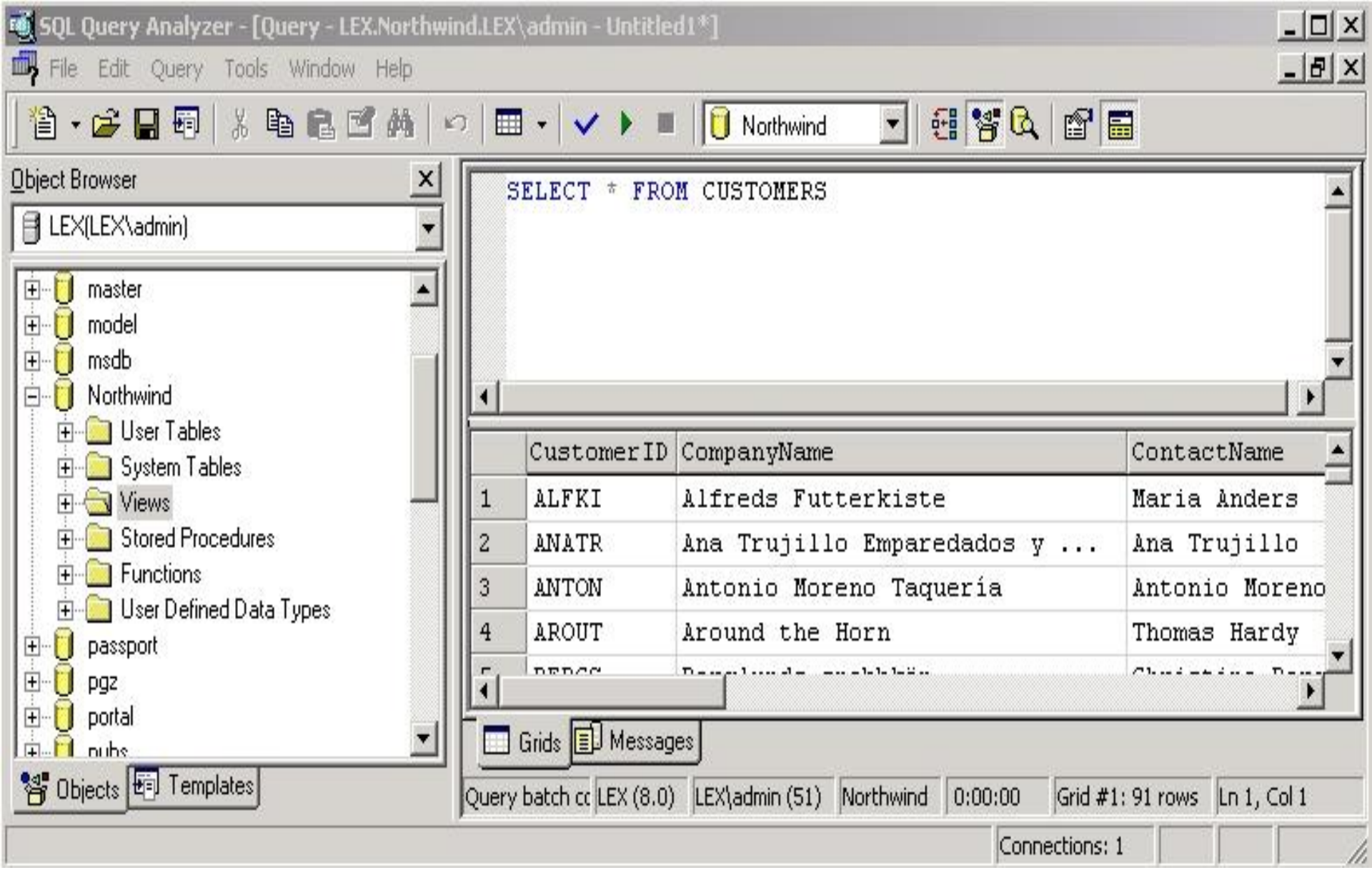


1. Библиотека
2. Университет
3. Отдел продаж
4. Производство
5. Кооперативы
6. Автомастерская
7. Сессия
8. Управление проектом
9. Поликлиника
10. Телефонизация
11. Спорт
12. Сельскохозяйственные поставки
13. Городской транспорт
14. География
15. Домоуправление
16. Аэропорт
17. Рынок ПЭВМ
18. Деканат
19. Зоопарк
20. Шахматы
21. Судоходство
22. Автотранспортное предприятие
23. Научные конференции
24. Программные продукты
25. КВН
26. Добыча полезных ископаемых
27. Театр

Microsoft Enterprise Manager



Microsoft Query Analyzer



Клиенты и серверы локальных сетей

- **Рабочая станция** - для работы пользователя,
  - потребности пользователя определяют ее ресурсы
- **Сервер** - предоставляет ресурсы (услуги) рабочим станциям и/или другим серверам
  - по его функциональному назначению
  - под потребности сети
  - должен иметь соответствующие ресурсы
- **Клиент локальной сети**
  - компонент сети, запрашивающий услуги у некоторого сервера
- **Сервер локальной сети**
  - компонент сети, оказывающий услуги некоторым клиентам

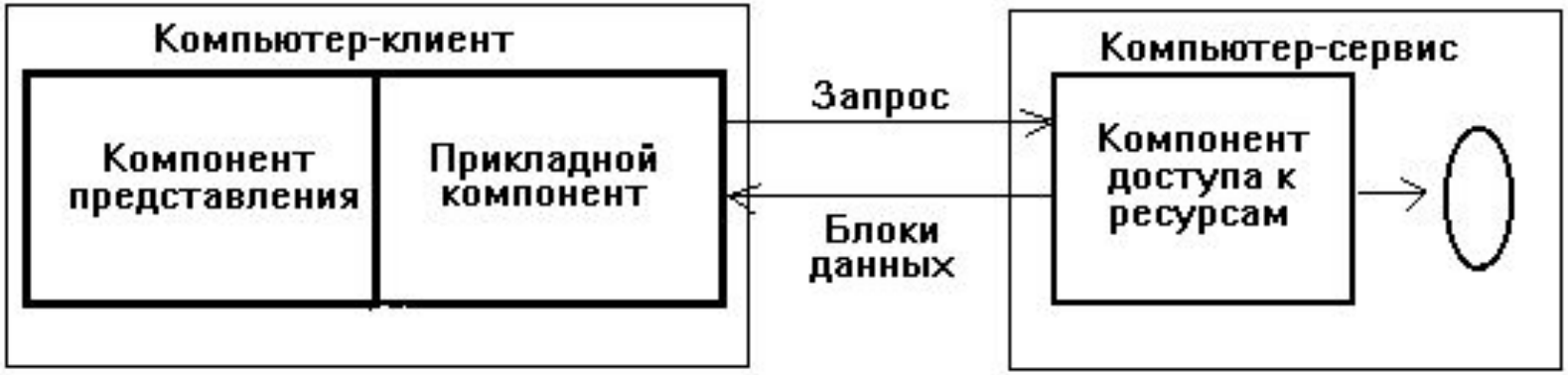
Архитектура "клиент-сервер"

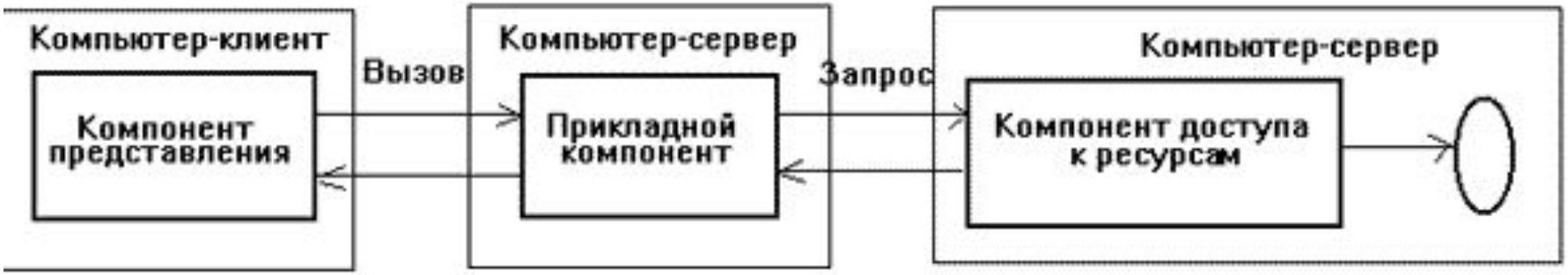
- Обеспечение **коллективного доступа** к ресурсам сети, для которого требуется
  - некоторый интерфейсный программный слой, поддерживающий взаимодействия клиента и сервера
- Система разбивается на **две части**
  - клиентскую и серверную части, которые могут выполняться в разных узлах сети
- Прикладная программа взаимодействуют с клиентской частью системы,
  - которая выступает для нее **как серверная часть,**
  - обеспечивает надсетевой интерфейс к серверной части – взаимодействует по сети с серверной частью

Примеры серверов

- Вычислительный сервер
  - производит вычисления, которые невозможно выполнить на рабочих станциях
- Файловый сервер
  - общее хранилище файлов для всех рабочих станций
- **Сервер баз данных**
  - СУБД, принимающая запросы и возвращающая результаты

Клиент-сервер





Разделение функций между клиентами и серверами

"Толстый клиент"

- клиентские станции должны иметь достаточную мощностью
- на стороне клиента - реализация всей или существенной части обработки данных (прикладной логики)
- сервер может только обеспечивать хранение данных

"Тонкий клиент"

- разница в мощностях клиента и сервера велика
- клиент только обеспечивает визуальный интерфейс системы
- на стороне сервера - хранение и вся обработка данных (необходимы соответствующие программные средства)

"Сервер приложений"

- **сервера БД** осуществляет хранение данных и обеспечивает их целостность
- **сервера приложений** (application serever) обеспечивает обработку данных системы
- **клиент** представляет визуальный интерфейс системы

БД и СУБД

Базовая концепция ИТ

- данные должны быть организованы в **базы данных** (БД) с целью (*тоже в файлах*)
  - адекватного отображения изменяющегося реального мира,
  - удовлетворения информационных потребностей пользователей.
- БД создаются и функционируют под управлением специальных программных комплексов
  - **системами управления базами данных** (СУБД)

Что есть база данных (БД)?

- структурированная коллекция **логически согласованных** данных
  - служащая для специфических целей
  - предназначенная для групп пользователей
- совокупность **связанных** данных,
  - организованных по определенным правилам (**модель данных**),
    - предусматривающим общие принципы описания, хранения и манипулирования,
  - **независимая** от прикладных программ

Характеристики БД

- **структурированные** данные (*в отличие от файлов*)
  - типы данных & «поведение» данных
- постоянное существование (**хранимость**)
  - данные хранятся во внешней памяти
- **манипулирование** данными
  - декларативные языки запросов
  - процедурные языки программирования БД
- **согласованность, корректность** данных
- исполнение операций
  - быстрое извлечение и сохранение данных
- **разделение** данных
  - одновременный доступ
- **достоверность**

Что есть СУБД?

- коллекция **программных средств** для создания и управления базами данных, обеспечивающих
- **определение**
  - хранимых типов данных (моделей данных предметной области)
- **построение**
  - сохранение, наполнение (поддержка постоянного существования данных)
- **манипулирование**
  - извлечение, модификацию, формирование производных данных (средства формулировки и выполнения разнообразных запросов отчетов)
- **система баз данных** – СУБД + одна или несколько БД, управляемых это СУБД.

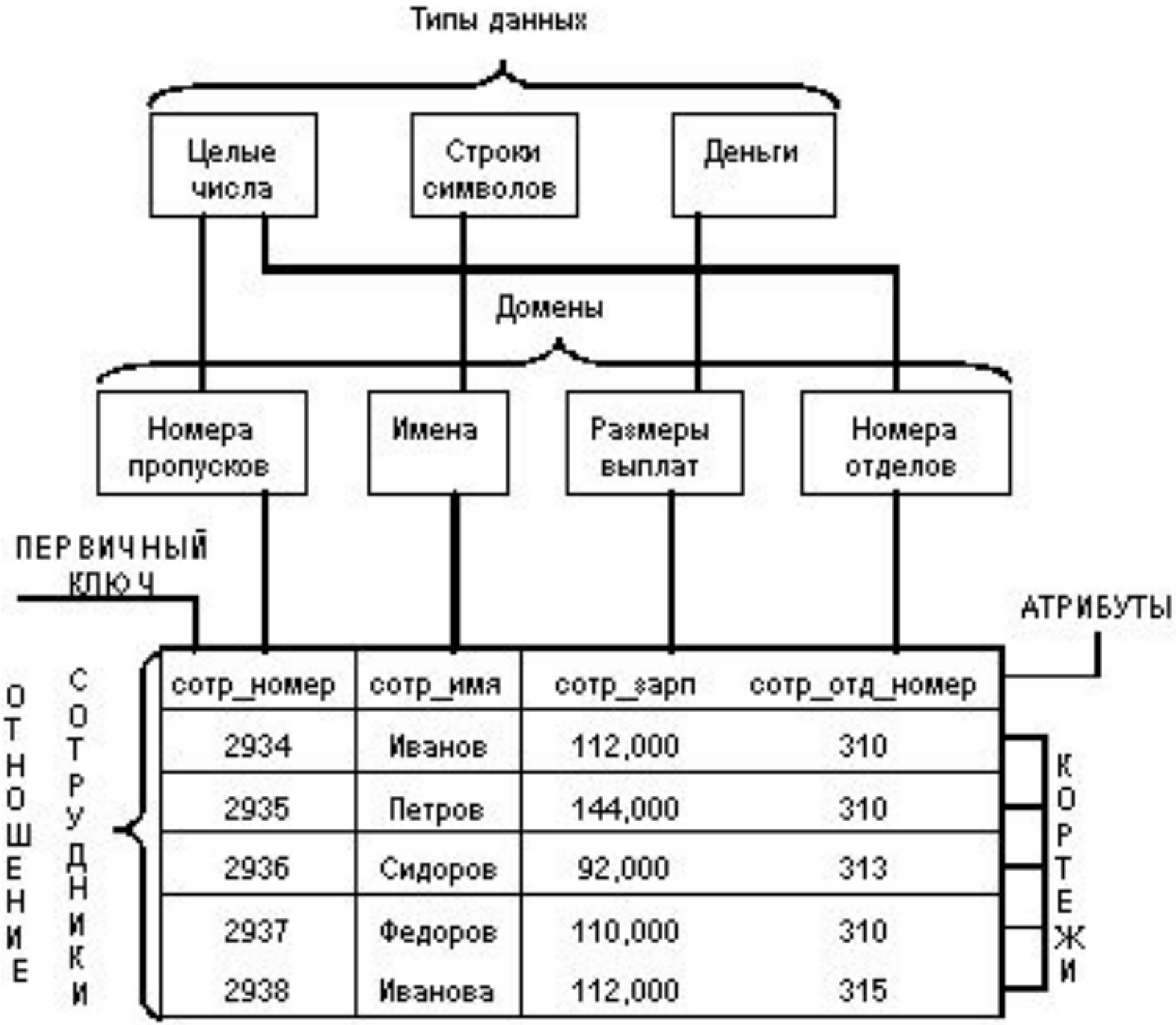
Ключевые требования к СУБД

- согласованное **хранение** данных
  - поддержка взаимосвязи данных
- обеспечение **надежности** хранения
  - быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя
- удобный, выразительный способ **выполнения запросов** к данным
- **параллельная** работа с базой данных

Реляционные СУБД (РСУБД)

- Распространение **реляционных** (**табличных**) СУБД обусловлено:
  - увеличением объема хранимых данных,
  - их структурной сложностью,
  - расширением круга пользователей ИС,
  - сравнительно простым для понимания набором понятий
    - РБД - совокупность **таблиц**
  - математическим обоснованием (**реляционная модель данных - РМД**)

Основные понятия и термины



Пример-1

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Black	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Rome
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

SP

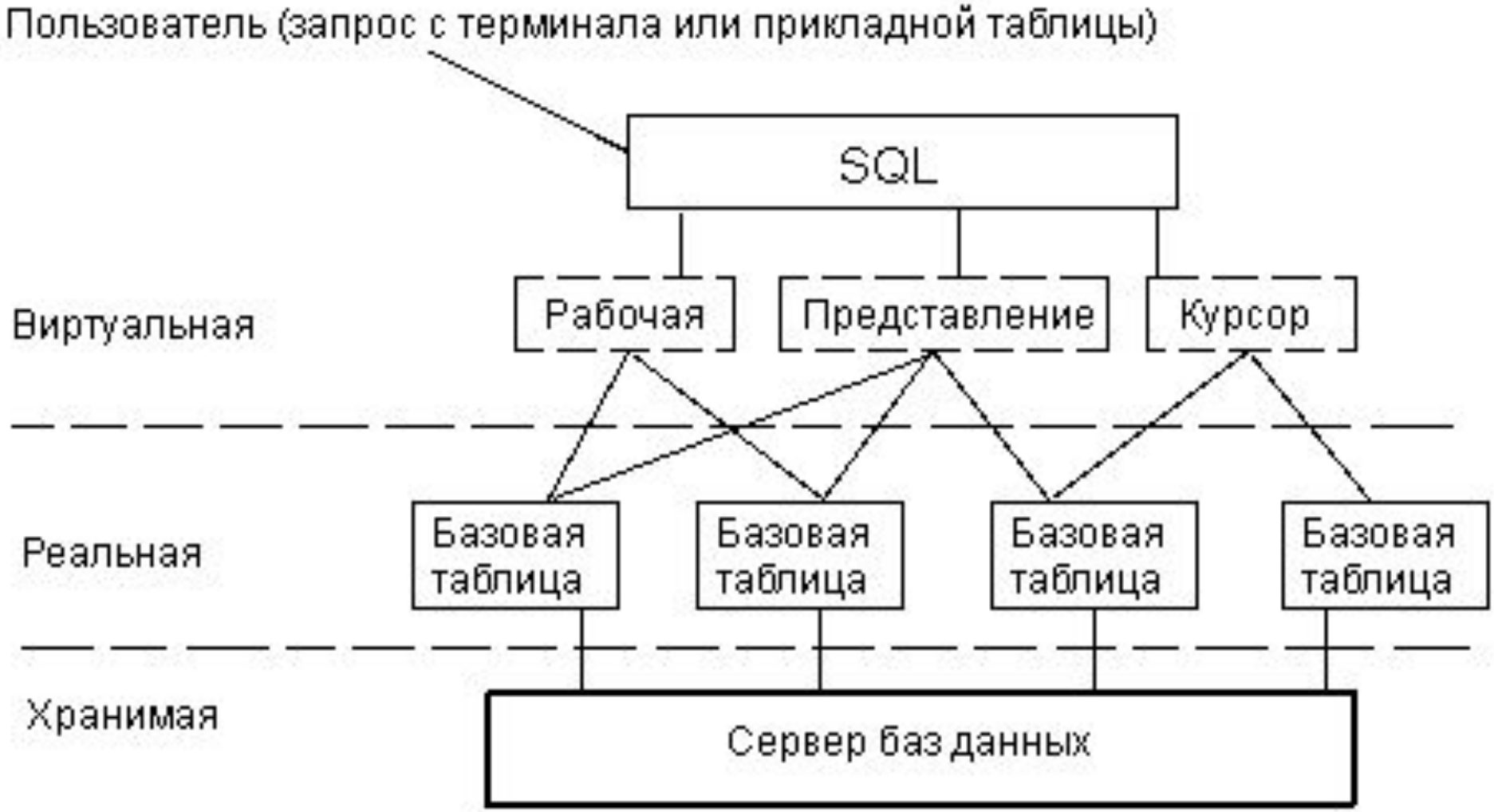
S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Пример-2

№ пп	Имя столбца	Тип данных DB2(MSSQL)	Комментарий
Таблица EMPLOYEE - сотрудники фирмы			
1	employee_id	SMALLINT	Код сотрудника
2	last_name	VARCHAR(15)	Фамилия
3	first_name	VARCHAR(15)	Имя
4	middle_initial	VARCHAR(1)	Средний инициал
5	manager_id	SMALLINT	Код начальника
6	job_id	SMALLINT	Код должности
7	hire_date	DATE(DATETIME)	Дата поступления в фирму
8	salary	NUMERIC(7,2)	Зарплата
9	commission	NUMERIC(7,2)	Комиссионные
10	department_id	SMALLINT	Код отдела

```
CREATE TABLE EMPLOYEE (
  employee_id SMALLINT NOT NULL PRIMARY KEY,
  last_name VARCHAR(15),
  first_name VARCHAR(15),
  middle_initial VARCHAR(1),
  manager_id SMALLINT,
  job_id SMALLINT,
  hire_date DATETIME,
  salary NUMERIC(7,2),
  commission NUMERIC(7,2),
  department_id SMALLINT  );
```

БД


Пользователь (запрос с терминала или прикладной таблицы)
SQL
Виртуальная
Рабочая
Представление
Курсор
Реальная
Базовая таблица
Базовая таблица
Базовая таблица
Базовая таблица
Хранимая
Сервер баз данных

Общие понятия реляционного подхода - 1

- **тип данных**
  - элементарные/простые
- **домен**
  - базовый тип данных и логическое выражение, применяемое к элементам типа данных.
    - множества допустимых значений
- **схема отношения** (*описание таблицы*)
  - именованное множество пар
    - {**имя атрибута, имя домена**}
  - называют *заголовком отношения*
  - атрибуты именуют столбцы таблицы
    - *«столбец таблицы» - «атрибут отношения»"*
- **схема БД**
  - набор именованных схем отношений (*описаний таблиц*)

Общие понятия реляционного подхода - 2

- **кортеж** (*строка, запись таблицы*)
  - множество пар {имя атрибута, значение}
    - одно вхождение каждого имени атрибута, принадлежащего схеме отношения.
    - "значение" является допустимым значением домена атрибута
- **отношение** (*таблица*)
  - это множество кортежей
    - соответствующих одной схеме отношения
  - называют телом отношения
  - *результат запроса – отношение*

Общие понятия реляционного подхода - 3

- **суперключ**, уникальный идентификатор
  - набор атрибутов
    - уникально идентифицируют кортежи отношений
    - по значениям которых можно однозначно найти требуемый кортеж отношений
- **потенциальный** ключ
  - минимальный ключ
    - подмножество его атрибутов не является ключом
  - может быть несколько
- **первичный**
  - потенциальный ключ, выбранный основным ключом отношения
  - один в отношении

Общие понятия реляционного подхода - 4

- Внешний ключ
  - множество атрибутов отношения, которые точно соответствуют первичному ключу другого отношения
  - имена атрибутов могут быть другими
  - области значений (домены) атрибутов должны быть такими же
    - ***внешний*** ключ в отношении B и соответствующий ему **первичный** ключ в отношении A обычно представляют отношение вида «один-ко-многим» между A и B.

      `Student(`**`studNo`**`,name)`
      `Course (`**`courseNo`**`, subject, equipment)`
      `Enrol(`***`studNo,courseNo`***`,labmark)`

Фундаментальные свойства РБД

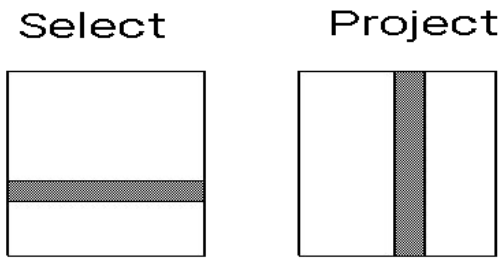
- отсутствие кортежей-дубликатов
- отсутствие упорядоченности кортежей
- отсутствие упорядоченности атрибутов
- атомарность значений атрибутов
- два базовых требования целостности РМД
  - целостность сущностей
    - любой кортеж любого отношения отличим от любого другого кортежа этого отношения
    - любое отношение должно обладать первичным ключом
  - целостности по ссылкам
    - для каждого значения внешнего ключа ссылающегося отношения, в отношении, на которое ведет ссылка, должен
      - найтись кортеж с таким же значением первичного ключа

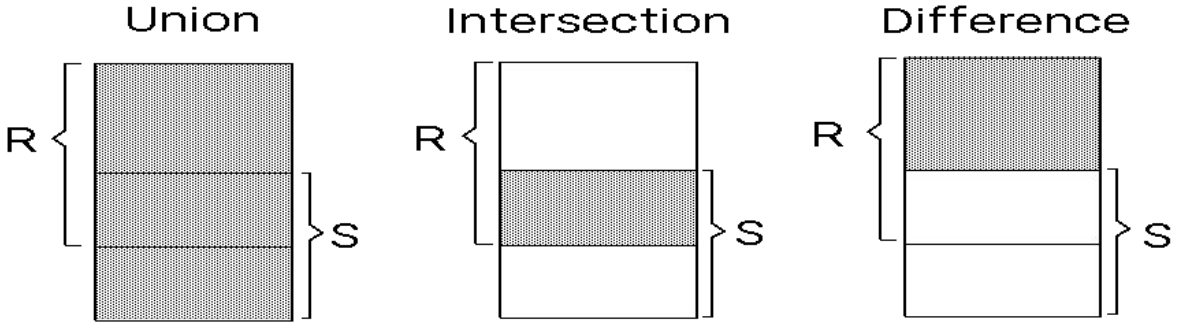
      либо
      - значение внешнего ключа должно быть неопределенным (NULL)

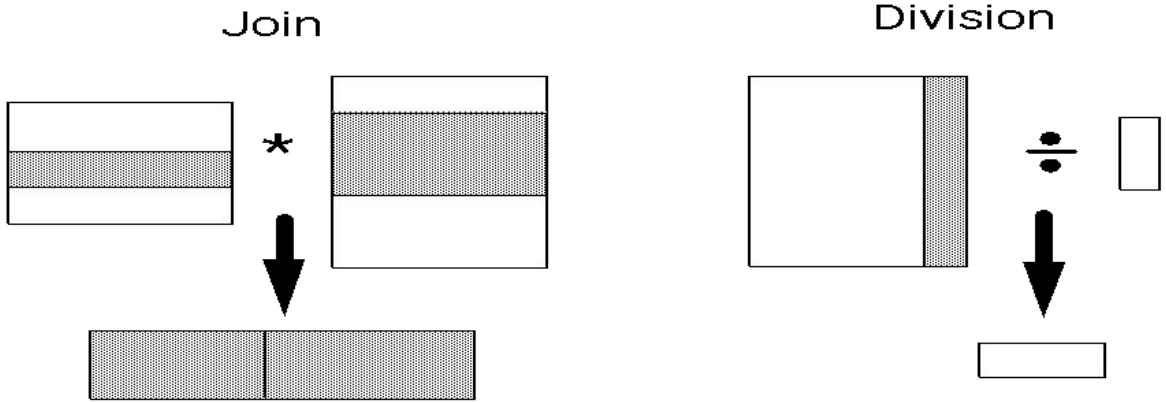
Сравнимые термины

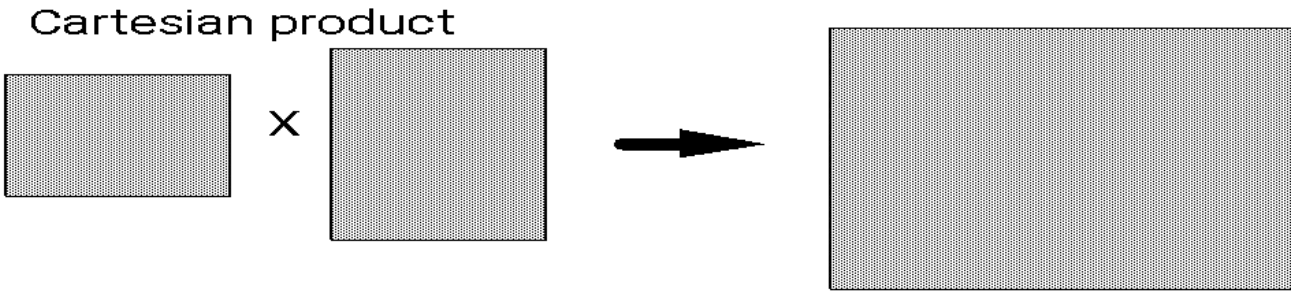
РМД	**РСУБД**
Схема отношения *заголовок отношения*	Описание таблицы
Отношение *тело отношения*	Таблица
Кортеж	Строка *или запись*
Атрибут	Столбец *или поле*
Домен	Множество значений *или домен*

Реляционные операции









SQL SELECT

Select-From-Where

- Основная форма запроса имеет вид:

SELECT (*ВЫБРАТЬ*) требуемые атрибуты
FROM (*ИЗ*) одной или более таблиц
WHERE (*ГДЕ*) условие на записи таблицы

БД примеров

- Большинство SQL запросов будет использовать БД со следующими заголовками отношений.

  Beers(name, manf)

  Bars(name, addr, license)

  Drinkers(name, addr, phone)

  Likes(drinker, beer)

  Sells(bar, beer, price)

  Frequents(drinker, bar)

  – Подчеркнуты атрибуты первичного ключа

Запросы с одной таблицей

Пример

- Используя Beers(name, manf), узнаем **какие виды пива производит «Guinness Brewing Worldwide»**?

```
SELECT name
FROM Beers
WHERE manf =
      'Guinness Brewing Worldwide';
```

Результат запроса

name
‘Guinness Extra Stout’
‘Guinness Foreign Extra Stout’
‘Harp’
‘Kilkenny’

Ответ - отношение с одним атрибутом name и записи с именами пива производимого «Guinness Brewing Worldwide», такими как «Guinness».

Соглашения (регулярные выражения)

- [**A**] – необязательно есть **A**, может отсутствовать
  A или `отсутствует`
- (**A**), (**A**)+ – непустая последовательность **A**
  A или **A A**
  A A A или **A A A A …**
- **A …**, (**A**)* ,
  {**A**} – возможно пустая последовательность **A**
  A или *`отсутствует`*
  A или **A A**
  A A A или **A A A A …**
- **A | B | C | D** – одно из **A**, **B**, **C**, **D**
  A или **B**
  C или **D**

Структура Select - a

```
SELECT[all|distinct]
   {*|{table.*|expr[alias]|view.*}
   [,{table.*|expr[alias]}]...}
FROM table [alias][,table[alias]] ...
[WHERE condition]
[GROUP BY expr [,expr] ...]
[HAVING condition]
[{UNION|UNION ALL|INTERSECT|MINUS}
   SELECT ...]
[ORDER BY {expr|position}
[ASC|DESC][,expr|position}[ASC|DESC].
```

SELECT

SELECT [[ALL] | DISTINCT]

{ * | элемент_SELECT **[,**элемент_SELECT**]**
...}

FROM {базовая_таблица **| представление}**
[псевдоним**] [,{**базовая_таблица **|**
представление**} [**псевдоним**]] ...**

WHERE
[NOT] WHERE_условие
[[AND|OR][NOT] WHERE_условие**]...**

[GROUP BY
фраза **[HAVING** фраза**]];**

Элемент_SELECT

[таблица.]* | значение | SQL_функция | системная_переменная

Значение – это

[таблица.]столбец | (выражение) | константа | переменная

Выражение– это

{[[+] | -]
{значение | функция_СУБД} [+ | - | * | **]}...

Фраза WHERE

WHERE_условие:

- значение { **=** | **<>** | **<** | **<=** | **>** | **>=** }
  { значение | **(** подзапрос **)** }
- значение_1 [**NOT**] **BETWEEN**
  значение_2 **AND** значение_3
- значение [**NOT**] **IN**
  { (константа [**,**константа]...) | **(** подзапрос **)** }
- [таблица**.**]столбец [**NOT**] **LIKE** '*строка_символов*' [**ESCAPE** *'символ*']
- значение **IS** [**NOT**] **NULL**
- **EXISTS (** подзапрос **)**

Смысл запроса с одной таблицей

- **Используя** отношения/таблицы, указанные в FROM части
- Осуществить **отбор** записей, следуя указаниям WHERE части
- Применить операцию **проекции** в соответствии с требованиями SELECT части
  - выбрать данные указанных столбцов,
  - выполнить необходимое их преобразование

Операционная семантика

- **Последовательно перебираем** записи таблицы, указанной в FROM части
- Проверяем, **удовлетворяет ли** текущая запись условиям WHERE части
- Если да, в соответствии с требованиями SELECT части **выбираем значения** атрибутов, вычисляем выражения, используя элементы текущей записи

* в SELECT части

- Если в FROM части указано одно отношение, то символ
  * в SELECT части означает - “все атрибуты этого отношения”
- Пример для отношения Beers(name, manf):

```
SELECT *
FROM Beers
WHERE manf =
      'Guinness Brewing Worldwide';
```

Результат запроса

name	manf
‘Guinness Extra Stout’	‘Guinness Brewing Worldwide’
‘Guinness Foreign Extra Stout’	‘Guinness Brewing Worldwide’
‘Harp’	‘Guinness Brewing Worldwide’
‘Kilkenny’	‘Guinness Brewing Worldwide’

Результат содержит все атрибуты Beers записей пива производимого «Guinness Brewing Worldwide».

Еще пример

Company(sticker, name, country, stockPrice)

Все компании UK, имеющие акции дороже > 50:

```
SELECT   *
FROM     Company
WHERE    country="UK" AND stockPrice > 50
```

Заголовок результирующего отношения:
R(sticker, name, country, stockPrice)

Переименование атрибутов

- Если необходимо, чтобы столбец результата имел другое имя, то используя

  AS <новое имя> ,

  можно переименовать столбец.
- Пример для Beers(name, manf):

```
SELECT name AS beer, manf
FROM Beers
WHERE manf =
    'Guinness Brewing Worldwide'
```

Результат запроса

beer	manf
‘Guinness Extra Stout’	‘Guinness Brewing Worldwide’
‘Guinness Foreign Extra Stout’	‘Guinness Brewing Worldwide’
‘Harp’	‘Guinness Brewing Worldwide’
‘Kilkenny’	‘Guinness Brewing Worldwide’

Выражения в SELECT части

- Любое имеющее смысл выражение можно указать как элемент **SELECT** части (атрибут результирующего отношения, столбец таблицы)
- Пример для Sells(bar, beer, price):

```
SELECT bar, beer,
    price * 28.5 AS priceInRub
FROM Sells;
```

Результат запроса

bar	beer	**priceInRub**
Удача	Kilkenny	180
Встреча	Miller	70
…	…	…

Константные выражения

- Используя Likes(drinker, beer):

```
SELECT drinker,
 'любит Харп' AS whoLikesHarp
FROM Likes
WHERE beer = 'Harp';
```

Результат запроса

drinker	**whoLikesHarp**
‘Алекс’	‘любит Харп’
‘Тим’	‘любит Харп’
…	…

Сложные выражения в WHERE части

- В Sells(bar, beer, price) найти **стоимость пива «Харп» в баре «Встреча»** :

```
SELECT price
FROM Sells
WHERE bar = 'Встреча' AND
      beer = 'Harp';
```

Элементы условий

Что можно использовать в WHERE части:

- имена атрибутов отношений
- операторы сравнения: =, <>, <, >, <=, >=
- арифметические операторы: stockprice*2
- строковые операторы (“||” конкатенация).
- лексикографический порядок при сравнении строк
- дополнение строк пробелами до равной длины
- сопоставление с шаблоном: s LIKE p
- операторы/функции над датами, временными значениями

Важные моменты

- Две одиночных кавычки в строке представляют саму кавычку (апостроф - ').
- Условия в WHERE части могут использовать операции AND, OR, NOT, и скобки в соответствии с обычным способом построения логических выражений.
- SQL не чувствителен к регистру букв. Прописные и строчные буквы суть одно и тоже, пока не заключены в двойные кавычки (").

Шаблоны

- WHERE часть может включать условия, в которых строки сопоставляются с шаблонами, чтобы обнаружить соответствия.
  - <Attribute> **LIKE** <pattern> или <Attribute> **NOT LIKE** <pattern>

- Шаблон это строка в кавычках с метасимволами
  - **%** = “любая цепочка символов”
  - **_** = “любой символ”

Пример

- В Drinkers(name, addr, phone) найти с любителей пива с тф. номером от «МТС» или «Мегафон» - 916 или 926 :

```
SELECT name
FROM Drinkers
WHERE phone LIKE '89_6%';
```

```
P -> 89D6S
D -> цифра |
S -> цифра S |
```

Пример

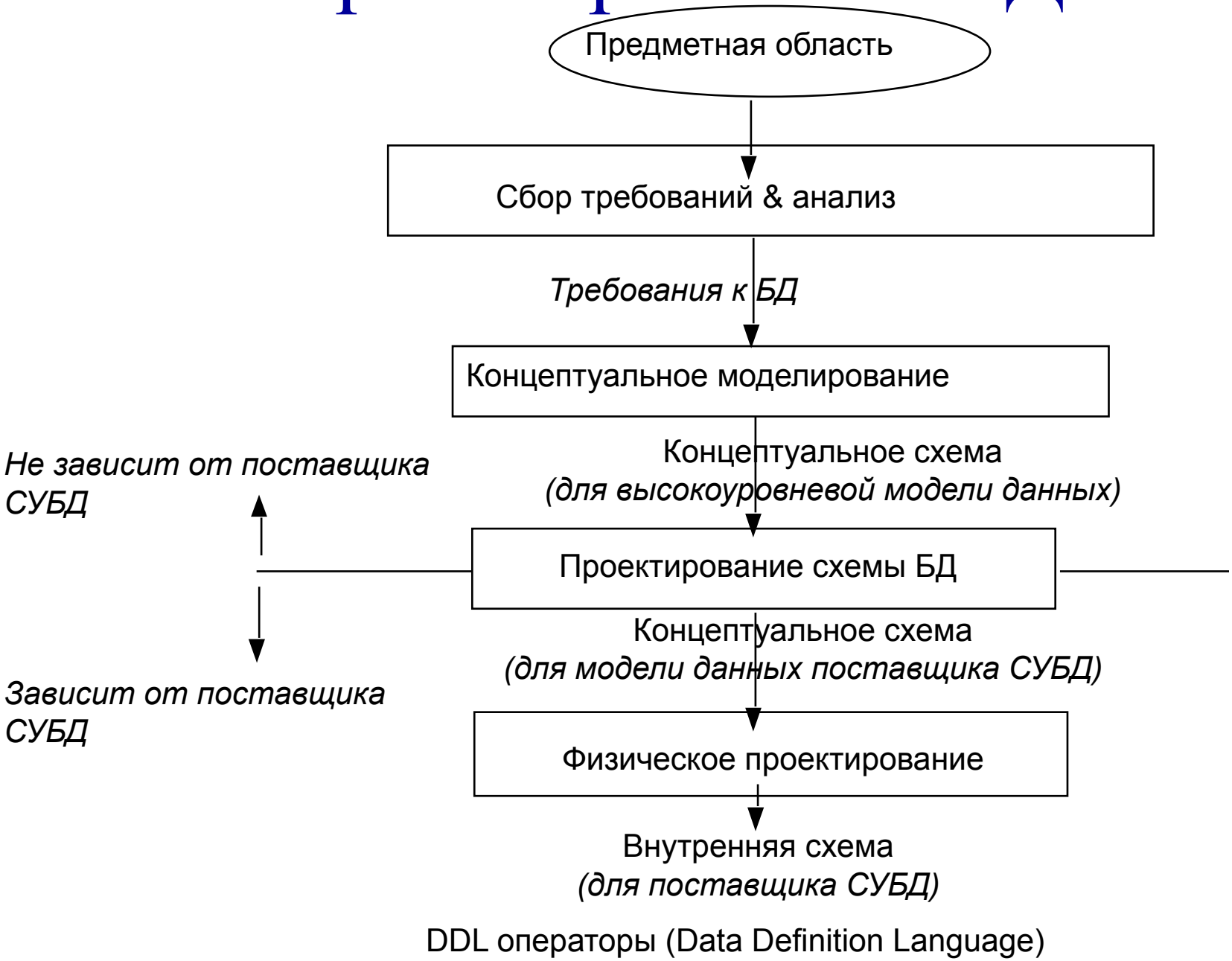
Company(sticker, name, address, country, stockPrice)

Найти все компании из UK, чей адрес содержит ‘London’:

```
SELECT   *
FROM     Company
WHERE    country=‘UK’ AND
          address LIKE ‘% London %’
```

Обзор проектирования и создания БД (лаб.2)

Проектирование РБД



Этапы построения приложения БД-1

- Этап 1: **анализ** предметной области приложения
  - Обсуждаем с заказчиком, коллегами, что подлежит моделированию в предметной области, какие требования выдвигаются к ней, к приложению

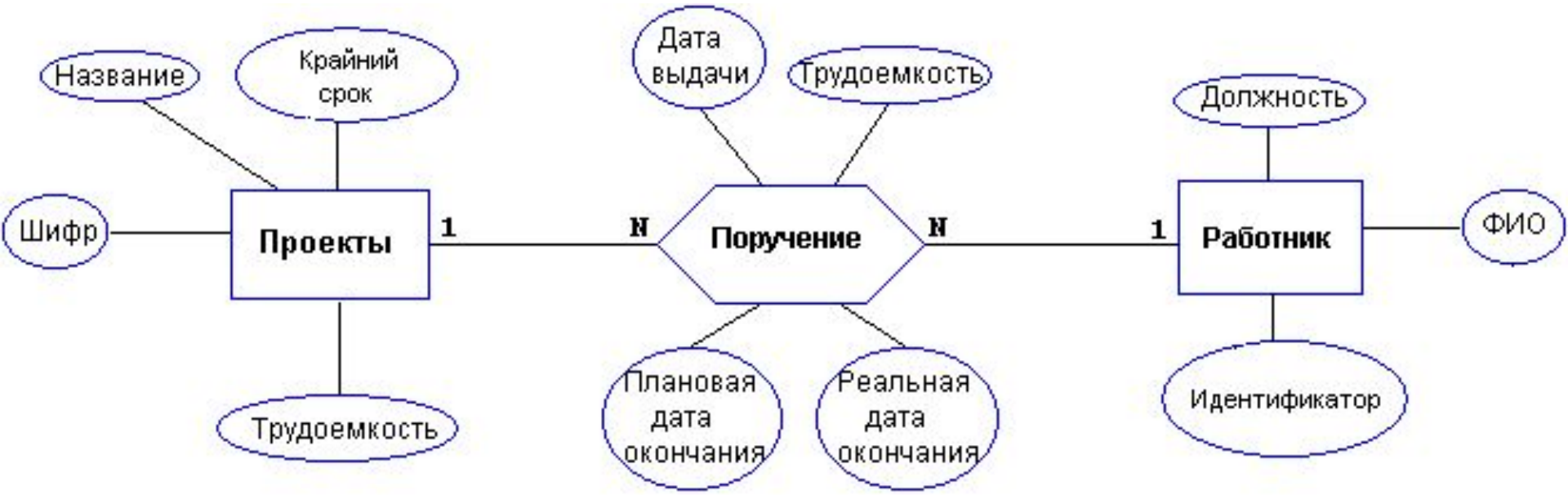
Требования предметной области

- предустановленный набор должностей
- перечень должностей отдела определяется штатным расписанием
  - строка штатного расписания содержит информацию о количестве ставок одной должности в одном отделе
- зачисление сотрудника может производиться только в соответствии со штатным расписанием.
- один сотрудник может быть зачислен на несколько работ
- у сотрудника есть руководитель
- образование характеризуется профилем образования и уровнем
- сотрудник может иметь несколько образований

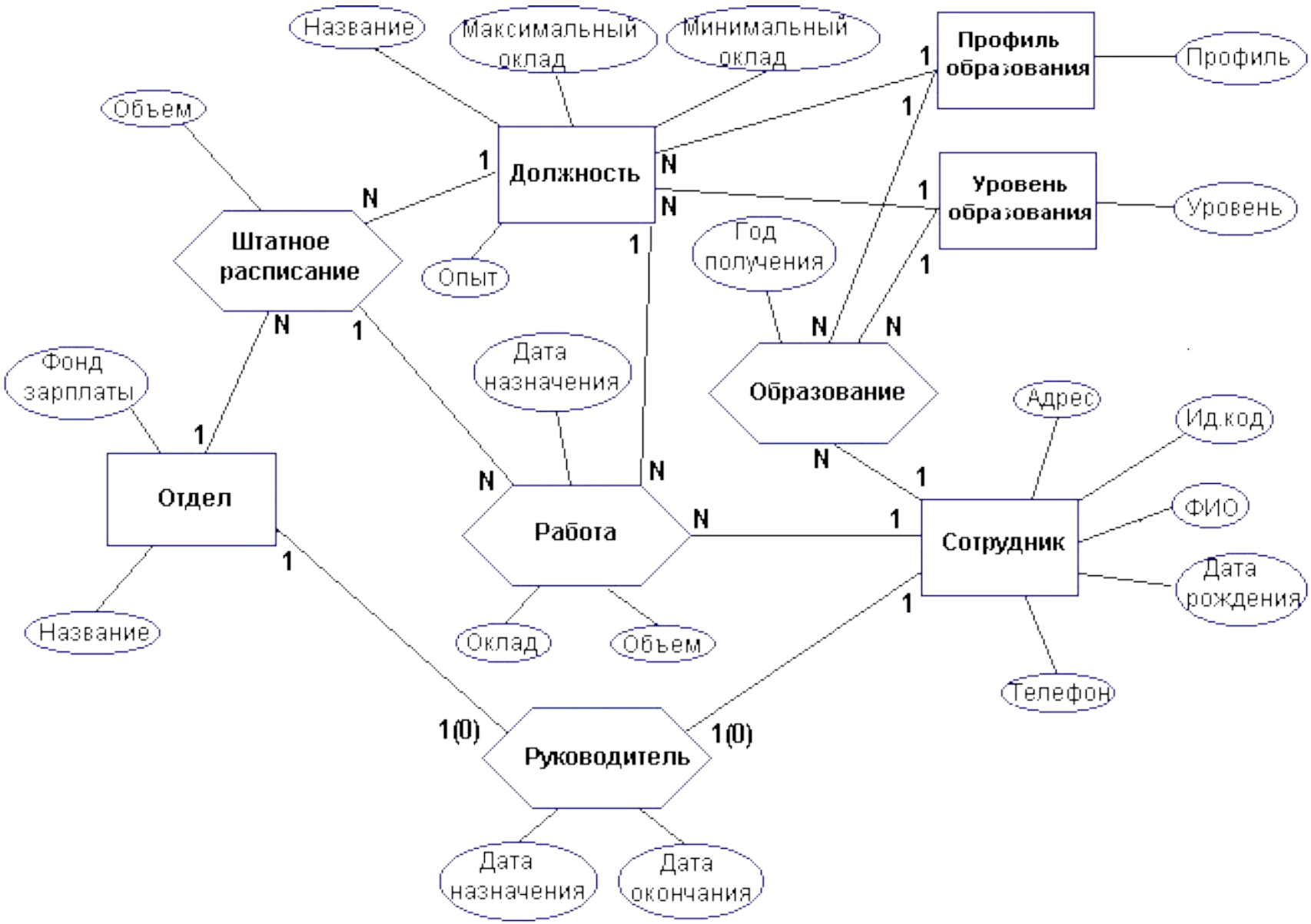
Этапы построения приложения БД-2

- Этап 2: **Концептуальное** моделирование
  - Требуется язык моделирования, чтобы выразить то, что мы хотим
  - ER модель данных – наиболее популярный язык для этих целей
  - выход: ER диаграммы предметной области

Лаб.2 - концептуальная схема данных - 1



Лаб.2 - концептуальная схема данных - 2



Этапы построения приложения БД-3

- Этап 3: **отображаем** концептуальную схему предметной области в реляционную схему
  - используем набор правил для этого отображения
  - используем набор правил усовершенствования схемы данных, чтобы по реляционной схеме получить хорошую реляционную схему
- Выход:
  - имеем на «бумаге» хорошую реляционную схему

Этапы построения приложения БД-4

- Этап 4: реализуем РБД
  - используя язык РСУБД SQL DDL
    - **create table,**
    - **alter table,**
    - **delete table …**
  - Физическое моделирование, в основном через SQL DDL
    - управление размещением файлов РБД, создание индексов …
- Этап 5: манипулируем РБД
  - используя язык РСУБД SQL DML
    - **select**
    - **insert**, **update**
    - **delete**

Этапы построения приложения БД-5

- Следующие этапы могут включать:
  - язык SQL может оказаться недостаточным для реализации требуемых вам действий
    - тогда пишем прикладную программу на Java, C/C++, Delphi, и т.п., чтобы осуществить взаимодействие с РСУБД, чтобы выполнить все требуемые от приложения БД действия.

Итого, проектирование РБД - 1

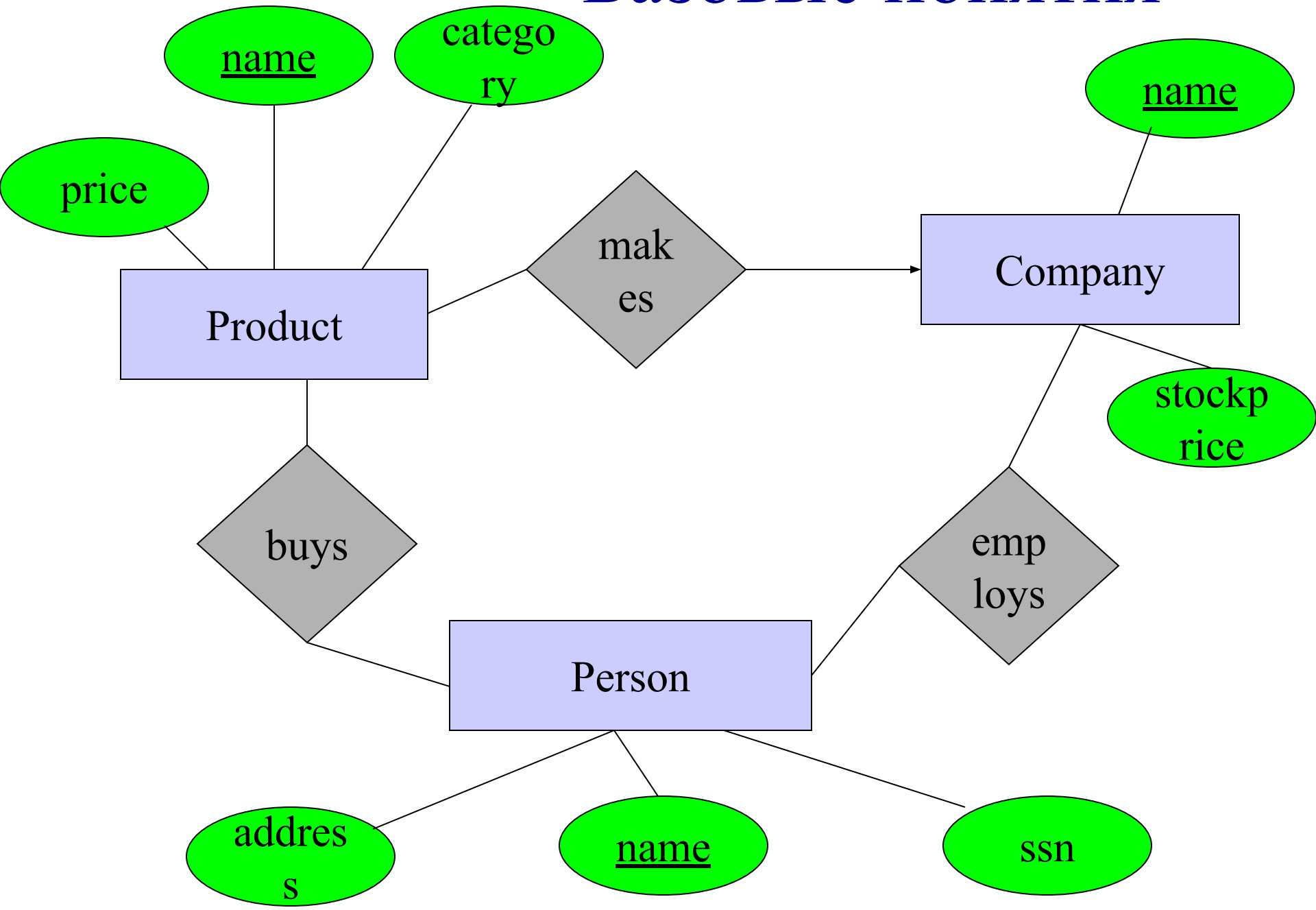
- **Анализ** предметной области
- **Концептуальное** моделирование
  - построение концептуальных схем(ER-диаграмм)
  - настройки/управление отображением
- **Логическое** моделирование
  - построение реляционных схем (ER-диаграмм
  - Схемы РБД
- **Физическое** моделирование
  - настройки/управление характеристиками физического размещения
- …

Обзор концептуального проектирования

Обзор концептуального моделирования

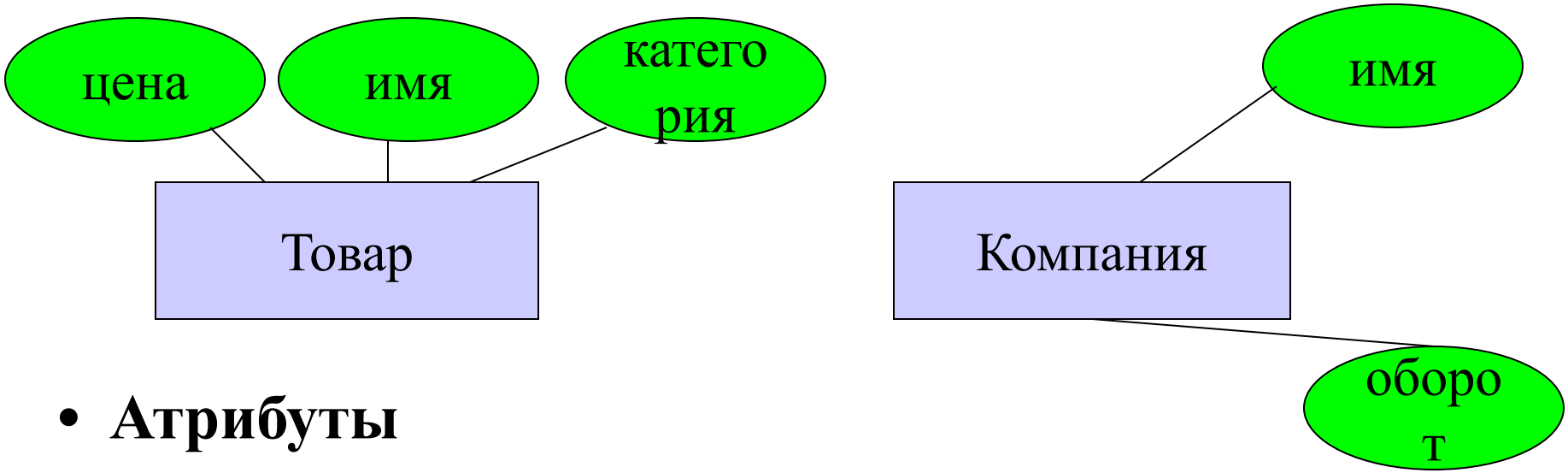
- Инфологическая модель данных "Сущность-связь"
- Основные понятия
  - **Сущность** – любой различимый объект
  - **Атрибут** – поименованная характеристика сущности.
  - **Ключ** – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.
  - **Связь** – ассоциирование двух или более сущностей.
- Графический язык описания концептуальных/ семантических/инфологических моделей
  - ER-диаграммы (Entity-Relationship) - запись описания предметной модели средствами ER-модели (Питер Чен)

Базовые понятия



Сущности и атрибуты

- **Сущности**
  - объекты реального мира, различимые с другими объектами
  - Описываются, используя набор атрибутов



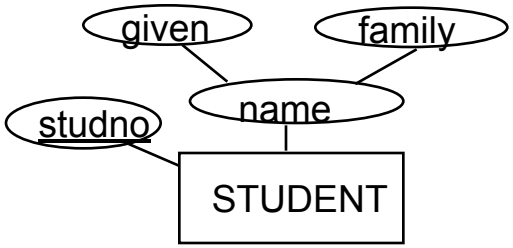
- **Атрибуты**
  - каждый имеет значения элементарных типов данных: строка, целые, вещественные, время/даты и т.п.
- **Множество сущностей**: коллекция аналогичных сущностей

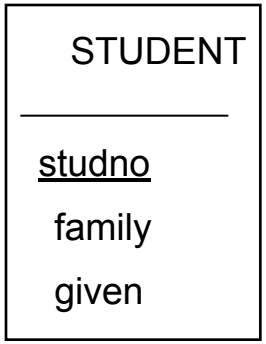
Сущность

- реальный или представляемый объект, существующий сам по себе, отличимый от других объектов.
  - представляется **типом сущности**
  - которому соответствует набор однородных **экземпляров сущности**
- *физические объекты, события, деятельность, ассоциации/взаимосвязи*
- изображаются
  - помеченными (именем сущности) прямоугольниками

Атрибут

- поименованная характеристика / свойство сущности.
- любая деталь, которая служит для
  - уточнения,
  - идентификации,
  - классификации,
  - числовой характеристики
  - выражения состояния сущности.
- изображаются
    - помеченными (имена атрибутов) овалами
  - или
    - *имена атрибутов заносятся в прямоугольник сущности*
- абсолютное различие между типами сущностей и атрибутами отсутствует.
  - Атрибут является таковым только в связи с типом сущности.
  - В другом контексте атрибут может выступать как самостоятельная сущность.

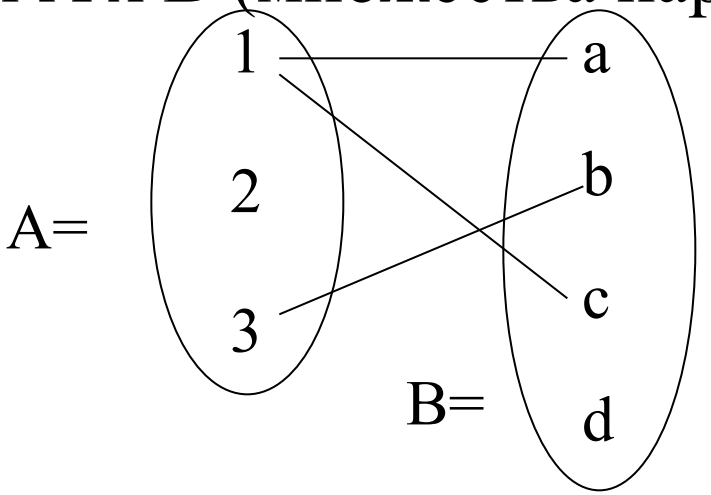




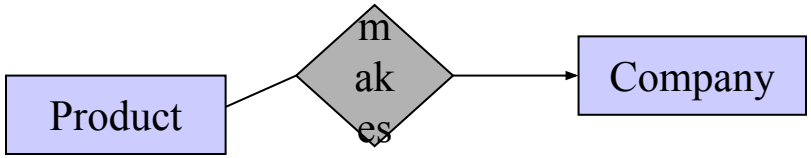
Ассоциации/отношения

- Математическое определение:
  - Если A, B - множества, то отношение R – это подмножество произведения A x B (множества пар элементов A, B)
- A={1,2,3}, B={a,b,c,d},

  R = {(1,a), (1,c), (3,b)}

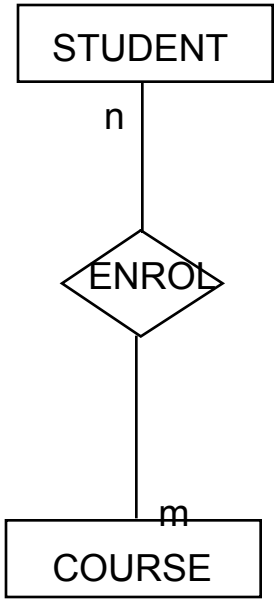


Подмножество **Product** x **Company** выражается:



Ассоциации

- ассоциация, устанавливаемая между двумя или более сущностями.
  - *бинарная связь*
- обеспечение возможности отыскания одних сущностей по значениям других
- изображаются
  - помеченными ромбами или шестиугольниками
- [могут иметь атрибуты]

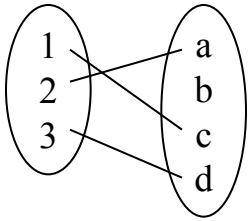


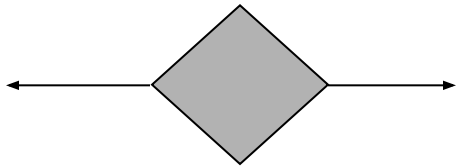
Связь

- бинарная, устанавливаемая между двумя
  - сущностями, атрибутами, ассоциациями
  - *сущностями или между сущностью и ей же самой (рекурсивная связь)*
- изображаются
  - ненаправленными линиями, над которыми может проставляться
    - степень связи – 0, 1, *цифра*, M ("много")
    - необходимое пояснение.
- возможны три вида связей.

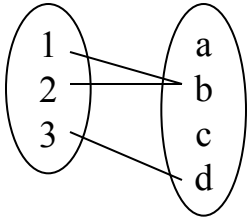
Множественность ER отношений/связей

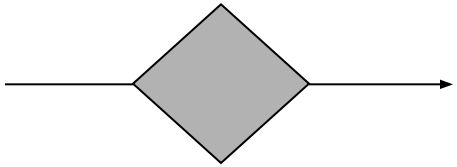
- один-к- одному:



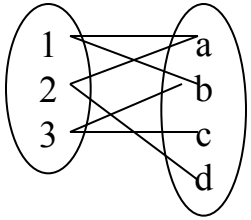


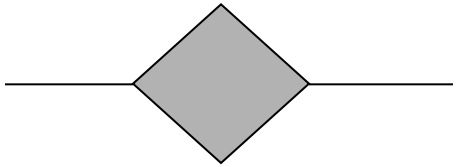
- многие-к-одному

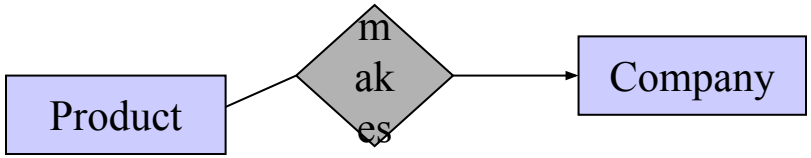




- многие-ко-многим





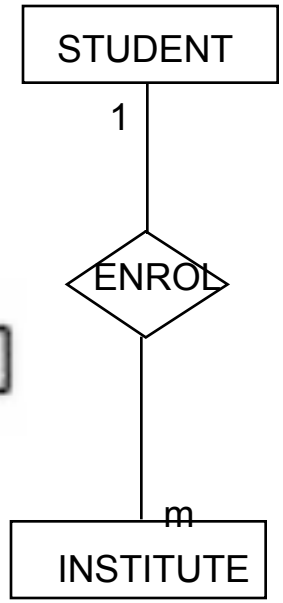


Связь **один-к-одному** (1:1)

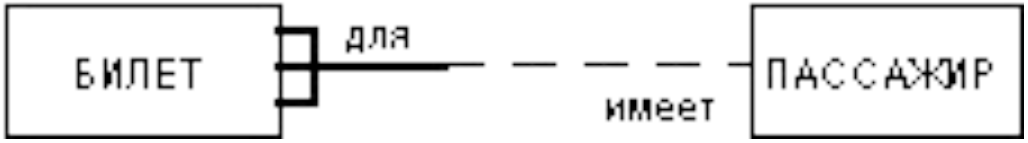
- связи может участвовать только один экземпляр
- каждому экземпляру A соответствует 1 [или 0] представителей B
- одноточечный вход
  - конец связи
    - обязательный
      - изображается сплошной линией,
      - указывается степень - 1
    - необязательный
      - изображается прерывистой линией.
      - указывается степень - 0

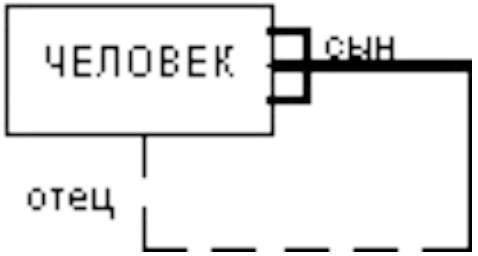
Связь **один-ко-многим** (1:M)

- одному представителю сущности A соответствуют 0, 1 или несколько представителей сущности B.
  - для сущности B в связи могут использоваться много экземпляров

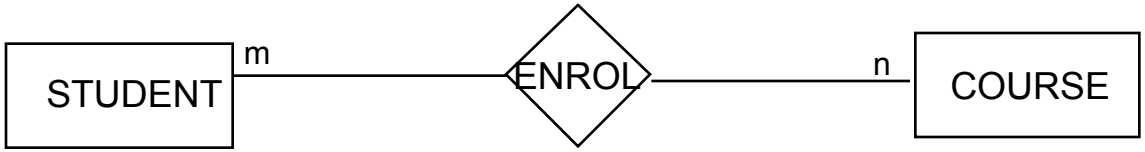


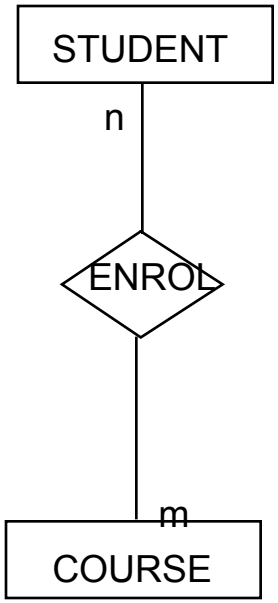
- *используются трехточечный вход в прямоугольник сущности*





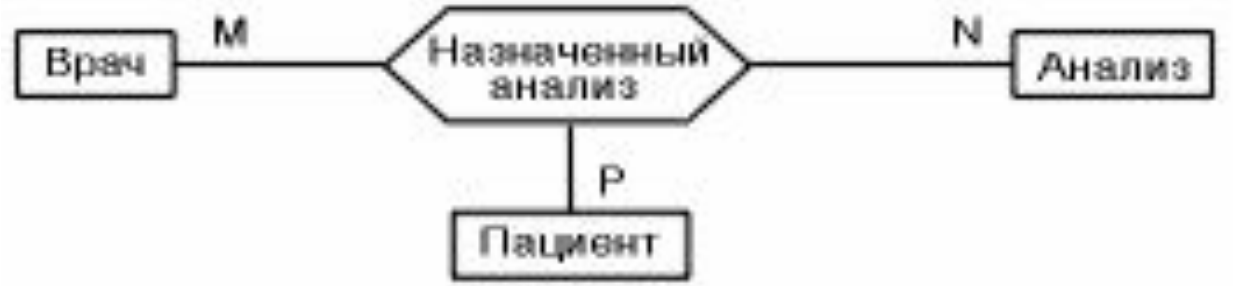
Связь **многие-ко-многим** (M :M)





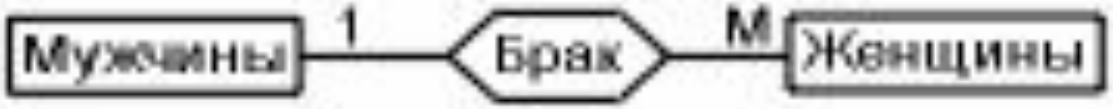
- множество связей между одними и теми же сущностями

- тернарные связи

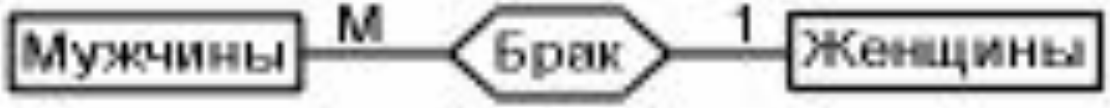


Многообразие связей

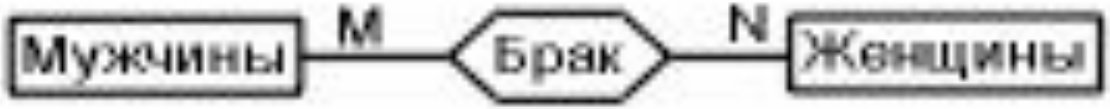
моногамный брак



многоженство



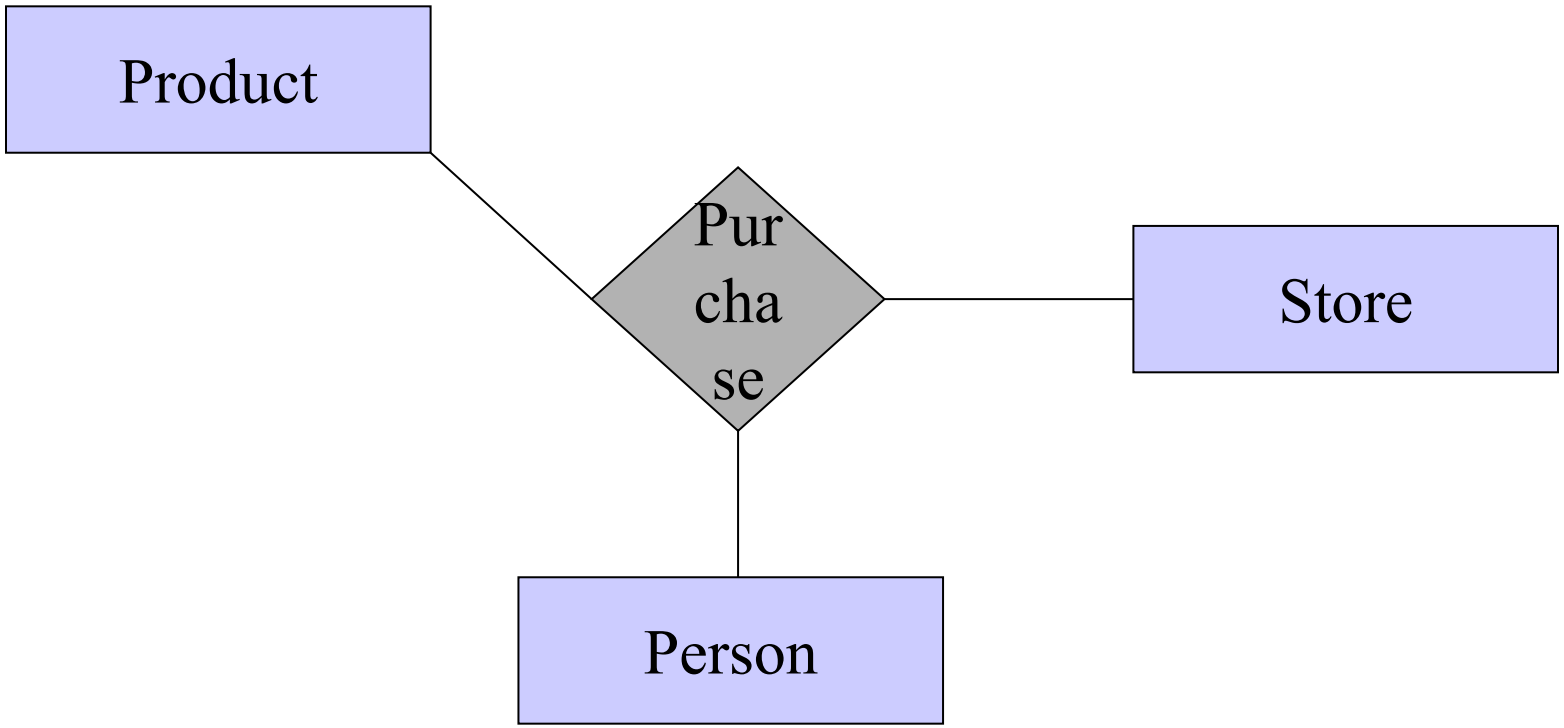
многомужие



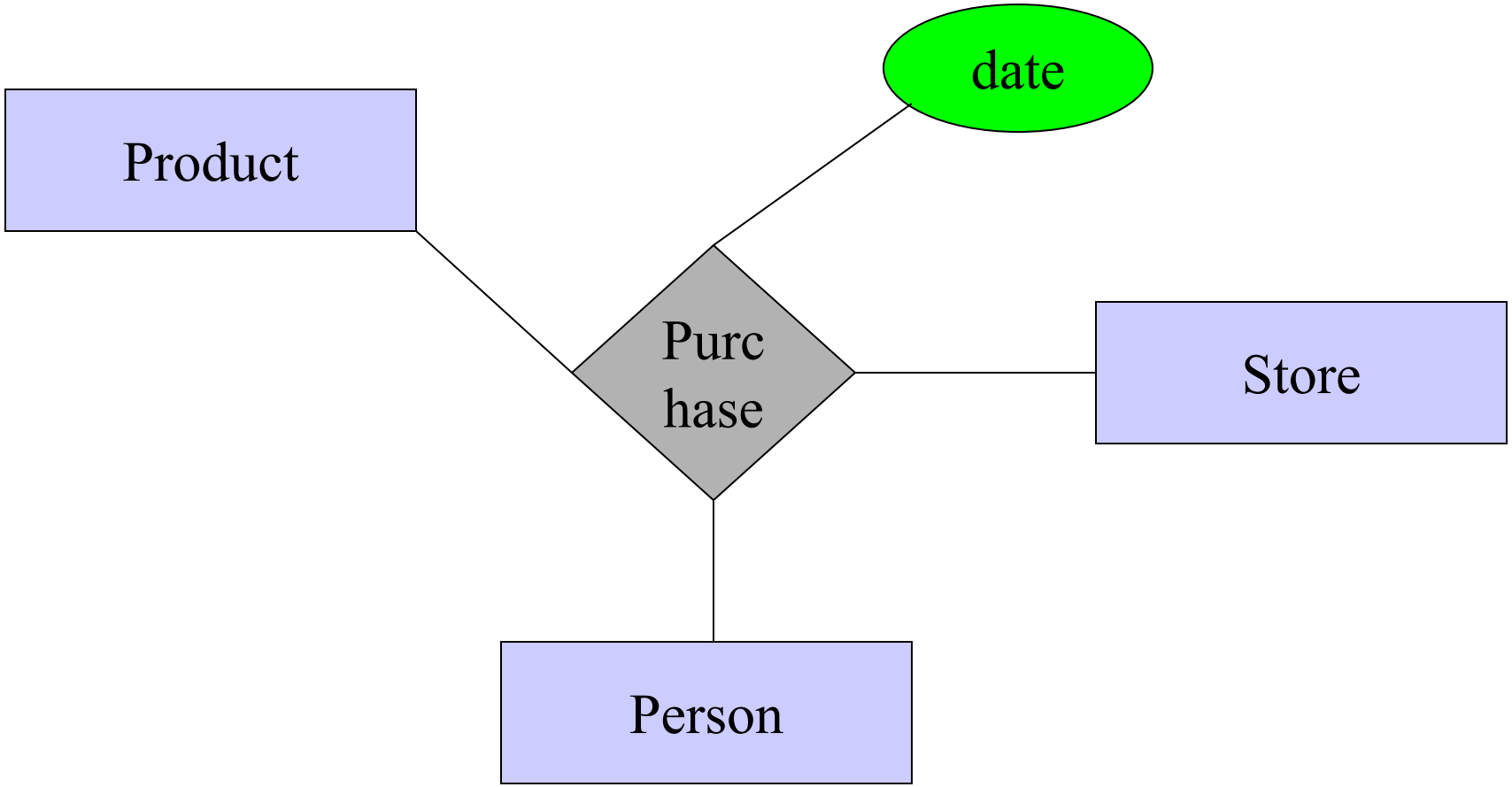
полигамный брак

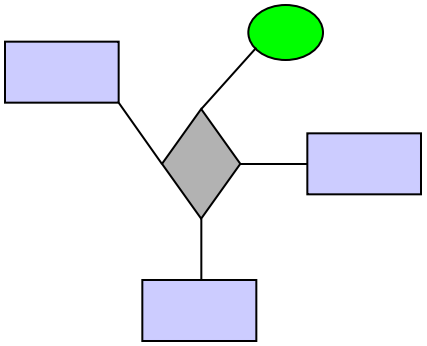
Многосторонние связи

Как моделировать отношение покупки(purchase) между покупателями (buyer), товарами(product) и магазинами(store)?

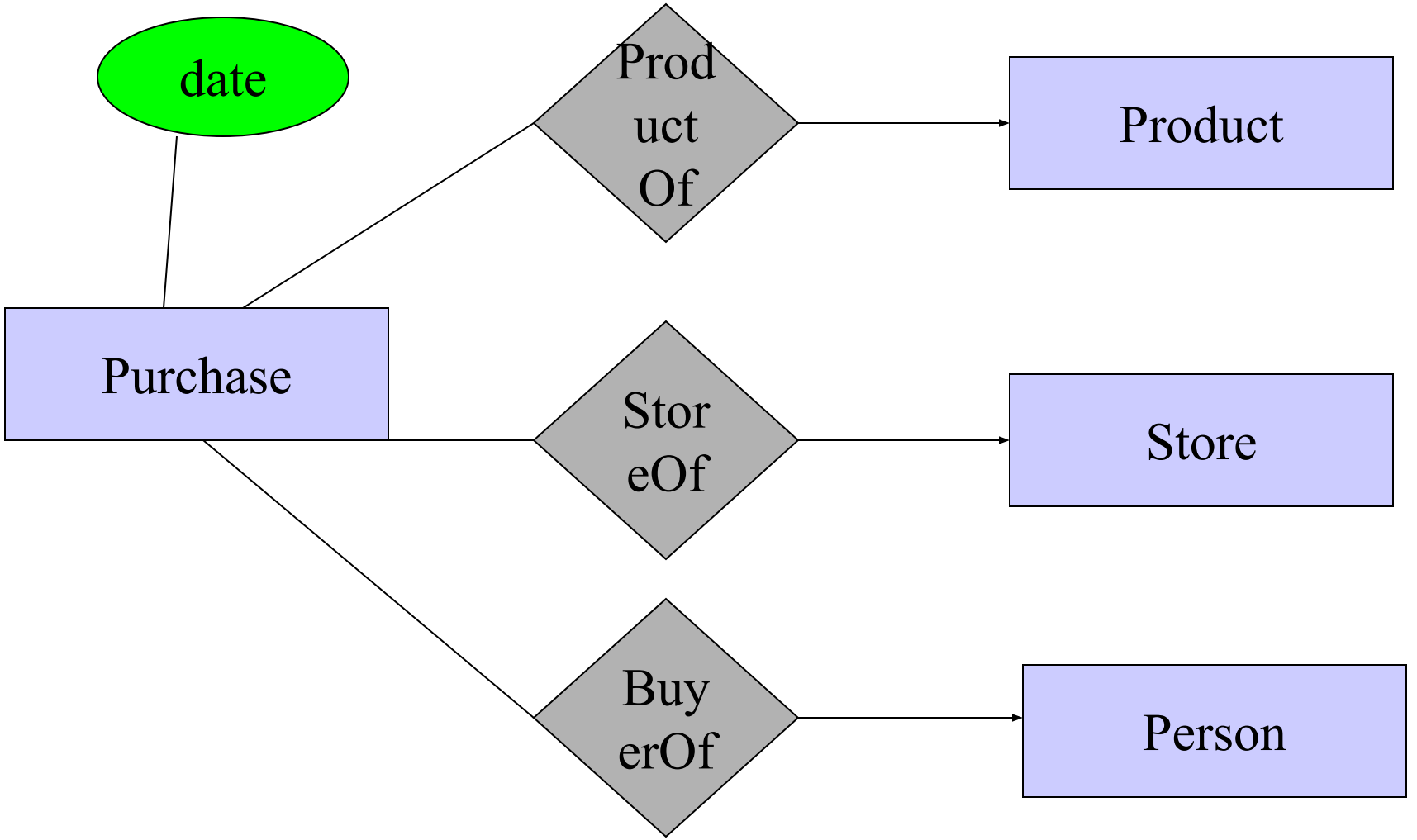


Атрибуты связи





Преобразование n-арных связей в бинарные



Связи: резюме

- Отношения моделируются как математическое множество
- имеются бинарные и n-арные связи
- n-арные связи можно выразить через бинарные
- ограничения на степень связи
  - многие-к-одному, один-к-одному, многие-ко-многим
  - ограничения стрелок
- атрибуты связей
  - Необязательны, но полезны

Ограничения целостности данных

Ограничения целостности данных

- Ограничения = утверждения о том, что обязательно должно быть **истинным** в массиве данных, в БД
- Указываются в **схеме** БД
  - Ограничение – это взаимосвязь между элементами данных, поддержка которых возлагается на СУБД.
- Важный аспект проектирования БД

Почему ограничения важны?

- Представляют больше **смысла/семантики** данных
  - помогают лучше понять данные
- Позволяют ссылаться на сущности (ключи)
- Дают возможность обеспечить **эффективное** хранение, поиск, извлечение данных

Моделирование ограничений

Выявление ограничений – часть процесса моделирования

Обычно используемые отношения:

Ключи:

номер паспорта, ИНН уникально идентифицируют персону

Ограничения уникальности значений (на отдельные):

персона может иметь одну мать

Ограничения ссылочной целостности:

если персона является сотрудником компании,

то компания должна иметься в БД

Ограничения области значений:

возраст людей представляется значениями между 0 и 150

Ограничения общего вида: все остальное

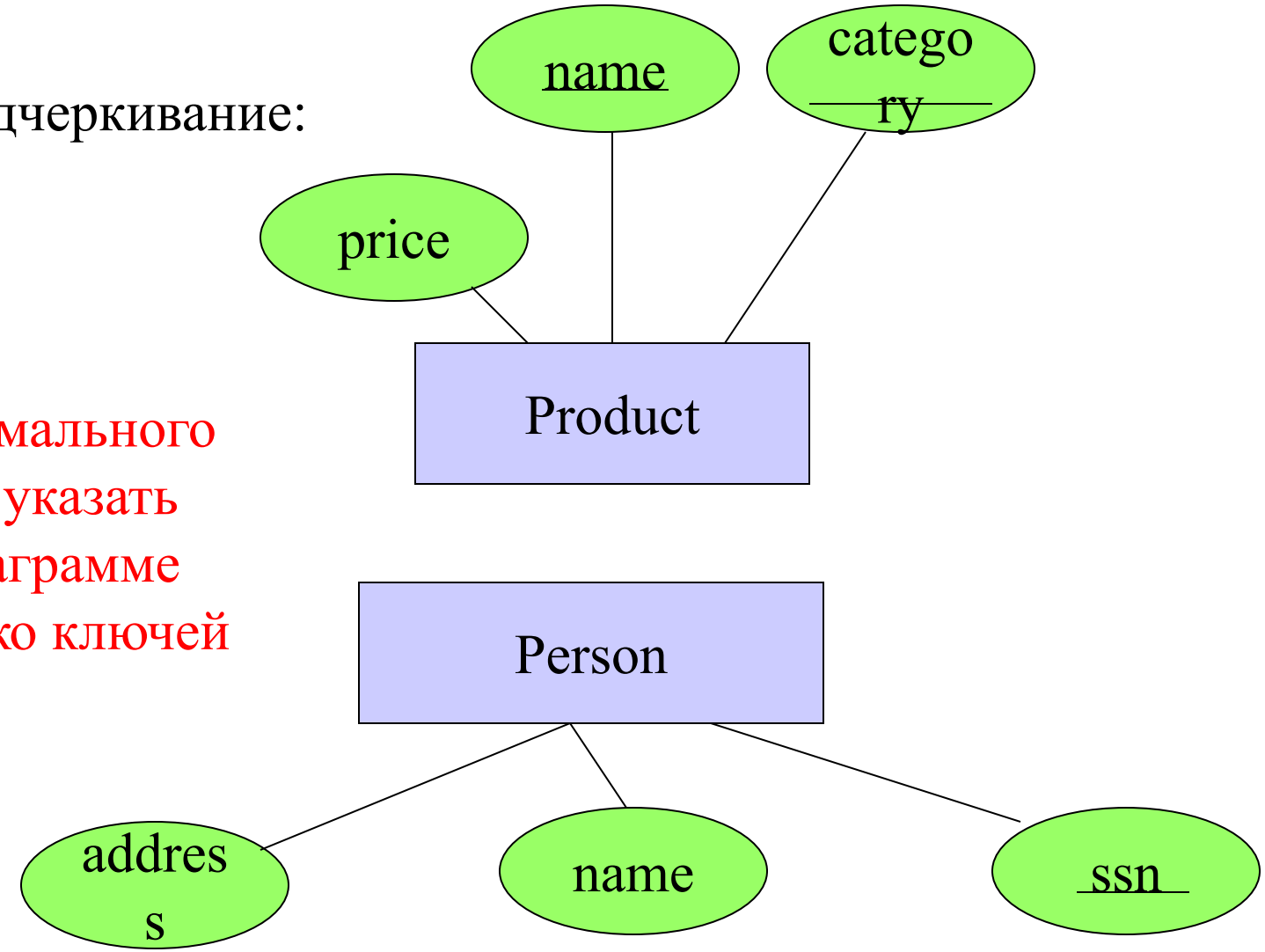
средняя цена пива не должна превышать 150

Ключи

- Каждая каждое множество сущностей должно иметь ключ
  - почему?
- Ключ может представляться более, чем одним атрибутом
- Может быть более одного ключа в множестве сущностей
  - Нельзя указать на ER-диаграмме
  - В РБД один из ключей является первичным

Ключи в E/R диаграммах

Подчеркивание:



Нет формального способа указать в ER диаграмме несколько ключей

Ограничения на отдельные значения

- Отдельное значение играет индивидуальную роль, представляет отдельный факт, конкретное понятие
- Атрибуты сущностей имеют одиночные значения
  - Можно указать является значение обязательным или нет (указывается NULL)
- Связи многие-к-одному, многие-ко-многим могут сопровождаться константой

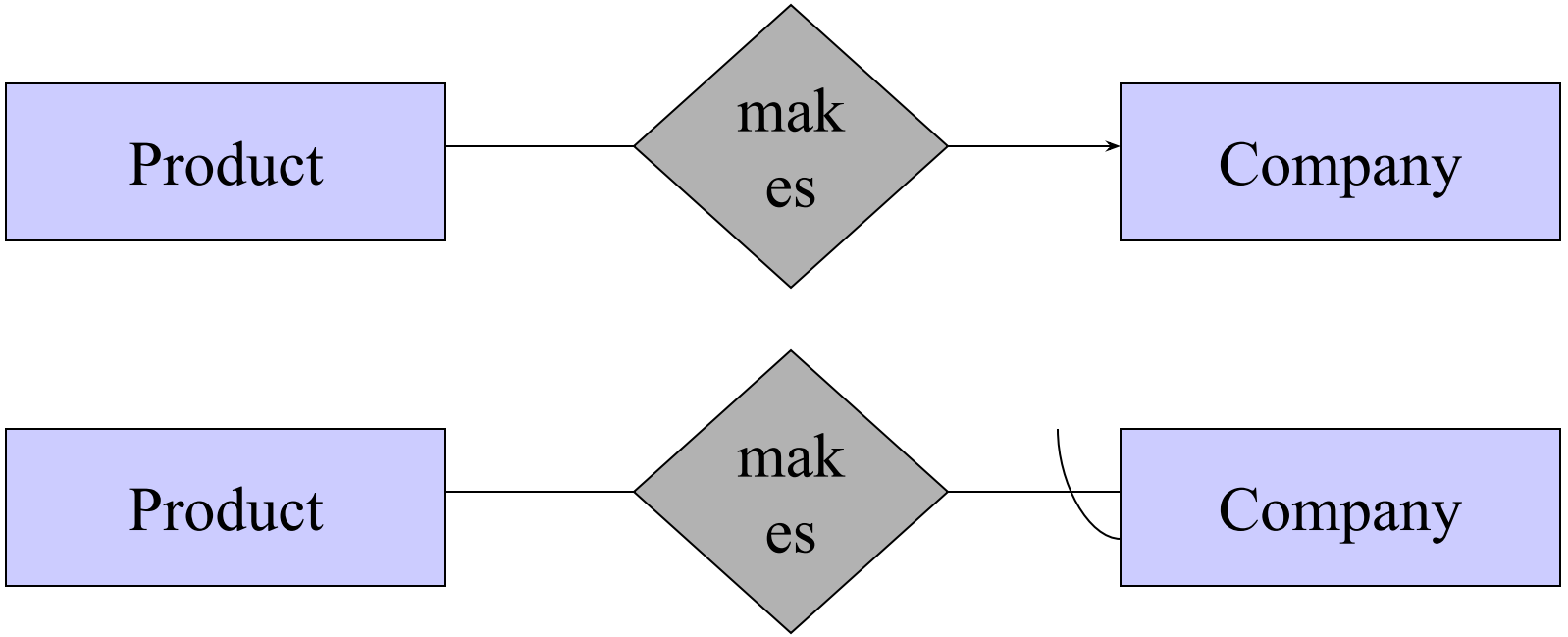
Ограничения ссылочной целостности

- **Ограничения на отдельные значения**
  - не более чем одно значение имеется в данной роли
- **Ограничения ссылочной целостности**:
  - ровно одно значение имеется в данной роли
- Если атрибут имеет обязательное значение
  - то это можно рассматривать как вид «ограничения ссылочной целостности».
- Наболее общее использование таких ограничений относится в связям

Ограничения ссылочной целостности

- В некоторых формализмах можно ссылаться на другие объекты и получать вместо них мусор
  - «висячие» указатели в C/C++
- «Ограничения ссылочной целостности» на связи явно требуют, чтобы ссылка существовала – указывала на существующий объект

Ограничения ссылочной целостности



Другие виды ограничений

- ограничения области значений
- более общие ограничения
  - можно указать комментриями

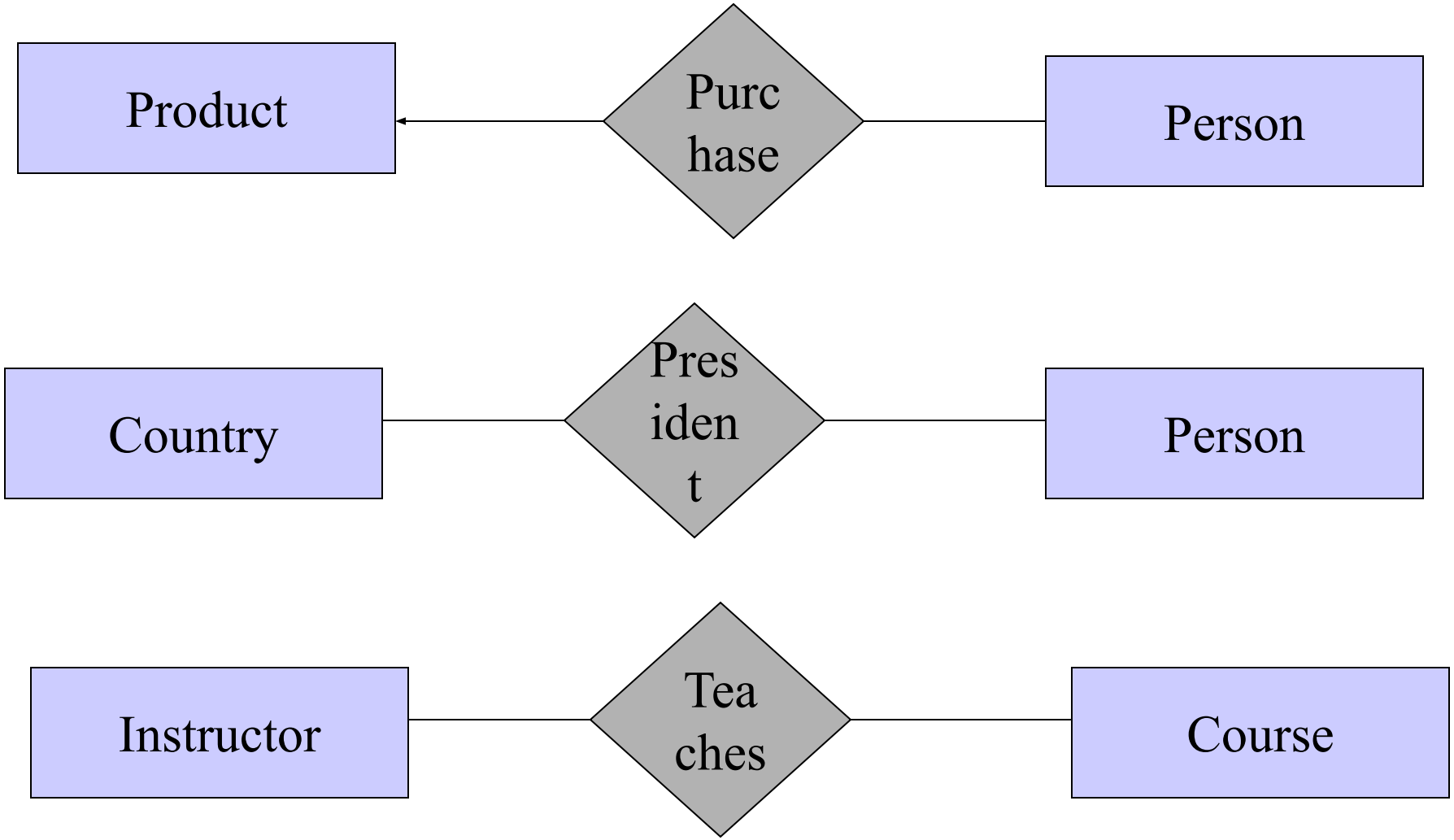
Слабые множества сущностей-2

- Иногда сущностям необходимо «помощь», чтобы обеспечить их уникальную идентификацию

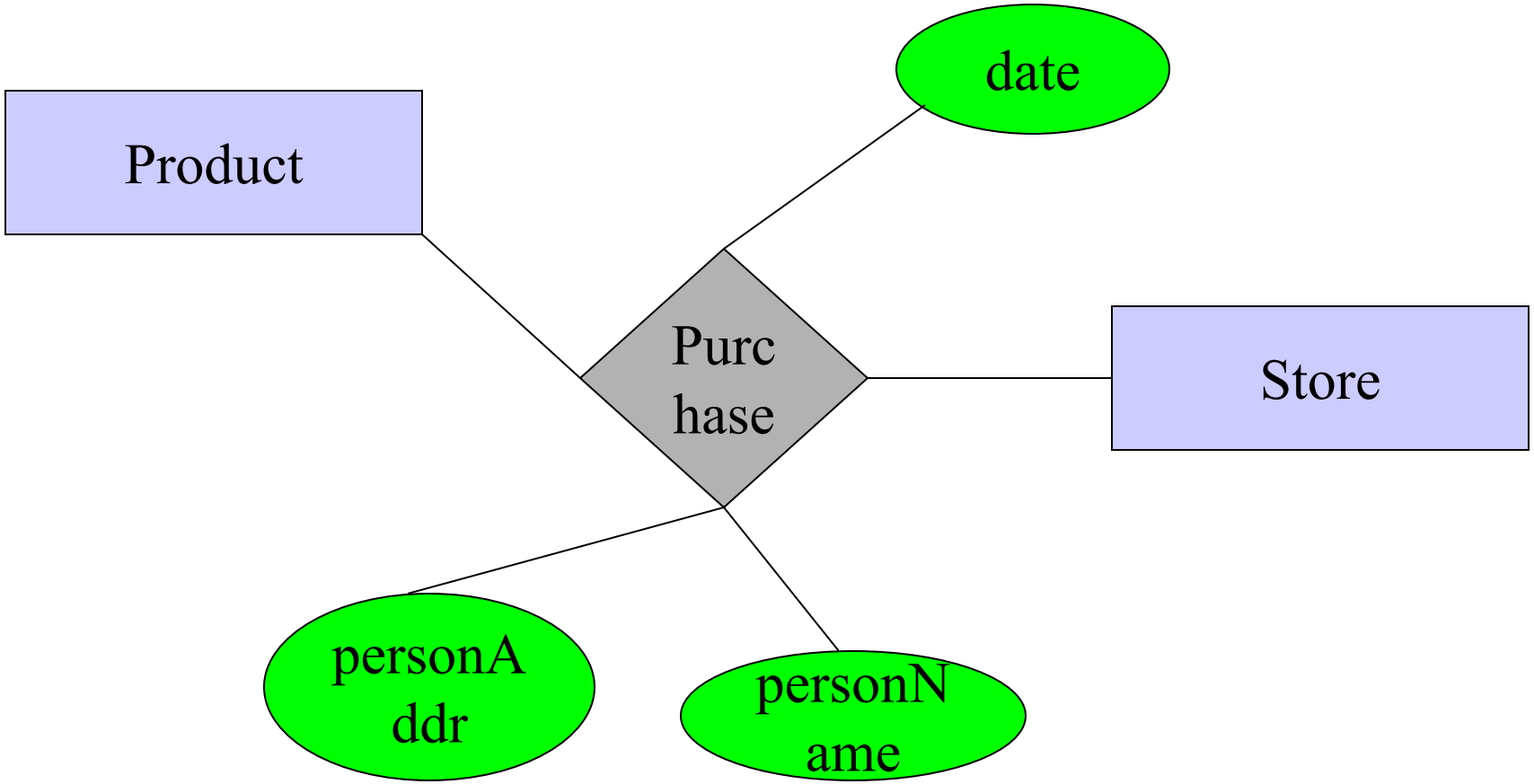
- Набор сущностей *E* называется *слабым* (*weak*), если для того, чтобы уникально идентифицировать сущности *E* , нам нужно «проследовать» от *E* по одной или нескольким связям и использовать ключ(и) соответствующих сущностей

Техники моделирования...

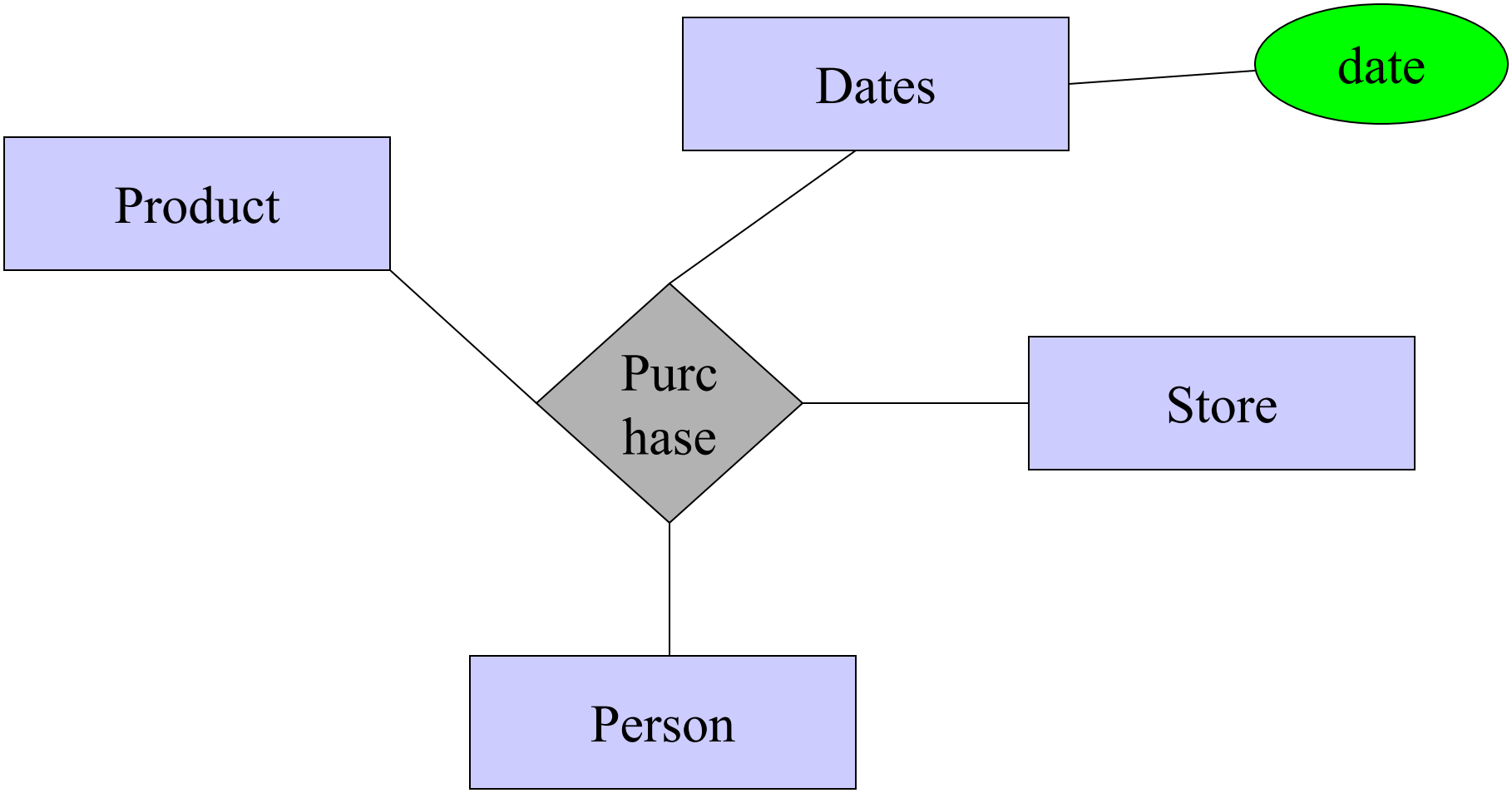
Принцип проектирования 1: **достоверность**, будь точен



Принцип проектирования 2: отсутствие избыточности



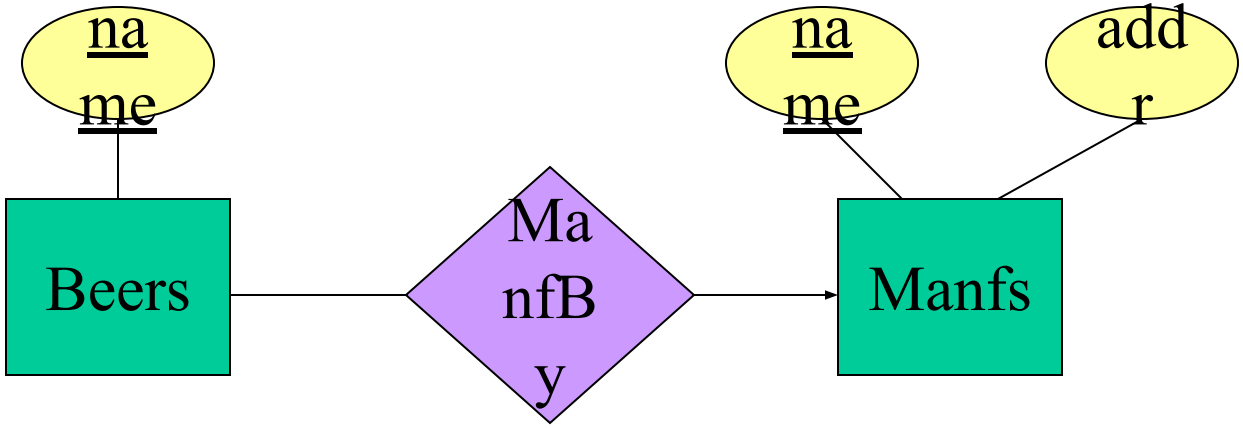
Принцип проектирования 3: адекватные сущности



Избегай избыточности

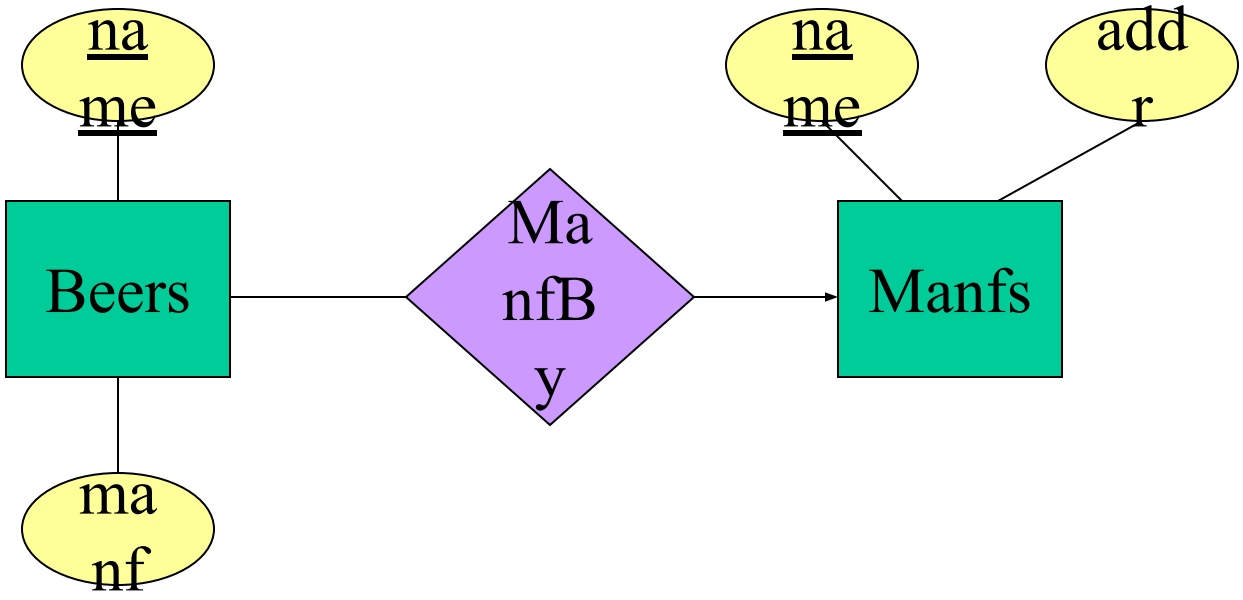
- Избыточность возникает, когда мы одно и тоже можем выразить/представить хотя бы двумя различными способами.
- Избыточность потребляет пространство и может привести к нарушению согласованности данных
  - Два экземпляра одного и того же факта могут стать противоречивыми, если мы изменим один и забудем другой, соответствующую версию.

Пример: хорошо



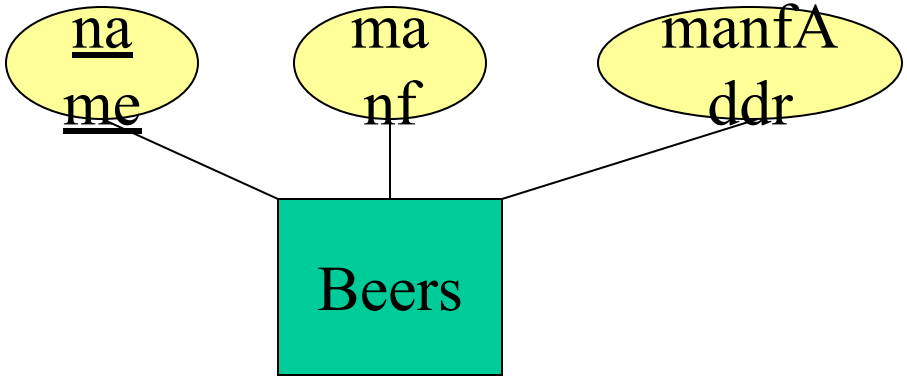
Производитель имеет ровно один адрес

Пример: плохо



Производитель пива указывается дважды:
как атрибут и как соответствующая сущность

Пример : плохо



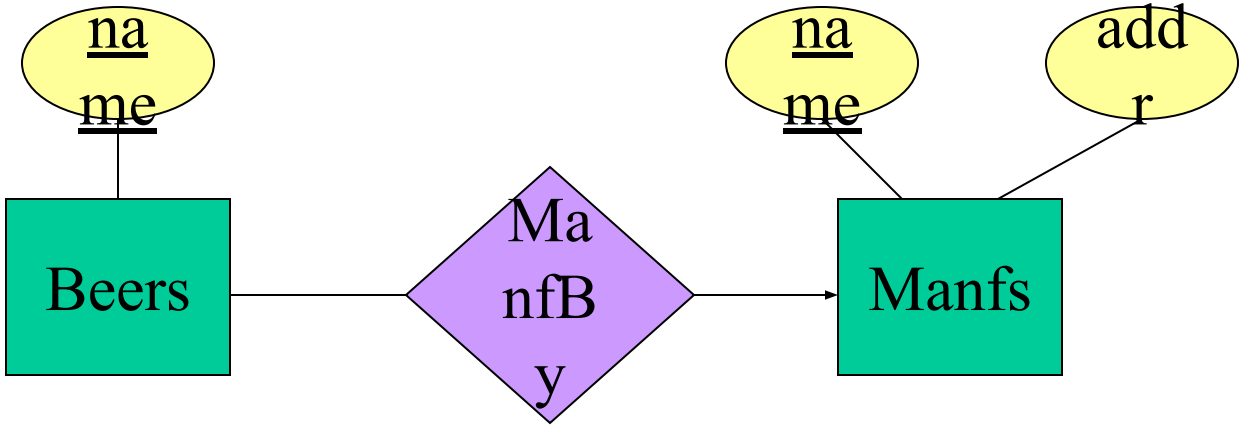
Повторяем адрес производителя каждый раз для каждого сорта пива.
Теряем адрес, если временно нет данных по сортам пива производителя

Множества сущностей vs. атрибуты

- Множество сущностей должно удовлетворять хотя бы одному из следующего:
  - имеет хотя бы один атрибут, не относящийся к ключу, более чем «имя» чего-то

    или
  - представляет “много” в многие-к-одному или многие-ко-многим связи

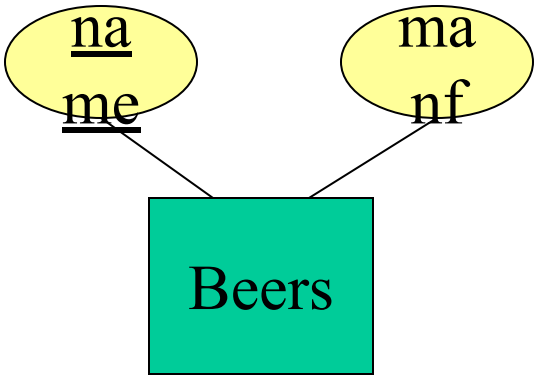
Пример : хорошо



- *Manfs* заслуживает того, чтобы быть сущностью, поскольку имеет неключевой атрибут *addr*.

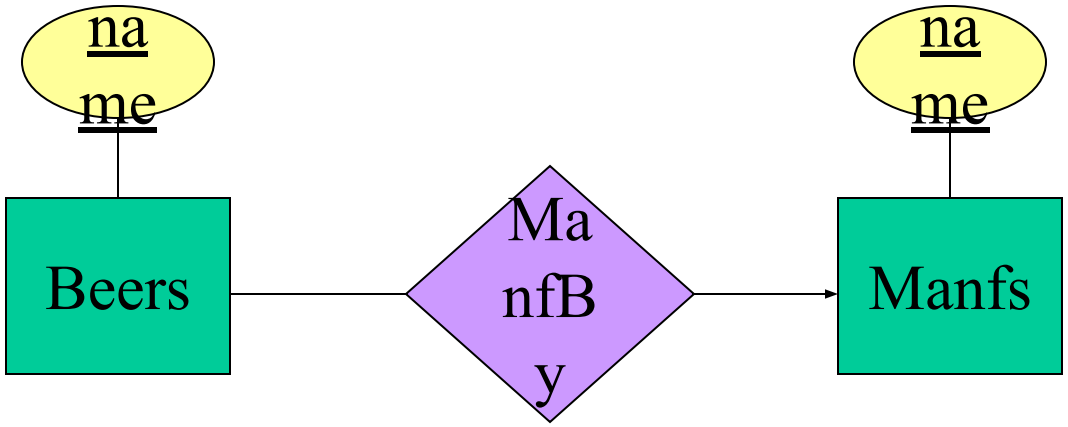
- *Beers* заслуживает того, чтобы быть сущностью, поскольку представляет “многие” в связи многие-к-одному с *ManfBy*

Пример: хорошо



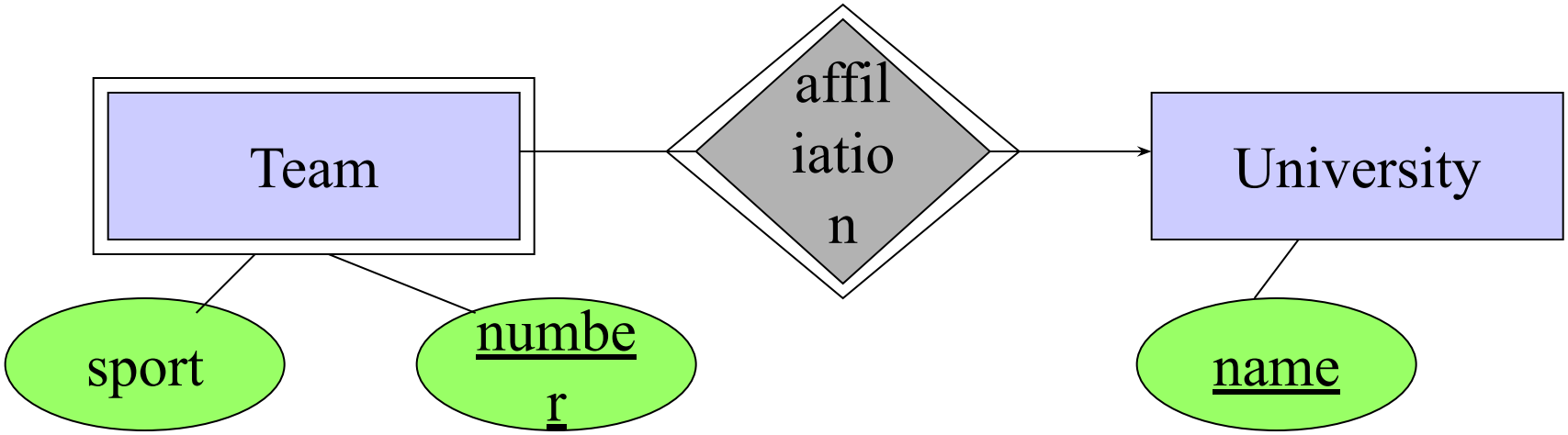
Производителя не нужно представлять сущностью, поскольку мы фиксируем о нем ничего более, чем имя

Пример: плохо



Поскольку фиксируем о производителе только имя и он не является концом “многие” в какой-либо связи, производитель не должен представляться множеством сущностей.

Слабые множества сущностей



отношение Team:

Sport	Number	AffiliatedUniversity
плавание	15	МФТИ

- нужны все атрибуты, что составляют ключ сущности Team
- не нужно отношение для Affiliation

Избегай чрезмерного использования слабых множеств сущностей

- Начинающие проектировщики часто сомневаются, может ли свойство быть ключем
  - Они делают все множества сущностей слабыми, которым содействуют все множества сущностей, с которыми первые связаны
- На практике мы обычно создаем уникальный ID для множеств сущностей
  - Номер паспорта, ИНН, авто.номер, номер двигателя

Когда нам нужны слабые множества сущностей?

- Обычная причина в том, что нет глобального авторитетного источника, способно формировать уникальные ID
- Маловероятно, что может быть достигнуто какое-согласие по сопоставлению уникальных номеров футболистам всех футбольных команд мира

ER резюме

- Основные элементы
  - сущности, атрибуты, множества сущностей
  - Связи/ассоциации: бинарные, n-арные, преобразование n-арных в бинарные
  - роли связей, атрибуты связей
  - подклассы (isa связи)
- Ограничения
  - на on связи
    - многие-к-одному, один-к-одному, многие-ко-многим
    - ограничения стрелок
  - ключи, отдельные значения, ссылочная целостность, области значений, ограничения общего вида

Построение ER-модели –1

- выявление потенциальных **сущностей**
- выявление **связей,** их типов
- выявление неявных сущностей, "замаскированных" под атрибуты
  - устраняются повторяющиеся атрибуты или группы атрибутов
- выявление уникального идентификатора сущности
  - атрибут, комбинация атрибутов, комбинация связей или комбинация связей и атрибутов, уникально отличающая любой экземпляр сущности от других экземпляров сущности того же типа; «указывает» экземпляр
  - устраняются атрибуты, зависящие только от части уникального идентификатора.
- выявление основы отдельной сущности
  - устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор

Построение ER-модели - 2

- выявление подтипов и супертипов сущностей
  - моделирование наследования типа сущности
- разрешение связей «многие-ко-многим»
- уточняемые степени связей
- выявление сильных связей «один-ко-многим»
  - каскадные удаления
- определения потенциально допустимого множества значений атрибута сущности (домена)
  - ограничения целостности значений атрибутов
- определение ограничений целостности сущностей и связей
  - процедурные действия

Трансляция ER модели в реляционную

Трансляция ER диаграмм в реляционную схему

- Базовые варианты
  - Множество сущностей E = отношение с атрибутами E
  - связь R = отношение с ключами связываемых множеств сущностей + непосредственно атрибуты R
- Специальные случаи
  - комбинирование двух отношений
  - трансляция слабых множеств сущностей
  - трансляция **isa** связей и подклассов

NULL значения

NULL значения

- Кортежи SQL отношений могут содержать NULL в качестве значения.
- Смысл зависит от контекста. Два общих случая:
  - *Пропущено/Неизвестно*: знаем бар «Встреча» имеет адрес, но не знаем какой.
  - *Неприменимо* : значение атрибута *spouse* (супруга) для холостого мужчины.
- Если x=NULL, то 4*(3-x)/7 - NULL

Сравнение NULL со значениями

- Логические условия в SQL относятся к трехзначной логике :
  – TRUE, FALSE, UNKNOWN (*неопределенно*).
- Когда некоторое значение сравнивается с NULL, значение результата - UNKNOWN.

- Если x=NULL, то (x='Встреча') – UNKNOWN

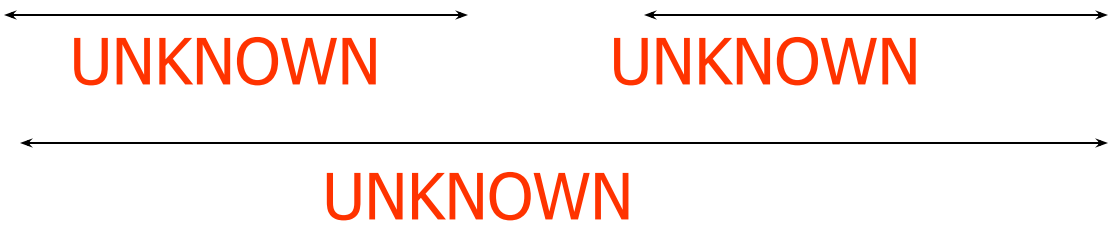
- Результат запроса включает кортеж, если значение выражения WHERE части
  – TRUE (ни FALSE, ни UNKNOWN).

Пример

- Из отношения Sells:

bar	beer	price
Встреча	Харп	NULL

```
SELECT bar
FROM Sells
WHERE price < 20.0 OR price >= 20.0;
```



Бар «Встреча» не попадает в результат.

Что делать?

Проверка на Null

Можно явно проверить на наличие NULL значения:

- *x* IS NULL
- *x* IS NOT NULL

```
SELECT bar
FROM   Sells
WHERE  price < 20.0 OR price >= 20.0
                    OR price IS NULL
```

Включены все бары.

Трехзначная логика

- Чтобы понять, что представляют AND, OR, и NOT в трехзначной логике можно считать, что
- **TRUE = 1, FALSE = 0, UNKNOWN = ½.**
- ***x* AND *y* = min(*x*, *y*)**
  ***x* OR *y* = max(*x*, *y*)**
  NOT(*x*) = 1-*x*
- Пример:

TRUE AND (FALSE OR NOT(UNKNOWN))

=

MIN(1, MAX(0, (1 - ½))) =

MIN(1, MAX(0, ½) = MIN(1, ½) = ½.

двухначная логика != трехзначной логике

- Некоторые общие законы, например, коммутативность AND, сохраняется в трехзначной логике.
- Но некоторые нет, например, :
  “закон отсутствия середины,”
  – ***p* OR NOT *p* = TRUE**.
  – Если *p* = UNKNOWN, то получаем
  – **MAX(½, (1 – ½)) = ½** (!= 1)

Запросы с несколькими отношениями

Запросы с несколькими отношениями

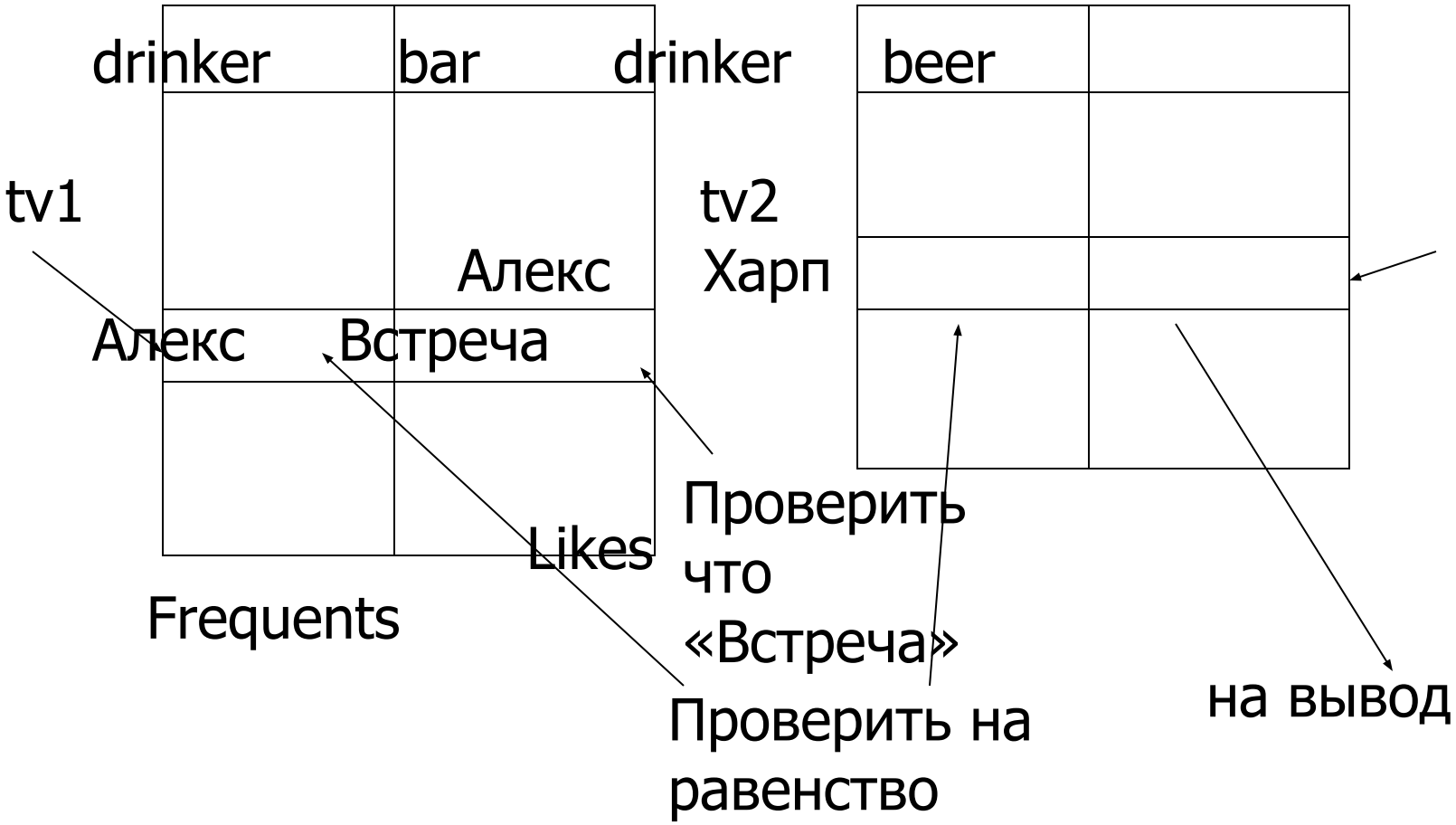
- Часто требуемые запросы должны комбинировать данные из более одного отношения
- Можно обратиться к нескольким отношениям в одном запросе указав их всех в FROM части
- Чтобы различить одноименные атрибуты разных отношений, следует указывать отношение
  - **<отношение>.<атрибут>**

Пример

- Используя отношения Likes(drinker, beer) и Frequents(drinker, bar), найти сорта пива, предпочитаемые посетителями бара «Встреча»

```
SELECT beer
FROM Likes, Frequents
WHERE bar = 'Встреча' AND
Frequents.drinker = Likes.drinker;
```

Пример


drinker
bar
drinker
beer
tv1
tv2
Алекс
Харп
Алекс
Встреча
Likes
Frequents
Проверить что «Встреча»
Проверить на равенство
на вывод

Еще пример

Product (pname, price, category, maker)
Purchase (buyer, seller, store, product)
Company (cname, stockPrice, country)
Person(pname, phoneNumber, city)

Найти персон, живущих в Долгопрудном, покупающих компьютеры, указать наименования магазинов, в которых делаются эти покупки

```
SELECT   pname, store
FROM     Person, Purchase
WHERE    pname=buyer AND city='Долгопрудный'
         AND product='компьютер'
```

Одноименные атрибуты

Найти имена персон покупающих телефоны:

Product (name, price, category, maker)
Purchase (buyer, seller, store, product)
Person(name, phoneNumber, city)

```
SELECT   Person.name
FROM     Person, Purchase, Product
WHERE    Person.name=Purchase.buyer
     AND product=Product.name
     AND Product.category='телефон'
```

Псевдонимы

- Иногда в запросах нужно использовать две копии одного и того же отношения
- Чтобы различать копии, в FROM части после имени копии указывают «псевдоним» (алиас, alias).
- Можно переименовывать отношения для лучшей «читаемости» запросов

Пример

- В Beers(name, manf) найти все пары наименований пива одного и того же производителя.
  - не выводить пары вида (Харп, Харп).
  - выводить пары в алфавитном порядке, например, (Миллер, Харп), не (Харп, Миллер).

```
SELECT b1.name, b2.name
FROM Beers b1, Beers b2
WHERE b1.manf = b2.manf AND
 b1.name < b2.name;
```

Еще пример

Найти пары компаний, производящих товары одной и той же категории

```
SELECT  product1.maker, product2.maker
FROM    Product as product1, Product as product2
WHERE   product1.category=product2.category
    AND  product1.maker <> product2.maker
```

Product (name, price, category, maker)

Псевдонимы

Псевдонимы вводятся компилятором языка SQL автоматически

Product (name, price, category, maker)

```
SELECT  name
FROM    Product
WHERE  price > 100
```

Становится:

```
SELECT Product.name
FROM   Product as Product
WHERE Product.price > 100
```

Не срабатывает, когда Product встречается более одного раза:
В этом случае пользователь должен явно определить псевдоним.

Формальная семантика запросов с несколькими отношениями

- Почти тоже самое, что для запросов с одним отношением:
  1. Начиная с произведения всех отношений, указанных в FROM части.
  2. применить условия отбора из WHERE части
  3. выполнить проекцию на список атрибутов и выражений в SELECT части

Операционная семантика запросов с несколькими отношениями

- Последовательно перебираем записи всех таблиц, указанных в FROM части
  - Кортеж-переменная для каждого отношения
  - Кортеж-переменные перебирают все комбинации кортежей (вложенные циклы)
- Проверяем, удовлетворяют ли набор текущих записей условиям WHERE части
- Если да, в соответствии с требованиями SELECT части выбираем значения атрибутов, вычисляем выражения, используя элементы текущих записей

Вложенные циклы

SELECT $a_1, a_2, \ldots, a_k$
FROM R_1 as x_1, R2 as x_2, …, Rn as x_n
WHERE Conditions

1. Вложенные циклы:

```
Answer = {}
for x1 in R1 do
    for x2 in R2 do
        …..
            for xn in Rn do
                if Conditions
                    then Answer = Answer U {(a1,…,ak)
return Answer
```

Параллельное вычисление

SELECT $a_1, a_2, \ldots, a_k$
FROM R_1 as x_1, R2 as x_2, …, Rn as x_n
WHERE Conditions

2. Параллельное вычисление

Answer = {}
for all assignments x_1 **in** R_1, …, x_n **in** R_n **do**
    if Conditions **then** Answer = Answer U $\{(a_1,\ldots,a_k)\}$
return Answer

Не предполагает какого-либо порядка

В реляционной алгебре

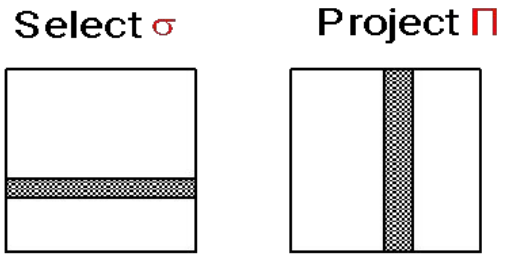
SELECT $a_1, a_2, \ldots, a_k$
FROM R_1 as x_1, R2 as x_2, …, Rn as x_n
WHERE Conditions

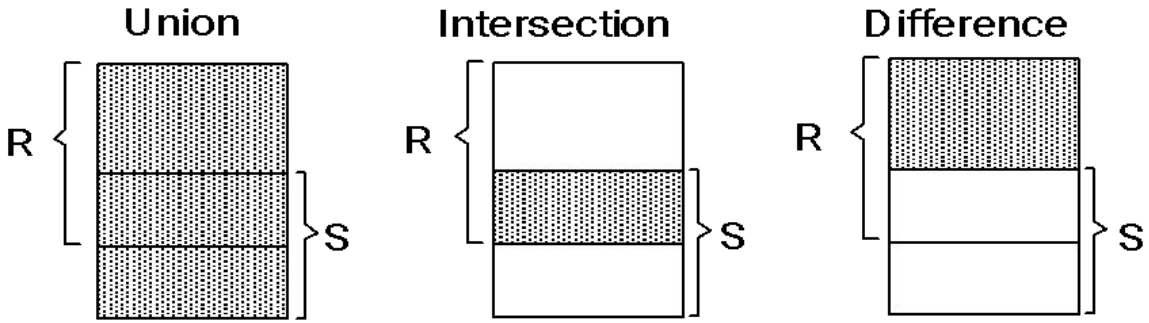
3. Трансляция в реляционную алгебру:

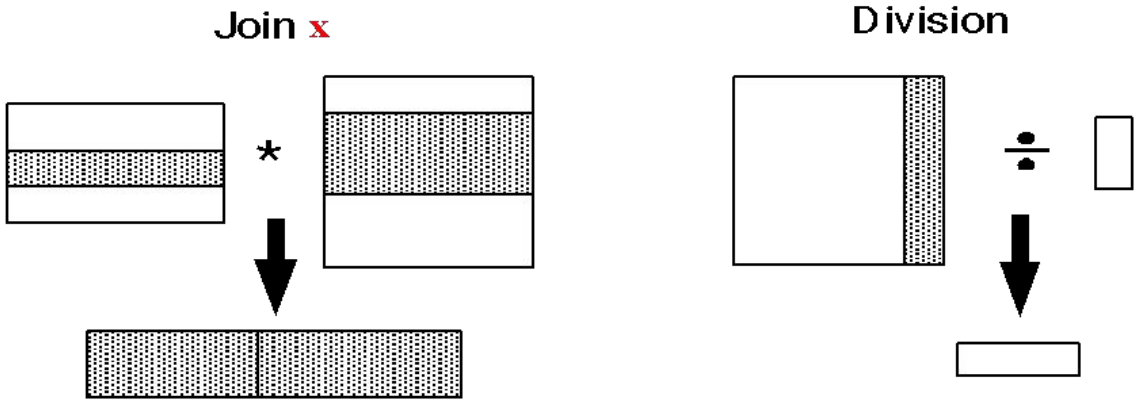
$$\Pi_{a1,\ldots,ak} (\sigma_{\text{Conditions}} (R_1 \times R_2 \times \ldots \times R_n))$$

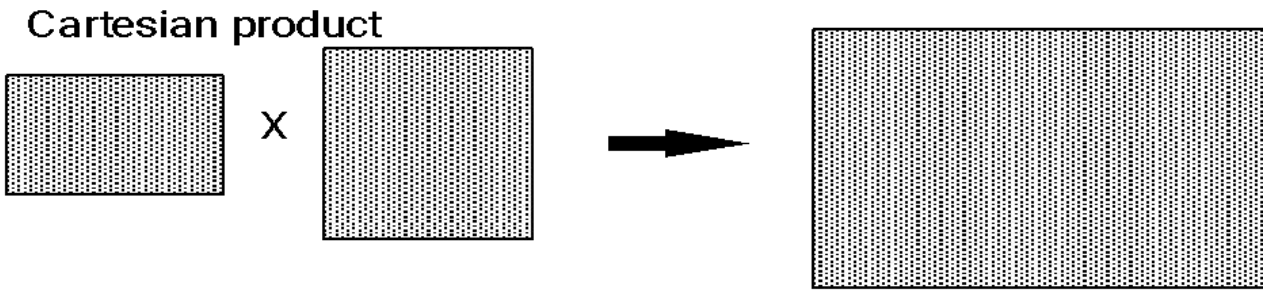
Select-From-Where = Select-Project-Join
запросы выражения

Реляционные операции









Подзапросы

Подзапросы

- SELECT-FROM-WHERE оператор в круглых скобках (*подзапрос*) может использоваться в качестве операнда в ряде мест, включая FROM и WHERE части.
- Пример: на место отношения в FROM части, можно поместить другой запрос и выполнить запрос к его результату.
  - Лучше использовать псевдонимы для именования кортежей результата

Подзапросы, возвращающие одну запись

- Если гарантируется, что запрос возвращает одну запись, то подзапрос может использоваться в качестве значения
  - обычно такие записи имеют одно поле
  - обычно единственность записи гарантируется первичным ключом.
  - если запрос не возвращает записей или возвращает больше одной, то возникает ошибка времени исполнения.

Пример

- В Sells(bar, beer, price) найти бары предлагающие «Миллер» по той же цене, что «Встреча» продает «Харп».
- Два запроса обеспечивают ответ:
  1. найти стоимость «Харп» в баре «Встреча»
  2. найти бары продающие «Миллер», по этой цене

Запрос + Подзапрос

```
SELECT bar
FROM Sells
WHERE beer = ‘Миллер’ AND
 price = (SELECT price
          FROM Sells
          WHERE bar = ‘Встреча’
          AND beer = ‘Харп’);
```

Цена продажи «Харп» в баре «Встреча»

Логические операторы IN, EXISTS, ANY, ALL

Оператор IN

- <кортеж> IN <отношение>
  истинно, если кортеж содержится в отношении.
- <кортеж> NOT IN <отношение>
  если кортеж не содержится в отношении

- очень часто <отношение> – это подзапрос

- IN-выражения могут использоваться в WHERE части

Пример

- В Beers(name, manf) и Likes(drinker, beer) найти наименование и производителя сортов пива, предпочитаемых Алексом.

```
SELECT *
FROM Beers
WHERE name IN
    (SELECT beer FROM Likes
     WHERE drinker = 'Алекс');
```

Набор сортов любимых Алексом

Оператор Exists

- EXISTS(<отношение>) истинен, тогда и только тогда, когда <отношение> не пусто
- EXISTS логический оператор, может входить в WHERE часть
- Пример: В Beers(name, manf) найти сорта пива, уникальные для их производителя (им производится только этот сорт пива)

Пример запроса с EXISTS

```
SELECT name
FROM Beers b1
WHERE NOT EXISTS(
   SELECT *
   FROM Beers
   WHERE manf = b1.manf AND
      name <> b1.name);
```

Области видимости: manf относится к непосредственно охватывающему FROM с отношением, имеющем этот атрибут

Выбирает сорта пива с тем же производителем, что и b1, но с другим сортом

Ссылка на b1 – запросы как вложенные циклы.

Оператор ANY

- *x* = ANY(<отношение>) логическое условие, означающее, что *x* равен хотя бы одному кортежу отношения
- вместо = может стоять любой оператор сравнения.
  - Пример : *x* >= ANY(<отношение>) означает, что *x* не меньше, чем все кортежи отношения
- В кортежах отношения ДОЛЖЕН быть только один атрибут

Оператор ALL

- *x* <> ALL(<отношение>) истинен, тогда и только тогда, когда для каждого кортежа *t* отношения, *x* не равен *t*.
  - то есть *x* не входит в отношение
- вместо <> может стоять любой оператор сравнения
  - Пример: *x* >= ALL(<отношение>) означает, что в отношении нет кортежей больших, чем *x*
- В кортежах отношения ДОЛЖЕН быть только один атрибут

Пример

- В Sells(bar, beer, price) найти самые дорогие сорт(а) пива

```
SELECT beer
FROM Sells
WHERE    price >= ALL(
    SELECT price
    FROM Sells);
```

Цена пива для охватывающего Sells не должна быть меньше, чем какая-либо цена пива

Все

Базовые кафедры ФУПМ

Наши базовые кафедры

- МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ ПРОЦЕССОВ И СИСТЕМ
  - на базе ВЦ РАН, зав. кафедрой член-корр. И.Г.Поспелов.
- МАТЕМАТИЧЕСКИХ ОСНОВ УПРАВЛЕНИЯ
  - на базе МФТИ, зав. кафедрой к.ф.-м.н. С.А.Гуз
- КАФЕДРА НЕЛИНЕЙНЫХ ПРОЦЕССОВ
  - на базе ВЦ РАН,зав.кафедрой академик Ю.Г.Евтушенко
- ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ
  - на базе ВЦ РАН, зав. кафедрой член-корр. К.В.Рудаков.
- http://fupm.fizteh.ru/basechairs/

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ ПРОЦЕССОВ И СИСТЕМ

- Специализации:
  - Математическая физика — зав. специализацией Ю.Г. Евтушенко;
  - Теория управления и исследования операций — зав. специализацией Ю.Н. Павловский.
- Специализация «Теория управления и исследование операций»
  - Направления: методы оптимизации; **теория языков программирования; системное программирование**; математическая логика; теория игр; математическая теория макроэкономических процессов; **разработка систем поддержки принятия решений в управлении и планировании**.
  - В проблемно-ориентированной сфере разрабатываются системы распознавания образов (в медицине, геологии и т.д.), системы автоматизации проектирования, интерактивные системы имитации сложных процессов.
  - Учебный план предусматривает знакомство студентов с элементами системного анализа, теорией управления, теорией игр, теорией макроэкономических процессов, избранными главами системного программирования.

МАТЕМАТИЧЕСКИХ ОСНОВ УПРАВЛЕНИЯ

- Нынешние студенты ФУПМ имеют возможность прослушать курсы лекций ведущих российских ученых: д.т.н., профессора А.А. Натана (его основные работы посвящены **статистическим методам обработки данных**, теории и методам статистического распознавания и классификации, стохастическим методам в экономике); д.ф.-м.н., профессора, зав. отделом прикладных проблем оптимизации ВЦ РАН В. Г. Жадана; д.ф.-м.н., профессора зав. сектором комбинаторного анализа ВЦ РАН В.К. Леонтьева; д.ф.-м.н., профессора зав. отделом **систем математического обеспечения** ВЦ РАН, ведущего специалиста по языкам программирования и методам построения трансляторов В.А. Серебрякова; члена-корреспондента РАЕН, профессора, д.ф.-м.н., зав. отделом ВЦ РАН, ведущего специалиста в России в области создания информационно-вычислительных систем и систем автоматизированного проектирования в машиностроении Ю.А. Флерова.
- Направления:
  - фундаментальные **основы информатики** (дискретный анализ, теория и реализация языков программирования, [математическое] моделирование вычислительных систем, операционные системы);
  - **вероятностно-статистическое** (теория вероятностей, математическая статистика, случайные процессы);
  - теория и методы оптимизации (методы оптимизации, методы оптимального управления).

КАФЕДРА НЕЛИНЕЙНЫХ ПРОЦЕССОВ

- **Специализация «Математическая физика»**
  зав. специализацией академик Евтушенко Ю. Г.
  Направления:
  - газодинамические процессы (динамика излучающего газа, динамика плазмы, газовые разряды);
  - процессы в полупроводниковых структурах;
  - процессы в трубопроводах;
  - ударно-волновые течения конденсированных сред;
  - моделирование климата и биосферы
- **Специализация — «Прикладные методы оптимизации»**
  зав. Специализацией Кривцов В.М. **Направления:**
  - теория и численные методы оптимизации, в частности, теория и численные методы решения задач линейного и нелинейного программирования;
  - разработка диалоговых систем оптимизации, решение прикладных задач, требующих использования методов оптимизации

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

- **Специализация «Проектирование и организация систем»**
  зав. специализацией д.т.н., профессор А.И. Эрлих; **Направления:**
  - оптимизация сложных систем;
  - искусственный интеллект;
  - **методы автоматизации управления и проектирования**.
  - Исследования и разработки, выполняемые в рамках данной специализации, направлены на решение задач массового использования современных компьютеров в системах управления, научных исследованиях, проектировании и конструировании новой техники. Одной из научно-технических основ новых информационных технологий — идеи и методы искусственного интеллекта.
  - Благодаря широким международным научным связям всегда доступна новейшая информация о развитии исследований в области искусственного интеллекта за рубежом. Подготовка студентов ориентирована на их дальнейшую работу по развитию новых методов и средств создания сложных интеллектуальных систем различного прикладного назначения.
- **Специализация — «Интеллектуальный анализ данных»**
  зав. специализацией К.В. Рудаков. **Направления:**
  - комбинаторные и алгебраические методы анализа алгоритмов;
  - распознавание образов;
  - математические методы прогнозирования;
  - **data mining**;
  - **прикладные системы распознавания и прогнозирования**.
  - Интеллектуальный анализ данных является одним из наиболее актуальных и востребованных направлений современной прикладной математики. Окончание периода чисто экстенсивного развития вычислительной техники и телекоммуникаций, огромные объемы накопленной информации о самых разных областях человеческой деятельности требуют разработки и применения более изощренных методов анализа данных и подготовки специалистов соответствующей квалификации.