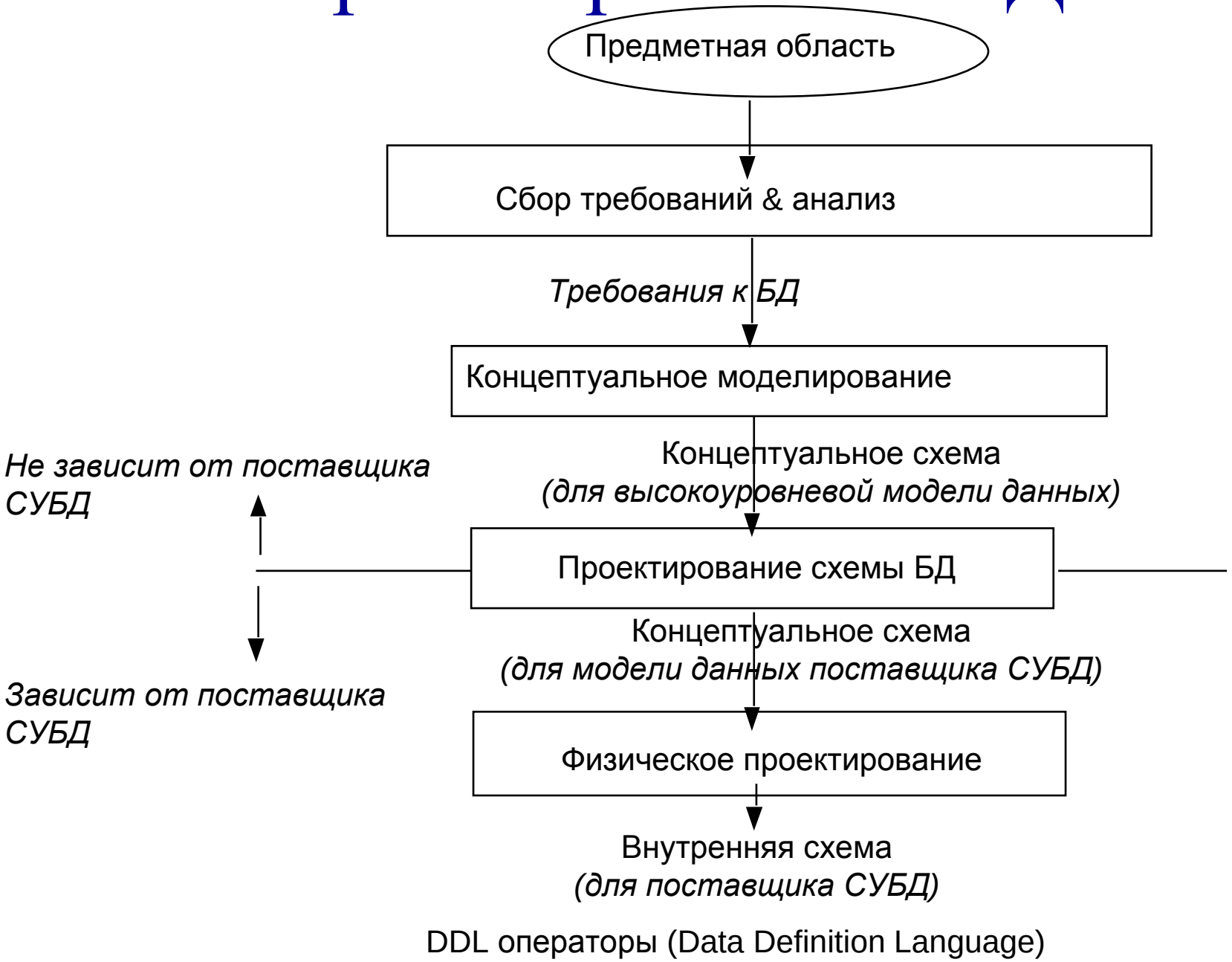
Введение в моделирование данных, базы данных и SQL

Лекции 3, 4, 5

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.
  - По результатам формируем отчет.
- На примере отчета демонстрационного задания **http://bdis.umeta.ru/db/db_course/labs/examp/02.htm**

…db/db_course/labs/examp/02.htm

- Должность выбирается из предустановленного набора должностей, и не зависит ни от сотрудника, ни от отдела. Поэтому "Должность" становится отдельной сущностью, ее атрибуты - граничные значения должностного оклада и требования к уровню и профилю образования (см.ниже) и опыту работы.
- Перечень должностей, которые могут быть в каждом отделе, определяется штатным расписанием. Таким образом, "Штатное расписание" является ассоциативной сущностью, связывающей сущности "Отдел" и "Должность". Строка штатного расписания содержит информацию о количестве ставок одной должности в одном отделе. Таким образом, одной должности может соответствовать несколько строк в "Штатном расписании" (для разных отделов), и одному отделу - тоже несколько строк (для разных должностей).
- Зачисление сотрудника на должность в отделе (на нашей диаграмме - "Работа") является ассоциаций, связывающей не сотрудника с должностью и отделом, а сотрудника - со штатным расписанием, так как зачисление может производиться только в соответствии со штатным расписанием. Одна строка штатного расписания может быть связана с несколькими строками работ, так как ставок по данной строке может быть несколько и зачисление может происходить на дробную ставку (например, 0,5). Один сотрудник может быть зачислен на несколько работ. "Теоретически" общий объем работ, связанных со строкой штатного расписания не должен превышать указанное в штатном расписании количество ставок, но на практике это правило, по-видимому, может нарушаться. Кроме того, опять-таки "теоретически", сумма окладов по работам, связанным (через штатное расписание) с одним отделом, не должна превышать фонда зарплаты отдела, но это правило тоже не абсолютное.
- Еще одна связь "Руководитель" связывает сущности "Отдел" и "Сотрудник".
- Среди запросов, предложенных в задании, есть запрос, подразумевающий наличие информации об образовательном цензе для должности ("инженерная должность") и об образовании сотрудника. Образование характеризуется профилем образования (общее, техническое, экономическое и т.д.) и уровнем (начальное, среднее, высшее и т.д.). Таким образом, "инженерное образование" выражается через комбинацию значений этих атрибутов: ("высшее", "техническое"). Важно, чтобы значения атрибута уровня образования были сравнимы ("высшее">"среднее"). "Уровень образования" и "Профиль образования" выделяются в отдельные сущности-обозначения. Связь этих сущностей с сущностью "Должность" - 1:N, т.е. один и тот же профиль (или уровень) образования может требоваться для нескольких должностей.
- Сотрудник может иметь несколько образований, поэтому вводится еще одна ассоциативная сущность "Образование". Атрибут этой сущности "Год получения" соотносится с атрибутом "Опыт" сущности "Должность". Предполагается, что число лет, прошедших с года получения образования и составляет опыт сотрудника. "Теоретически" при назначении сотрудника на должность его уровень и опыт должны быть не меньше уровня и опыта, требуемых для этой должности, а профиль - точно соответствовать требованиям должностям, однако, нарушения этого правила возможны.

Часть требований предметной области

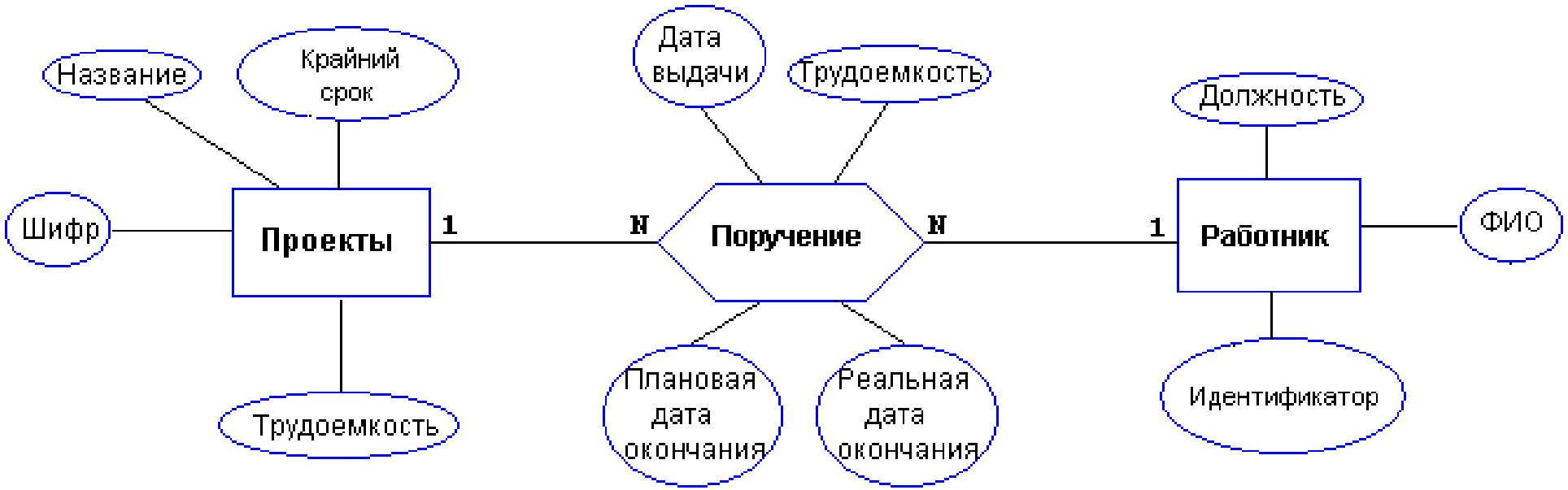
- предустановленный набор должностей
- перечень должностей отдела определяется штатным расписанием
  - строка штатного расписания содержит информацию о количестве ставок одной должности в одном отделе
- зачисление сотрудника может производиться только в соответствии со штатным расписанием.
- один сотрудник 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



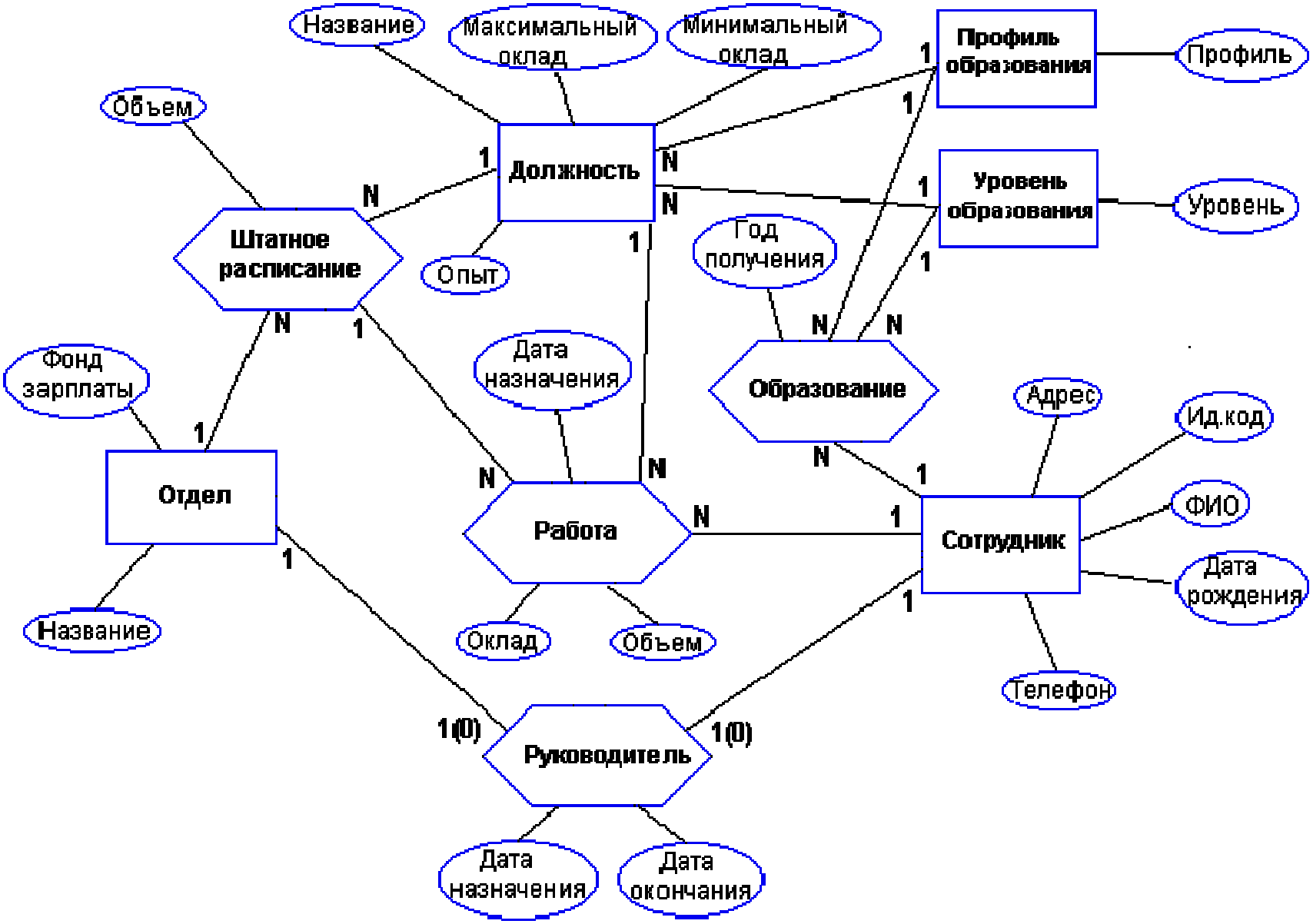
- В индивидуальных заданиях указаны две **сущности** и одна **ассоциация/связь** – это основа для сбора требований к реализации БД.
  - В результате осмысления предметной области концептуальную схему **следует довести хотя бы** до 4-х сущностей и 3-х ассоциаций.
- Например, видим,
  - что **Работник** есть ассоциация двух сущностей – **Персоны** (есть *ФИО* можно добавить день и место рождения, паспорт, прописку …) и **Сотрудника** (*Должность* в каком-то Подразделении в соответствии со Штатным Расписанием какой-то Организации …).
  - Могут ли быть учреждения без руководства?
    - У Подразделения есть Руководитель (со статусом: «уже» или «еще нет» - «исполняющий обязанности» - И.О.)
    - У Организации - целое Руководство, включающее Директора, его замов, секретарей и т.п.

- **Что есть** Штатное Расписание?
  - **Штатное Расписание -** это какие должности допустимы, для какого подразделения, с какой квалификацией должен быть сотрудник …?
- Далее можем предположить, что **Персона** имеет определенные квалификацию и Образование, Трудовую Книжку, поощрения и взыскания, пребывание в Отпуске, Командировках.
- А Контактная информация? Как связаться с Персоной, Подразделением, Организацией?
  - Вне работы. По месту работы. По месту реализации проекта…
- Получается, что вокруг **Сотрудника** у нас возникает даже отдельная БД
  - «БД отдела кадров».

- Аналогично с **Проектом**.
  - Непросто название проекта и дата его завершения.
  - Проект – это последовательность решаемых Задач, распределенных по времени (Вехам) – плановому и фактическому.
- В реализации Проекта, его Задач, участвует много Сотрудников в разных Ролях:
  - Менеджер, Аналитик, Проектировщик, Разработчик, Тестировщик, Документатор, Внедренец …
    - со своими характеристиками.
- Для решения Задач требуются не только людские Ресурсы, но и
  - финансовые (Зарплаты, Премии и Взыскания),
  - организационные (Роли, Методики, Среды и Системы),
  - инфраструктурные (Помещения, Оборудование).
  - …

- **В результате** получим приближение к концептуальной схеме «Примера выполнения лабораторных работ».
- **Следует обратить пристальное внимание**
  - на **мощности** связей
    - Кто, что определяет? В какой пропорции?
  - «**перечисления** наименований» - словари допустимых значений – LOV (List Of Values)
    - Перечислимый тип, который можно реализовать
      - Нерасширяемым образом
        - » домен, CHECK (SQL DDL CREATE TABLE)
        - » чтобы изменить, требуется менять схему БД
      - Расширяемым образом
        - » Таблица – словарь допустимых значений
        - » Использующий ссылается на словарь с помощью FK

Лаб.2 - концептуальная схема данных - 2



Этапы построения приложения БД-3

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

- **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
- …

Лабораторные работы

- Создание системы баз данных
  - **2**. Проектирование схемы (Концептуальное моделирование)
  - **3**. Создание таблиц (Логическое моделирование)
  - Нормализация БД
- Наполнение системы баз данных, обработка и подготовка выборок данных
  - **1**. Работа в среде интерактивного SQL
  - **4**. Манипулирование данными
  - **5**. Создание и использование представлений
- Обеспечение многопользовательской работы системы баз данных
  - **6**. Управление транзакциями
  - **7**. Управление доступом
- **Физическое** моделирование системы баз данных
  - **8**. Выборка метаданных
  - **9**. Создание и использование триггеров
- Оптимизация системы баз данных
  - Денормализация БД
  - **10**. Использование индексов и средств оптимизации запросов
  - Резервное копирование и восстановление данных

Лабы – это этапы **проекта** создания Системы БД

- Этап 1: анализ предметной области приложения
- Этап 2: концептуальное моделирование
- Этап 3: отображаем концептуальную схему предметной области в реляционную схему
- Этап 4: реализуем РБД, используя язык РСУБД SQL DDL
- Этап 5: манипулируем РБД, используя язык РСУБД SQL DML
- …
- Чтобы успешно сдать лабы,
  - следует последовательно вести работы(готовить и сдавать лабы),
  - документировать ведение работ

Документирование работ

- Техническое Задание (ТЗ) – постановка задачи по реализации Системы Базы Данных
  - **Что нужно сделать?**
  - Система Базы Данных = СУБД + наша БД
    - Добавим программную реализацию
      - логики управления данными, их использования,
      - визуальный интерфейс для использования и части управления
    - получим Информационную Систему
- Технорабочий проект (ТРП)
  - Как можно сделать?
  - Как сделали?
  - Почему? Зачем?
  - Насколько правильно, точно?
  - Как настроить?
  - Как пользоваться? …

Техническое Задание

- Постановка задач проекта по реализации Системы Базы Данных
  - Что есть предметная область проекта?
    - С чем и зачем мы собираемся иметь дело?
  - Сбор и анализ требований.
    - Что представляют собой объекты предметной области?
    - Какие у них характеристики/атрибуты?
    - Как они связаны?
    - Какие ограничения характерны объектам и их связям?
  - Построение концептуальной (инфологической) модели
    - Спецификация концептуальной модели
      - «рисование» диаграммы с объектами, их связями
  - Описание, перечисление ключевых операций управления и использования данных (вербально).
    - Чем, как наполним БД?
    - Что, как часто, почему будем выбирать, менять?
    - Что, как следует показывать пользователям?
    - Какова должна быть производительность, как добиться требуемой эффективности?

Технорабочий проект

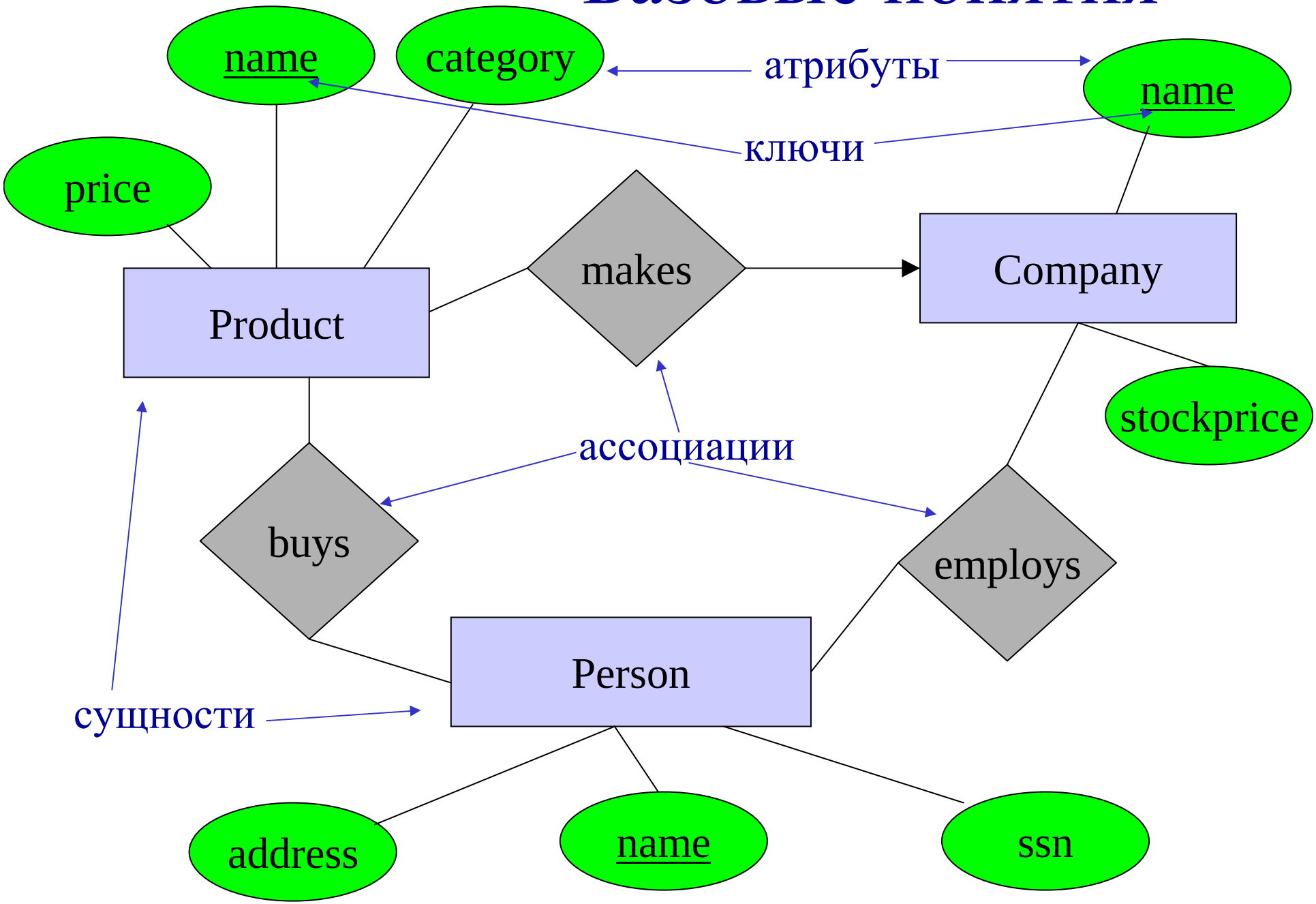
- [Дата-]логическое моделирование
- Отображение концептуальной схемы в реляционную схему
  - Соответствующая(ие) ER-диаграмма(ы) БД
  - Соответствующие скрипты/спецификации
    - создания таблиц БД на SQL DDL
    - ограничений целостности
    - представлений (view) …
- Манипулирование БД, используя язык SQL DML
  - какие команды наполнения БД,
  - как продемонстрировать содержимое БД, как извлечь,
  - какие типовые команды изменения наполнения БД…
- Оценка производительности, обеспечение эффективности …
  - Денормализация, индексы, оптимизация
  - Управление доступом

Обзор концептуального проектирования

Обзор концептуального моделирования

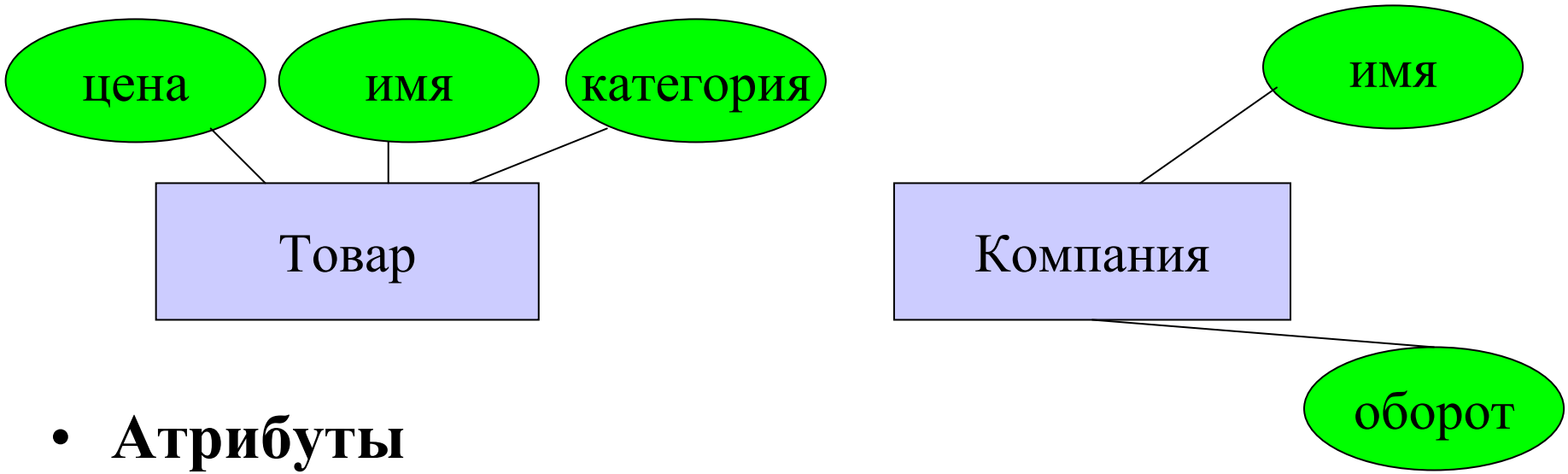
- Инфологическая модель данных "Сущность-связь"
- Основные понятия языка моделирования (графического, концептуального)
  - **Сущность** – любой **различимый** объект, существующий **сам по себе**, **отличимый** от других объектов
  - **Атрибут** – поименованная **характеристика** сущности.
  - **Ключ** – **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



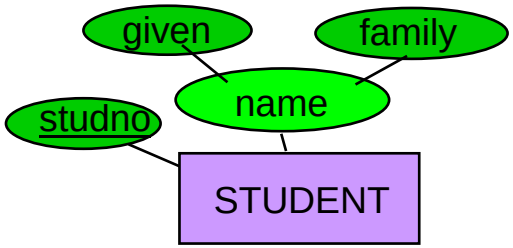
- **Атрибуты**
  - каждый имеет значения **элементарных** типов данных: строка, целые, вещественные, время/даты и т.п.
- **Множество сущностей**: коллекция аналогичных сущностей

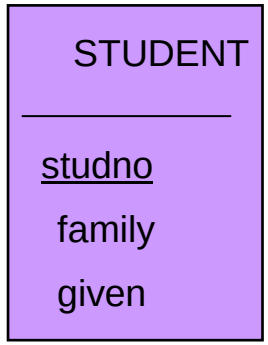
Сущность

- **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.

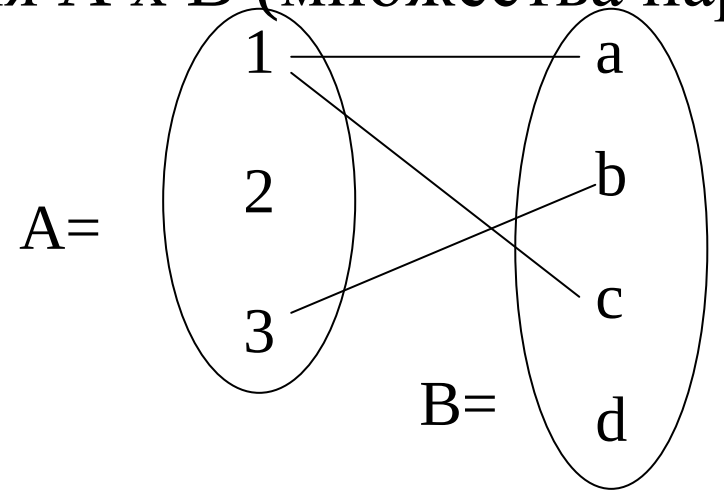




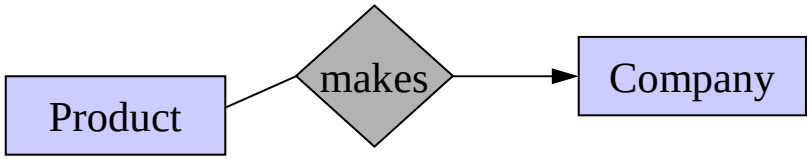
Ассоциации/отношения

- 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)}

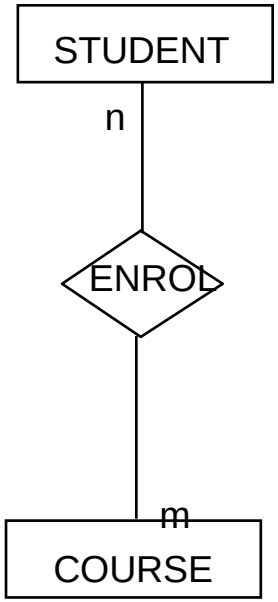


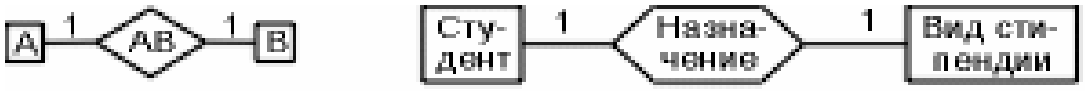
Подмножество **Product** x **Company** выражается:



Ассоциации

- ассоциация, устанавливаемая между двумя или более сущностями.
  - *бинарная связь*
- обеспечение возможности отыскания одних сущностей по значениям других
- изображаются





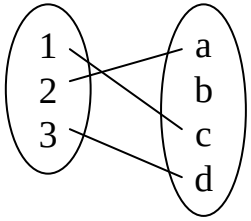
  - помеченными ромбами или шестиугольниками
- [могут иметь атрибуты]

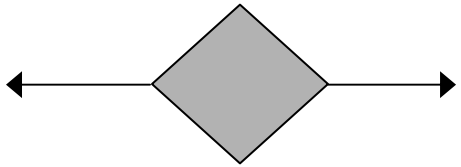
Связь

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

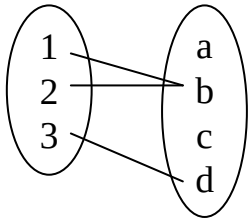
Множественность ER отношений/связей

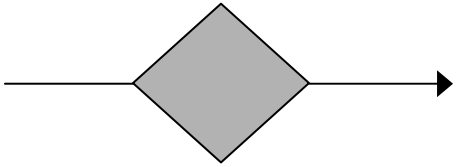
- один-к- одному:



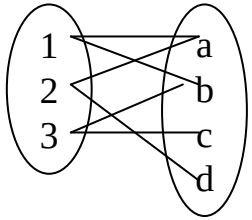


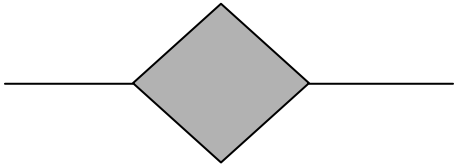
- многие-к-одному

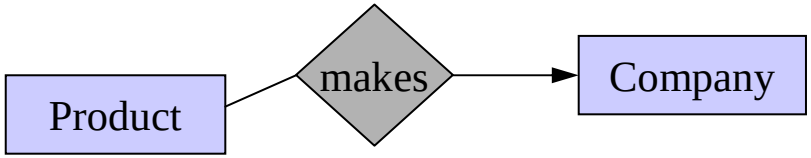




- многие-ко-многим





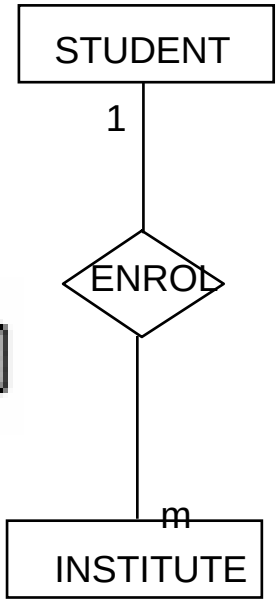


Связь **один-к-одному** (1:1)

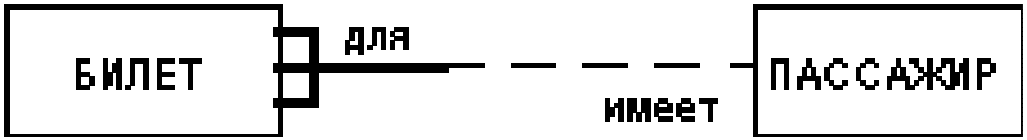
- связи может участвовать только один экземпляр
- каждому экземпляру A соответствует 1 [или 0] представителей B
- одноточечный вход
  - конец связи
    - обязательный
      - изображается сплошной линией,
      - указывается степень - 1
    - необязательный
      - изображается прерывистой линией.
 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

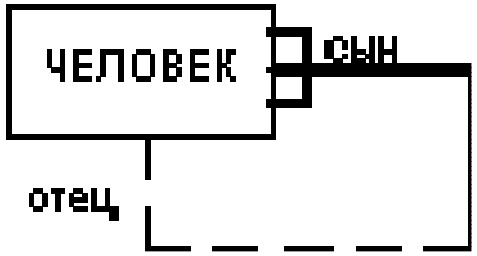
Связь **один-ко-многим** (1:M)

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

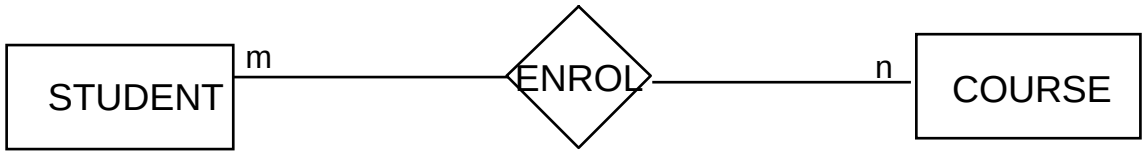


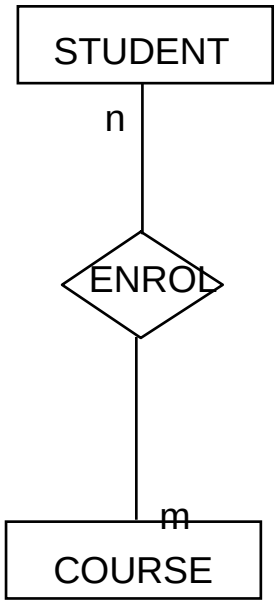
- *используются трехточечный вход в прямоугольник сущности*





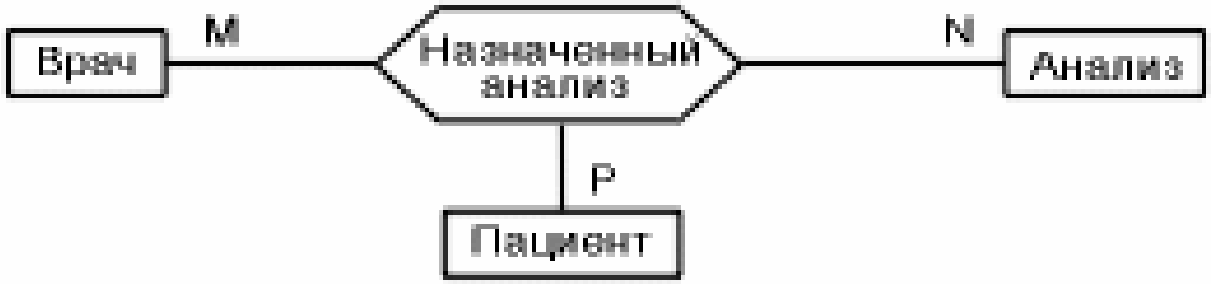
Связь **многие-ко-многим** (M :M)





- множество связей между одними и теми же сущностями

- тернарные связи



Многообразие связей

моногамный брак

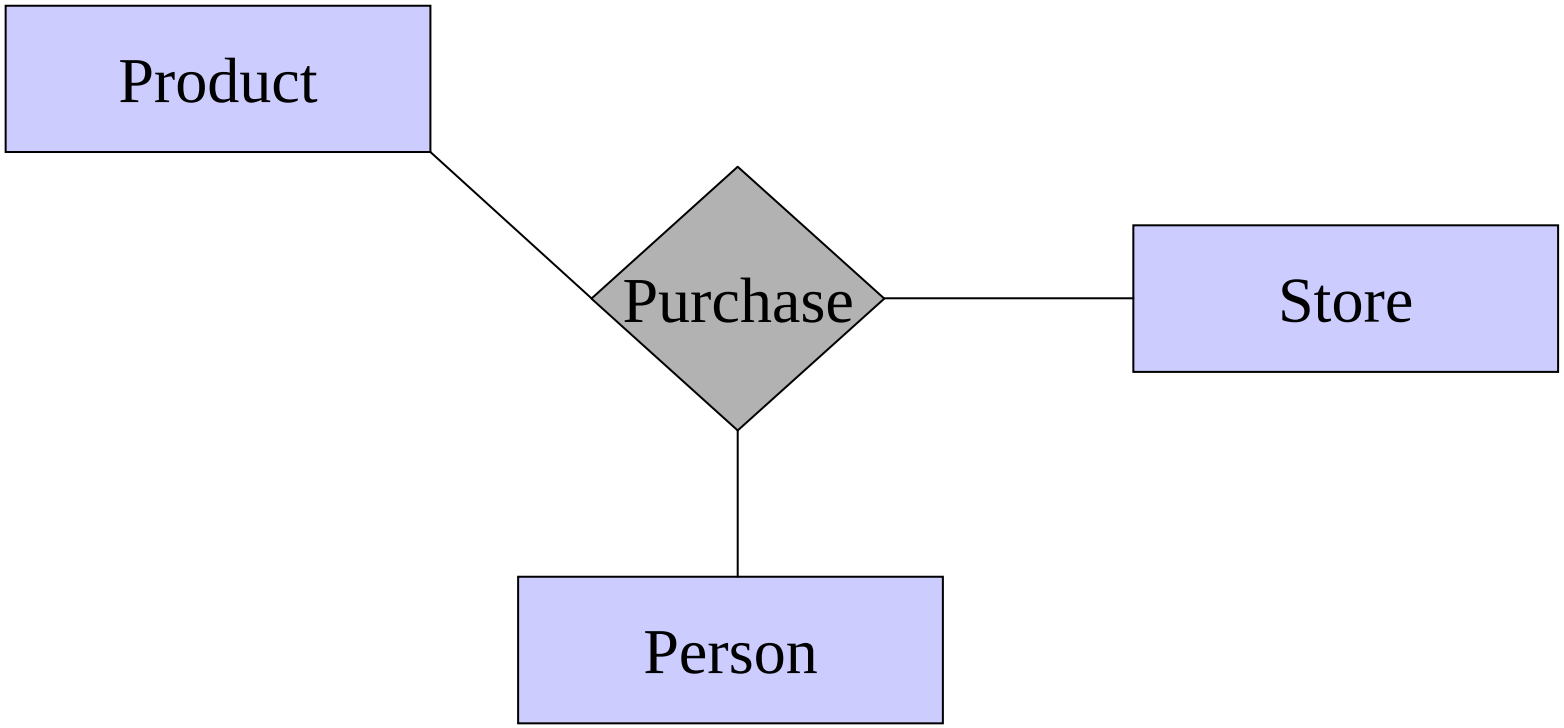
многоженство

многомужие

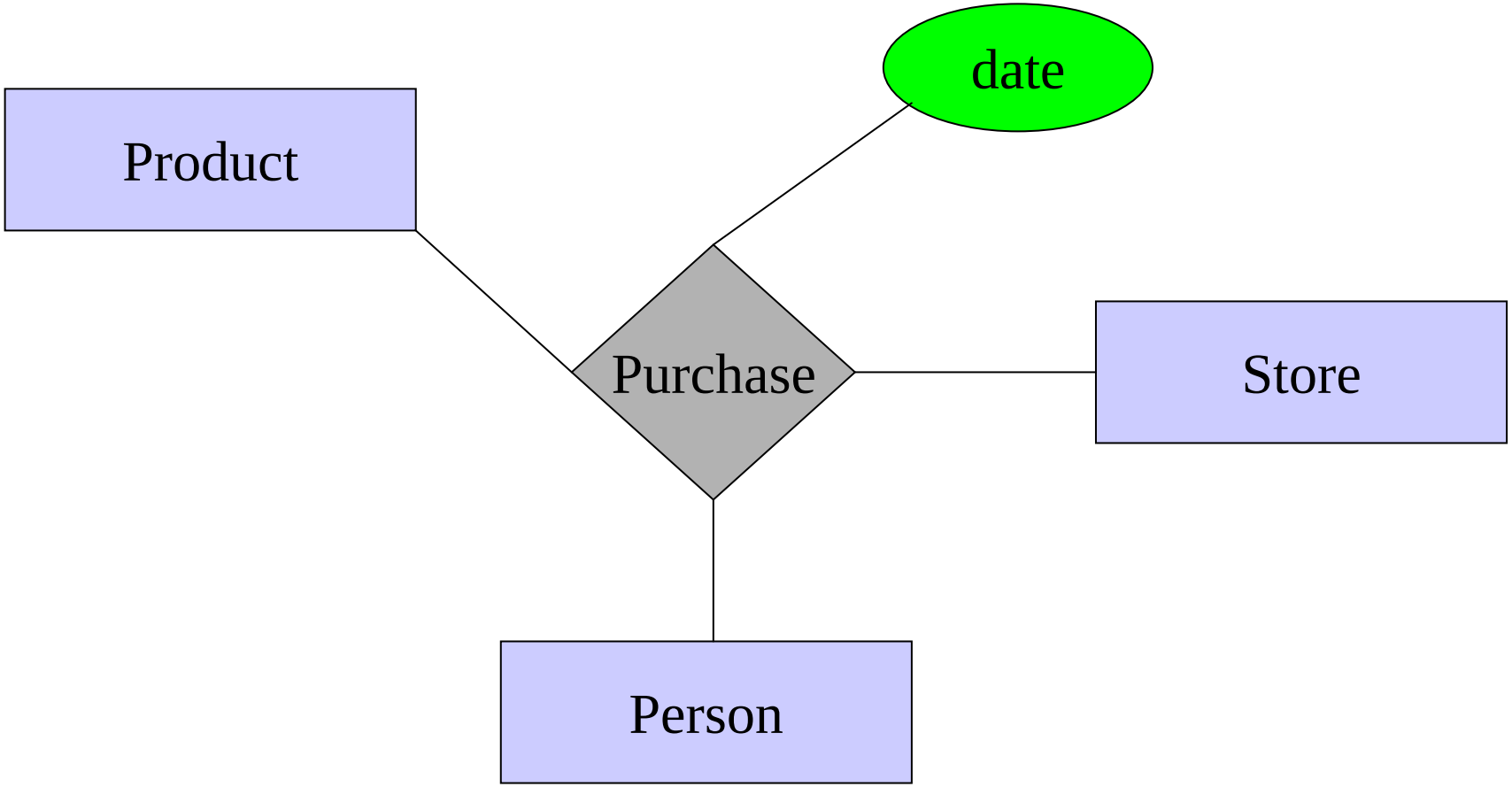
полигамный брак

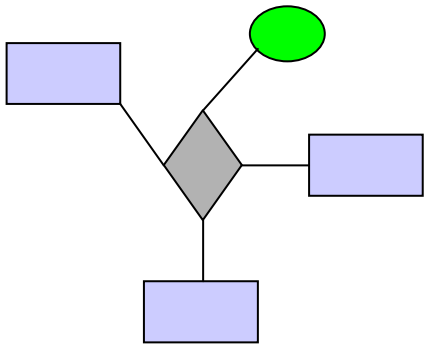
Многосторонние связи

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

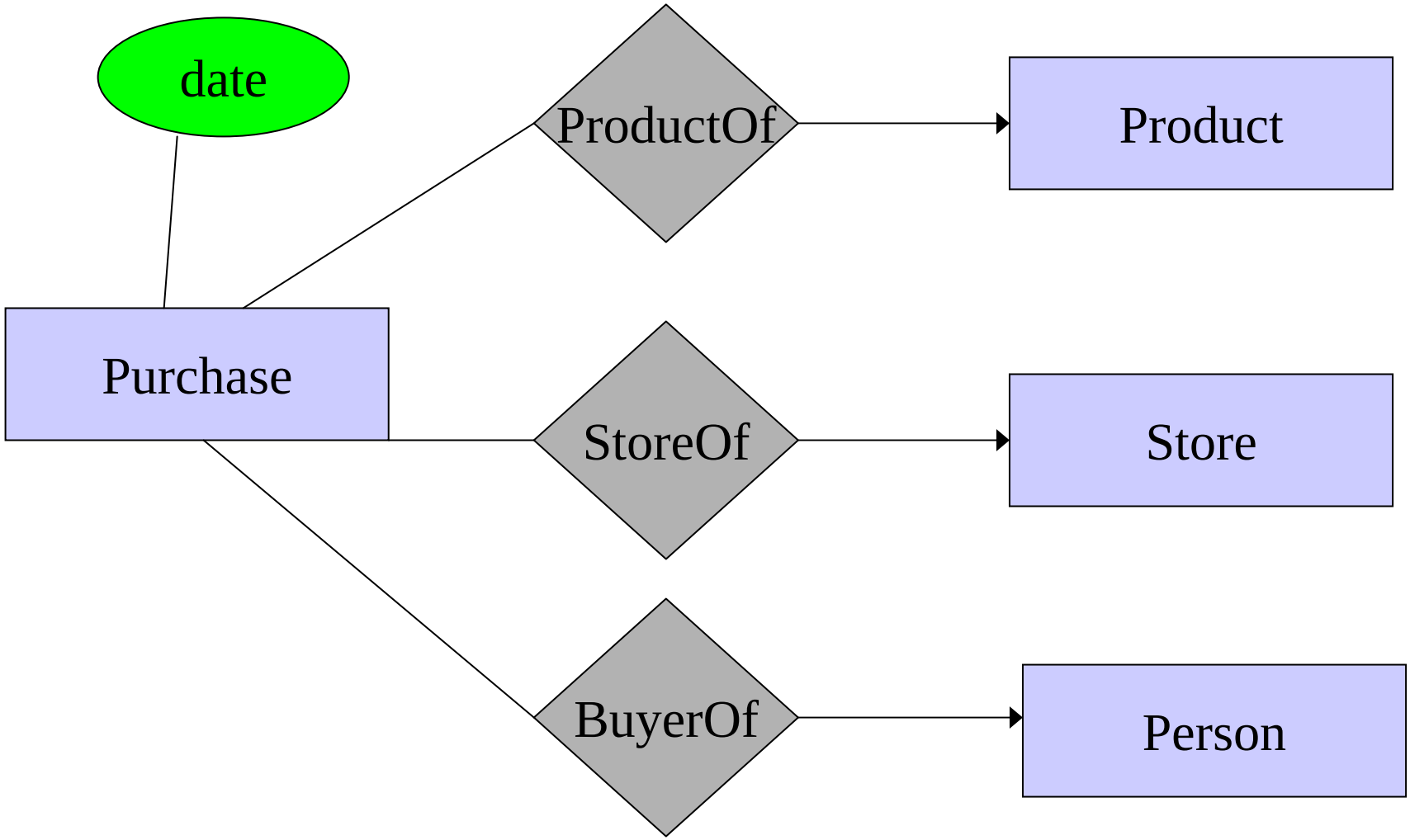


Атрибуты связи





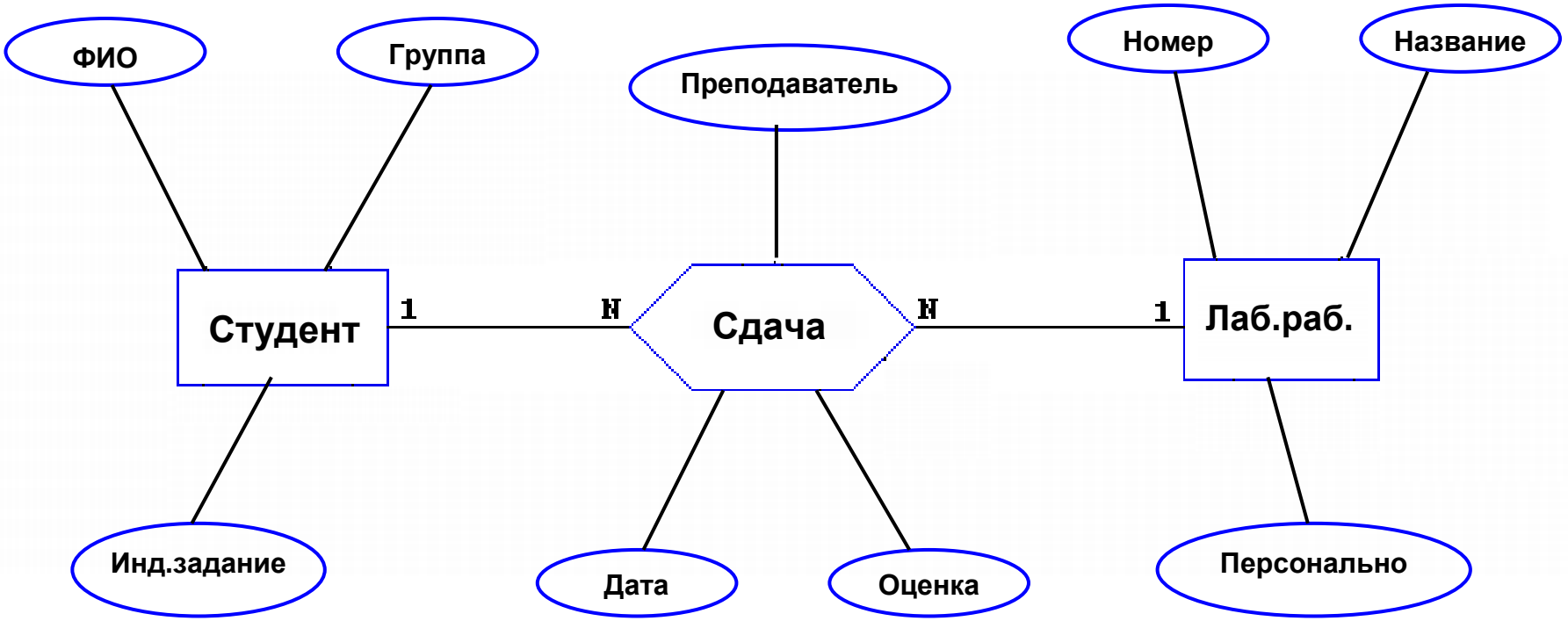
Преобразование n-арных связей в бинарные



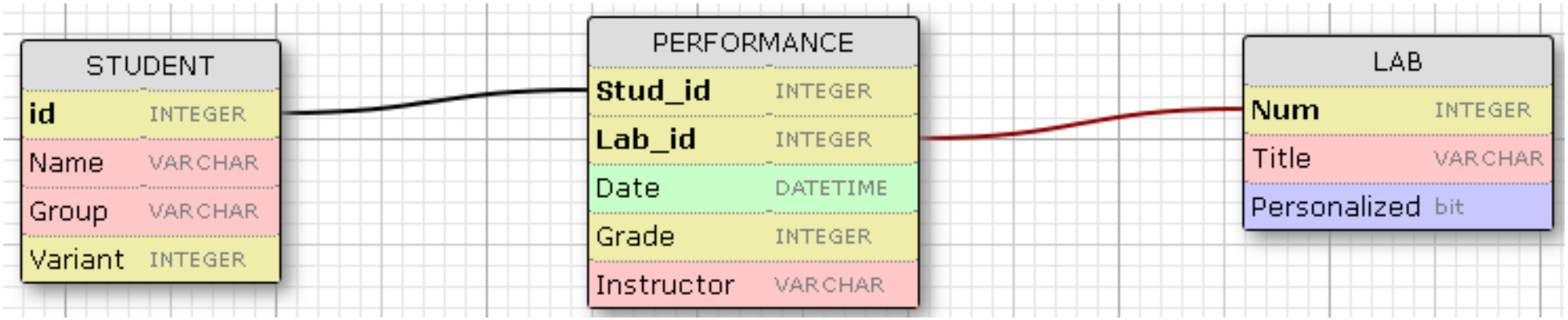
Связи: резюме

- 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

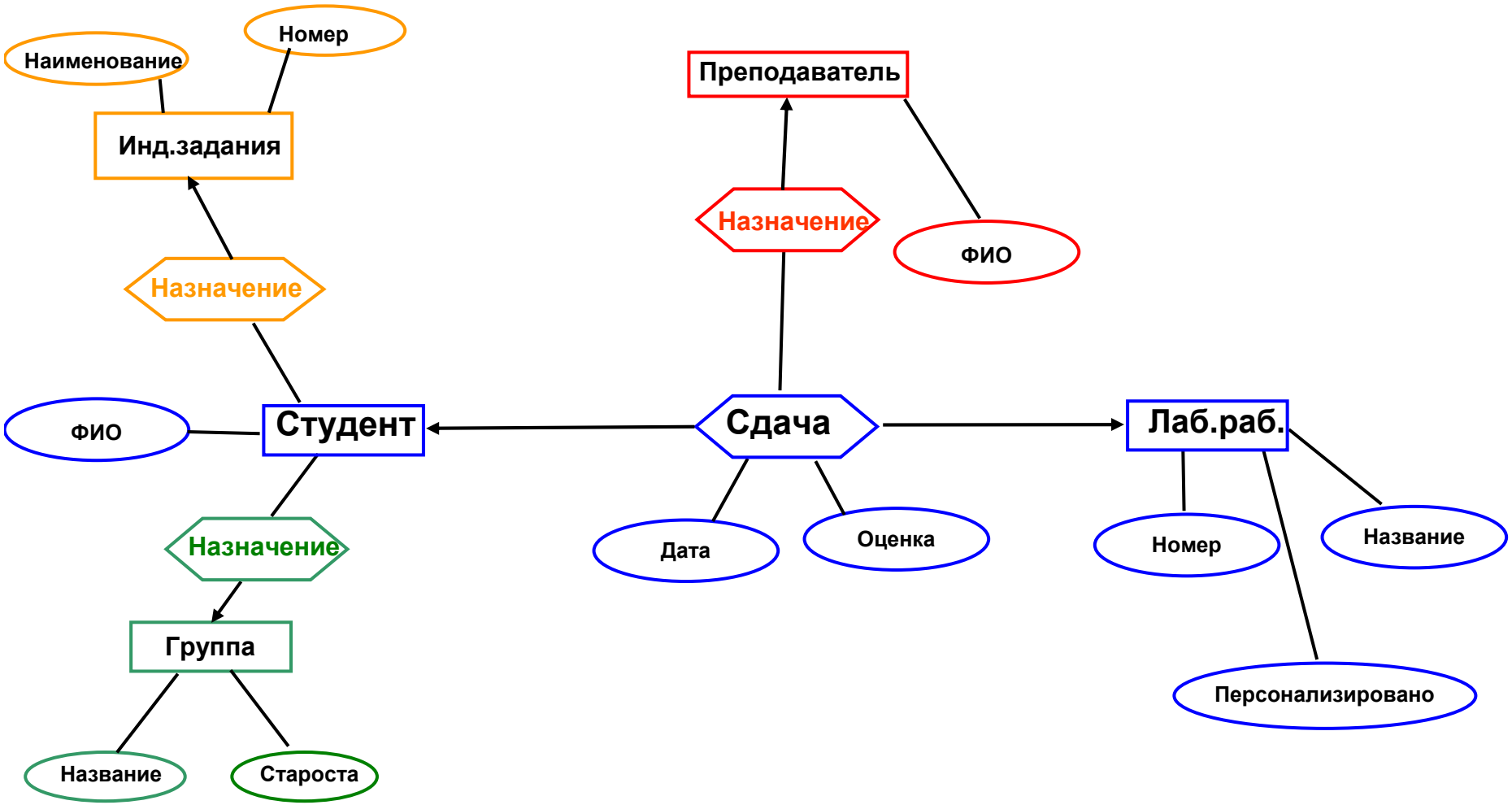
ER-модель задания-1



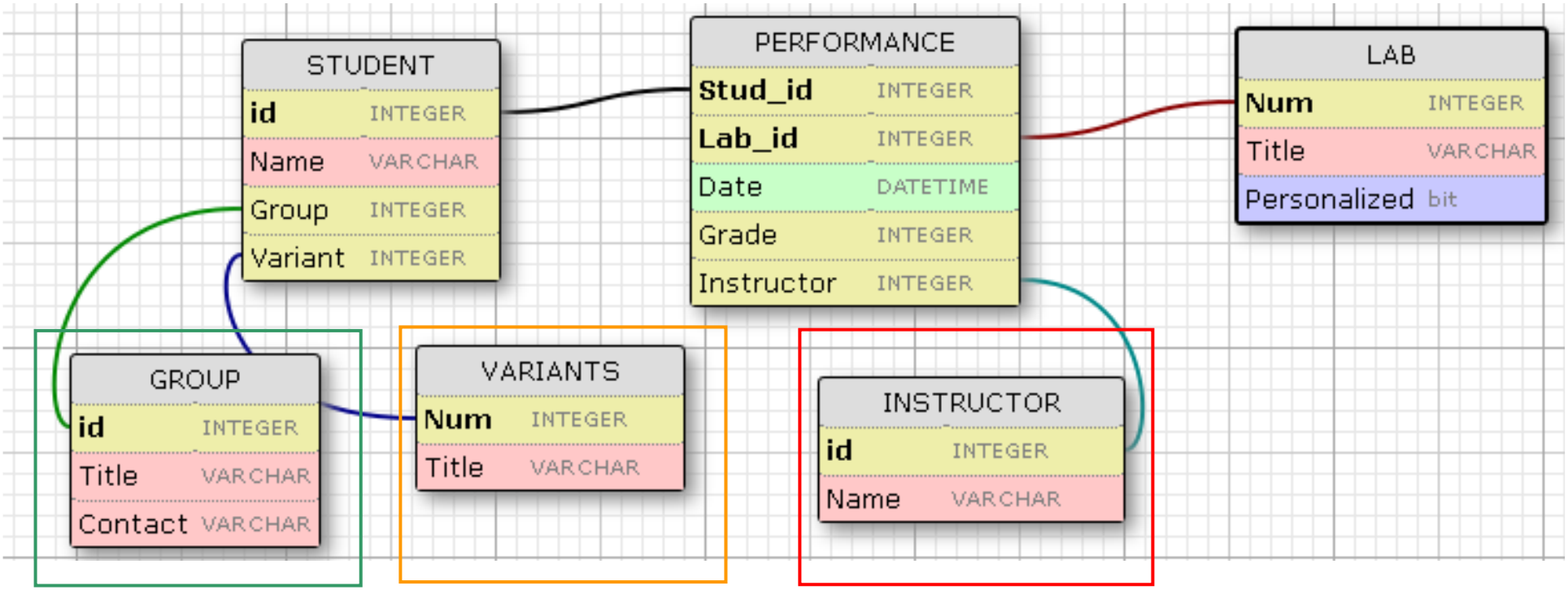
Реляционная модель – 2 балла



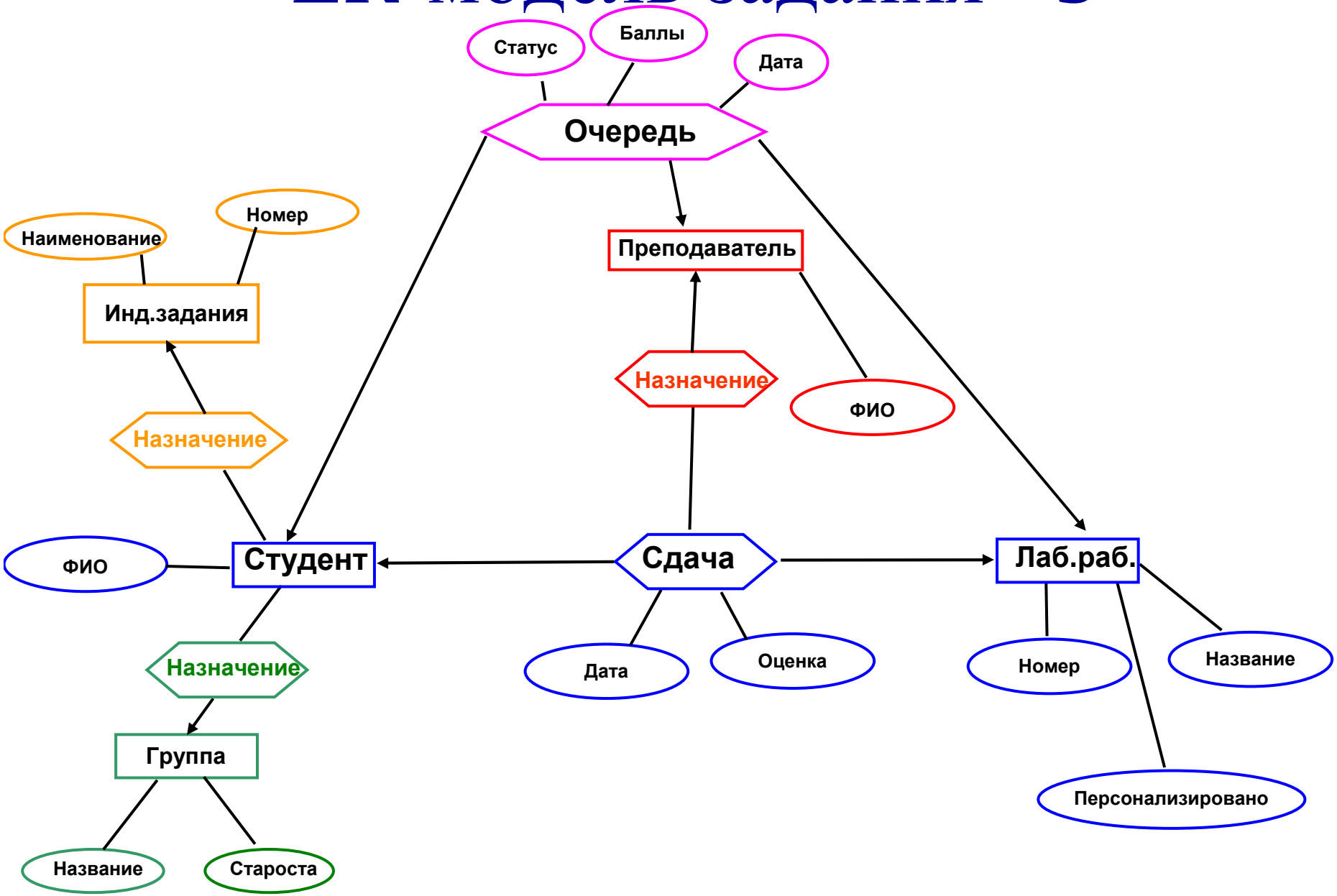
ER-модель задания-2



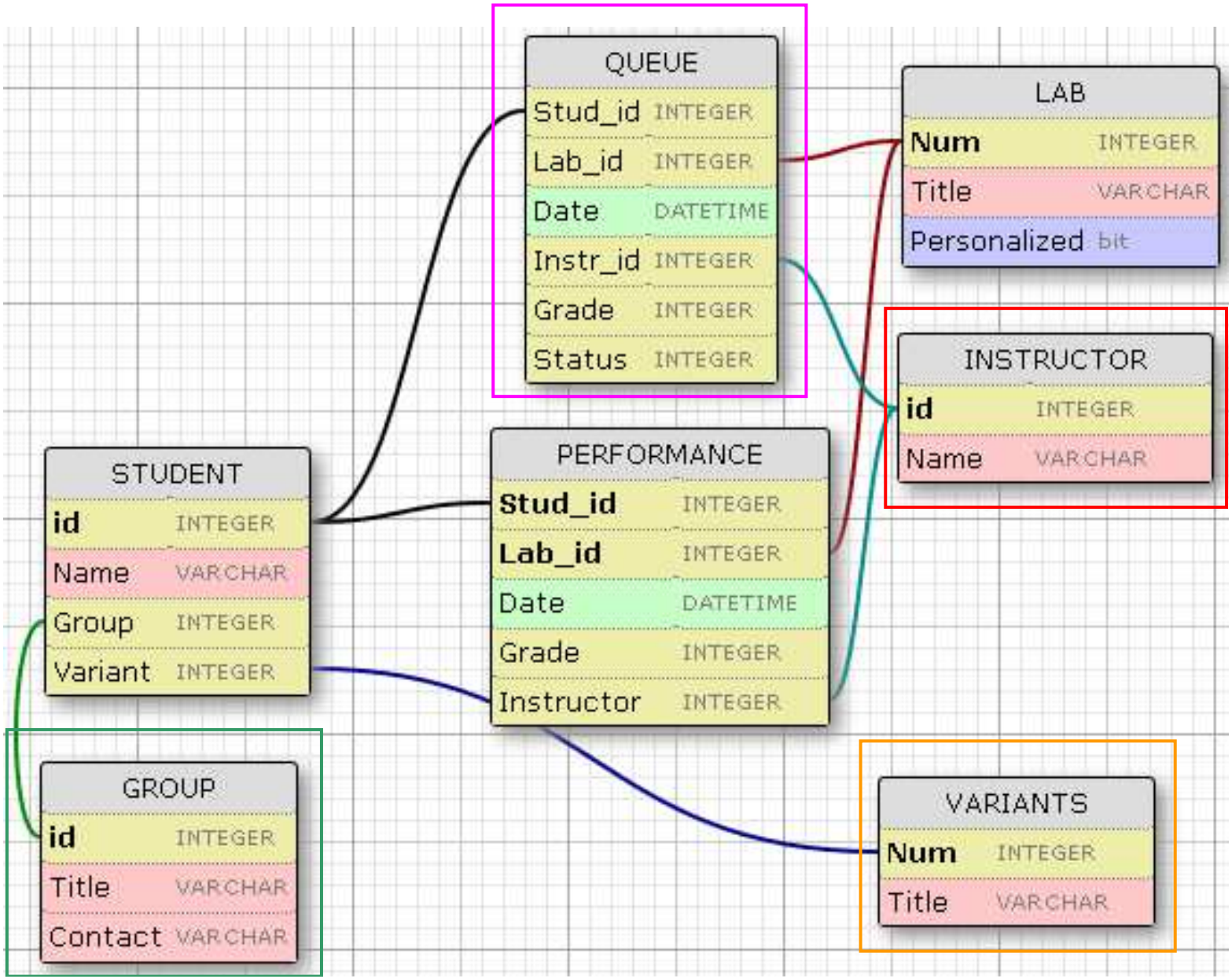
Реляционная модель – 4 балла



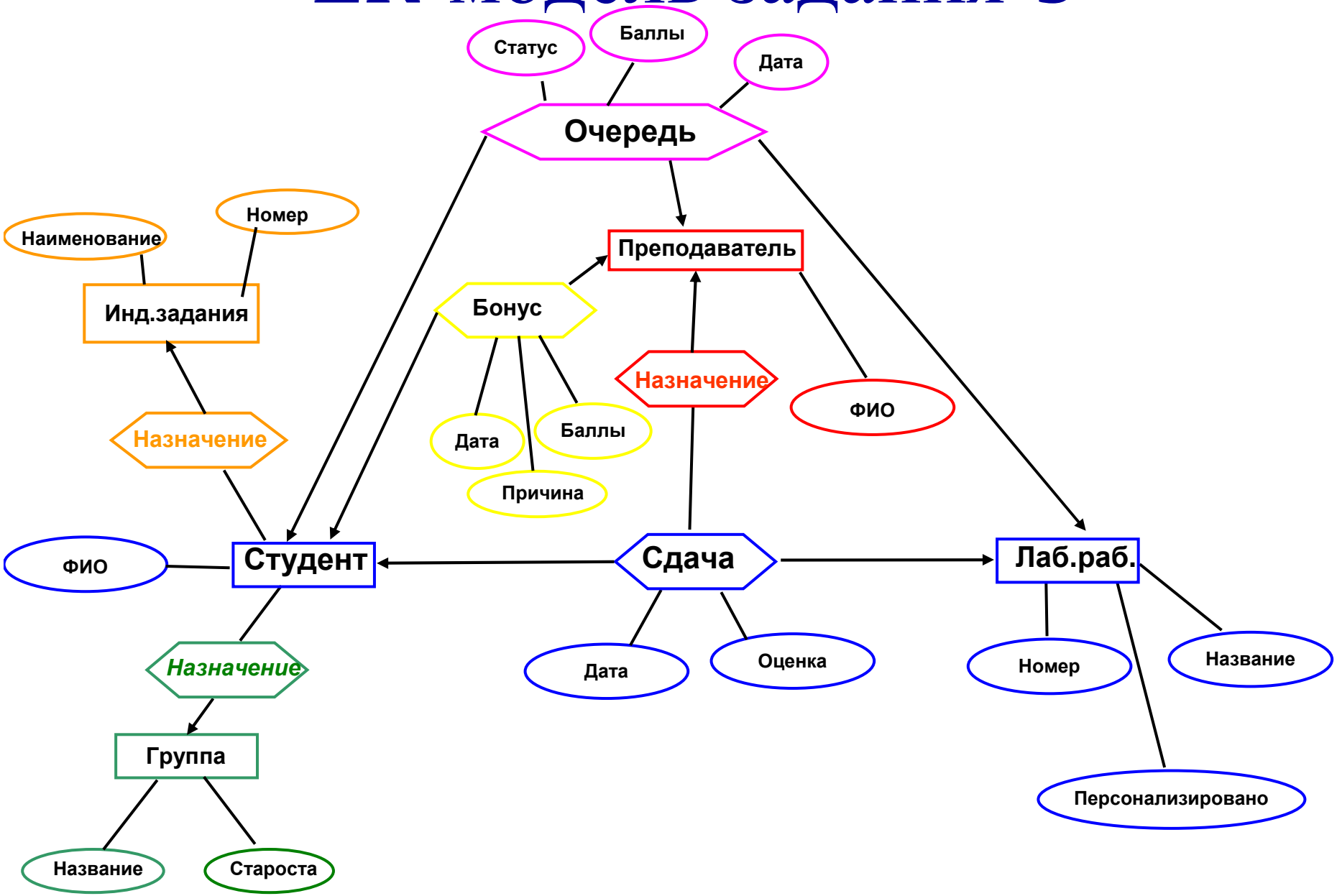
ER-модель задания - 3



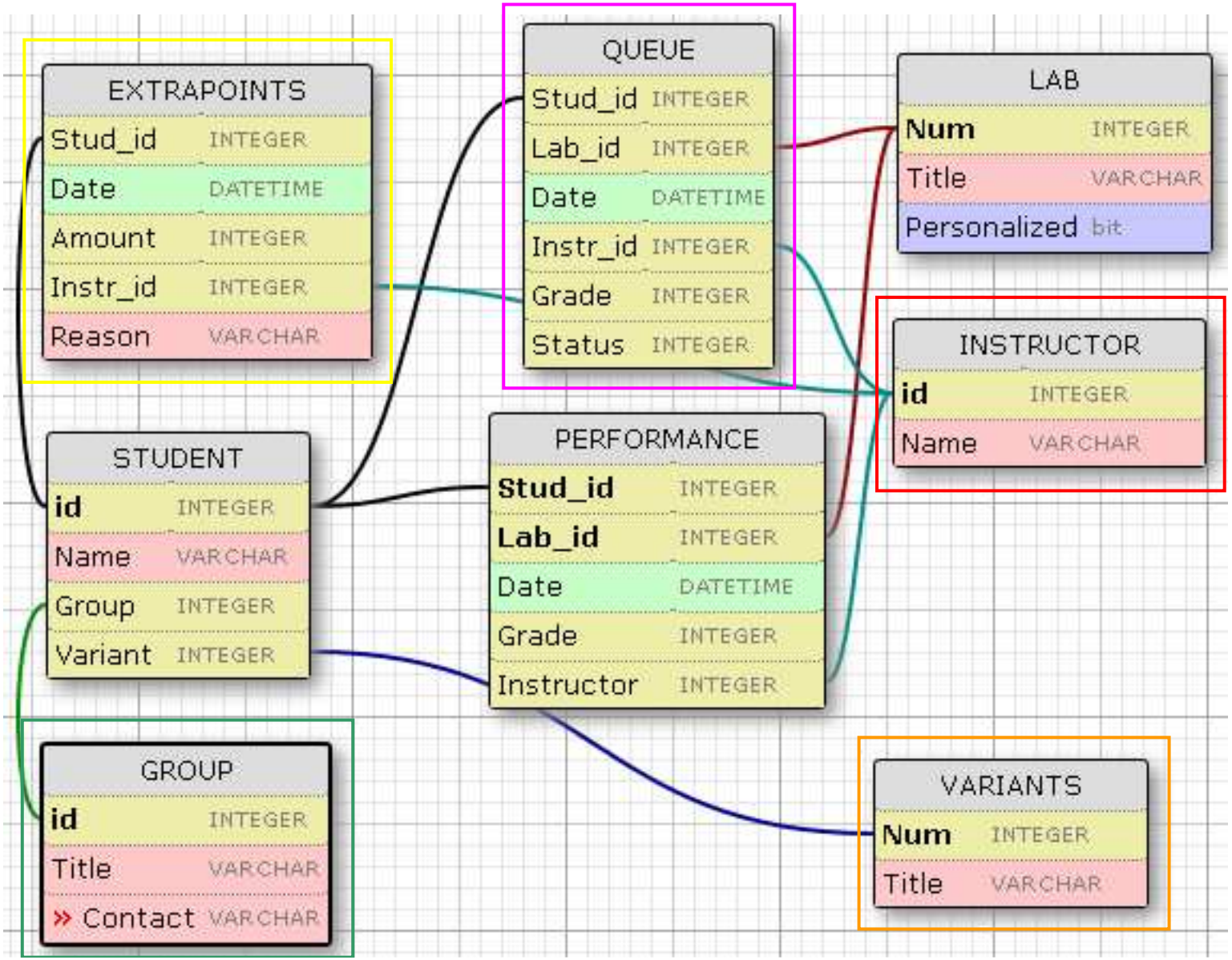
Реляционная модель – 6 баллов



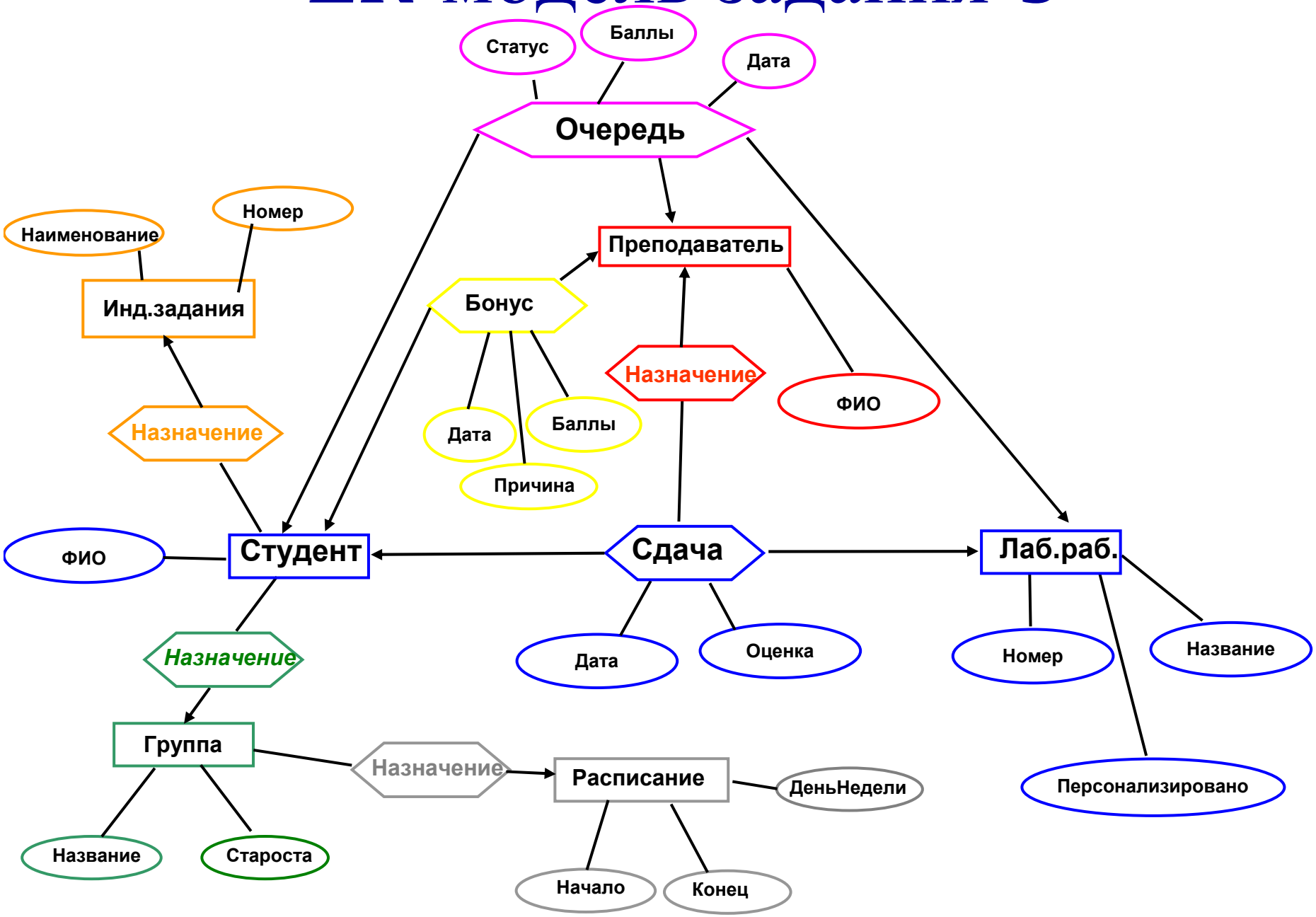
ER-модель задания-5



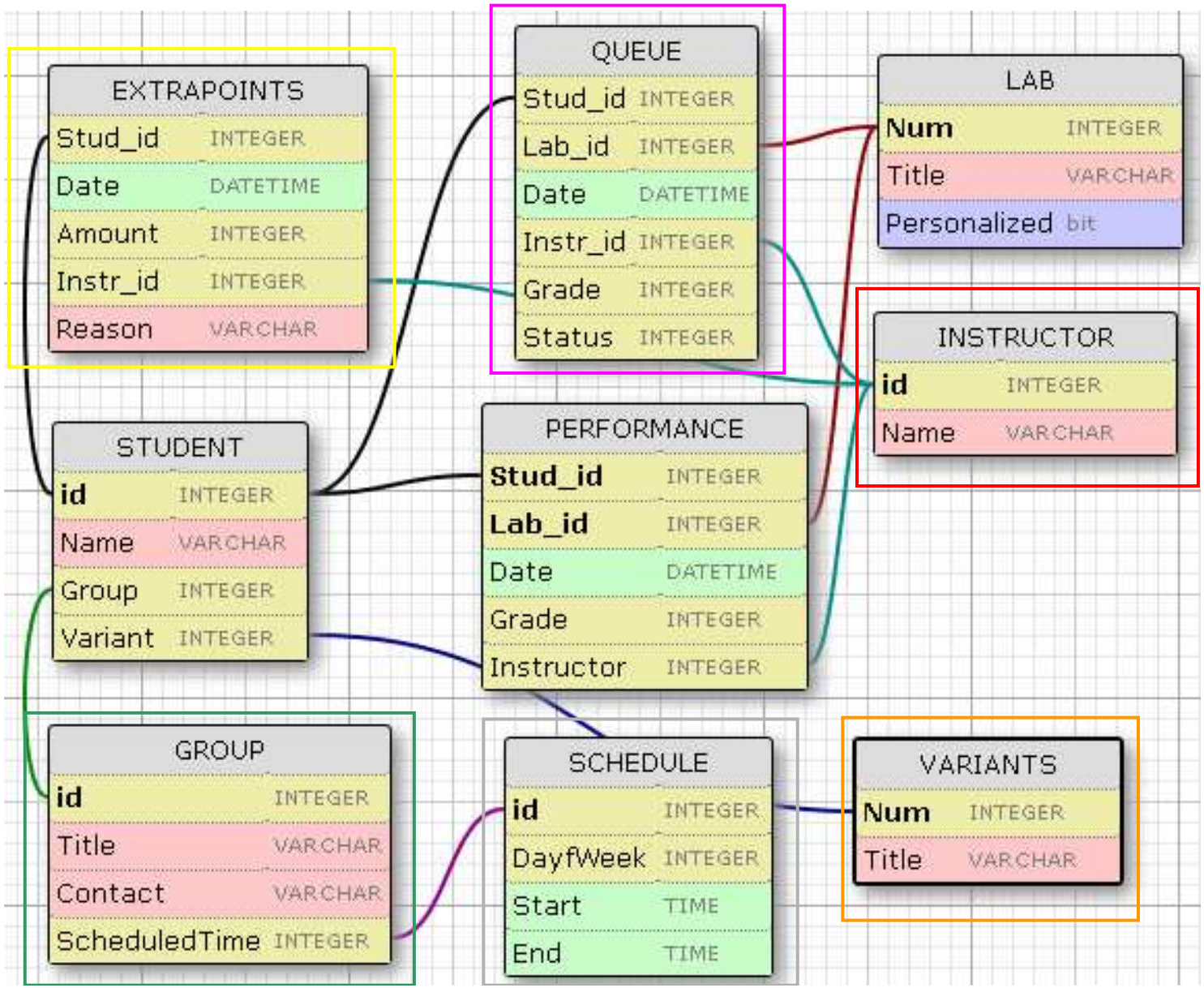
Реляционная модель – 4



ER-модель задания-5



Реляционная модель – 6 баллов



Реляционная модель – 8 баллов

Модель, готовая к эксплуатации в реальных условиях:

- предпочтения преподавателей
  - в отношении лабораторных работ
  - в отношении тем лаб. работы № 2
- поддержка процесса сдачи вне расписания (на ВЦ)
- поддержка индивидуальных графиков сдач

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

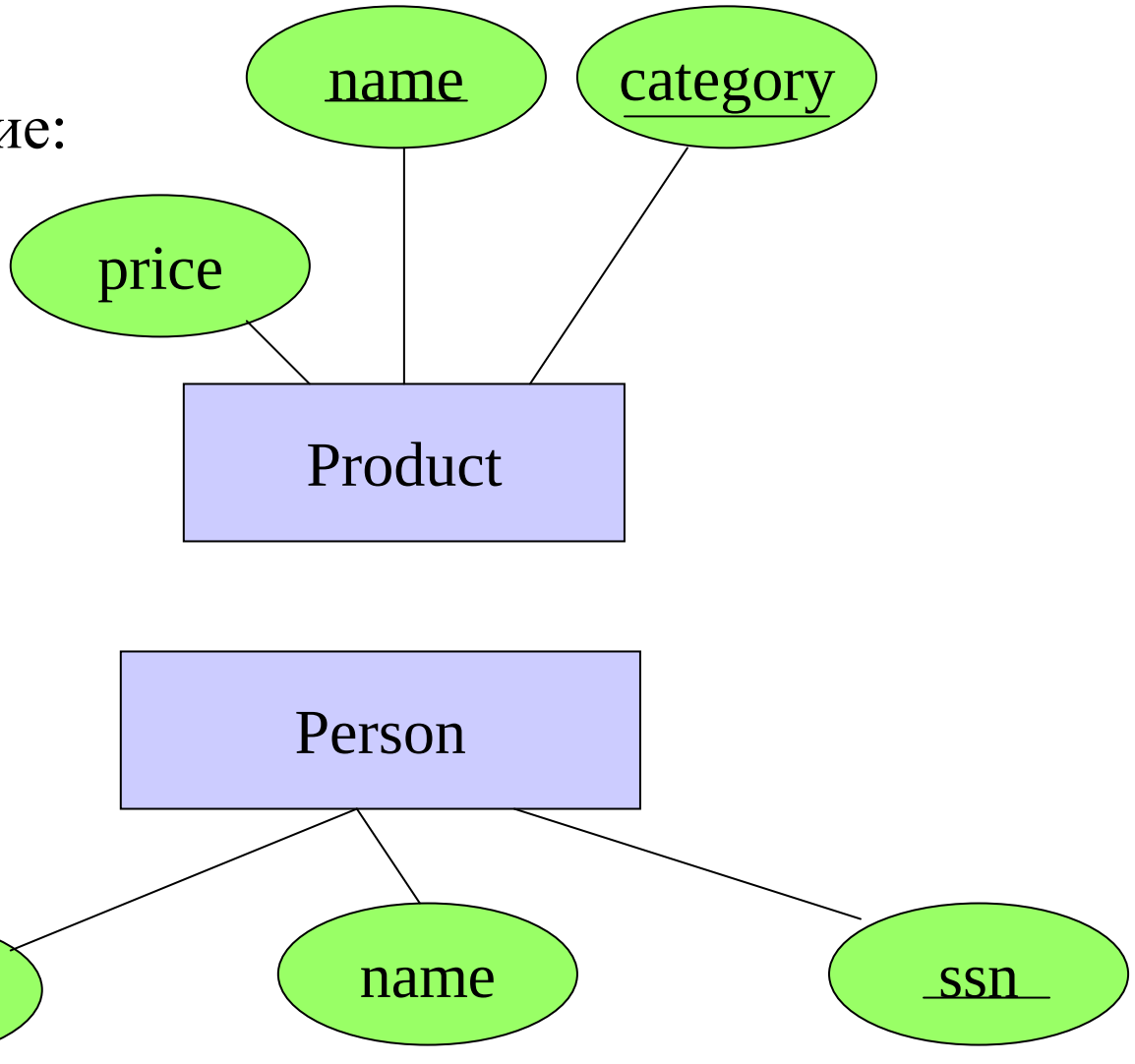
средняя цена пива не должна превышать 150

Ключи

- 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

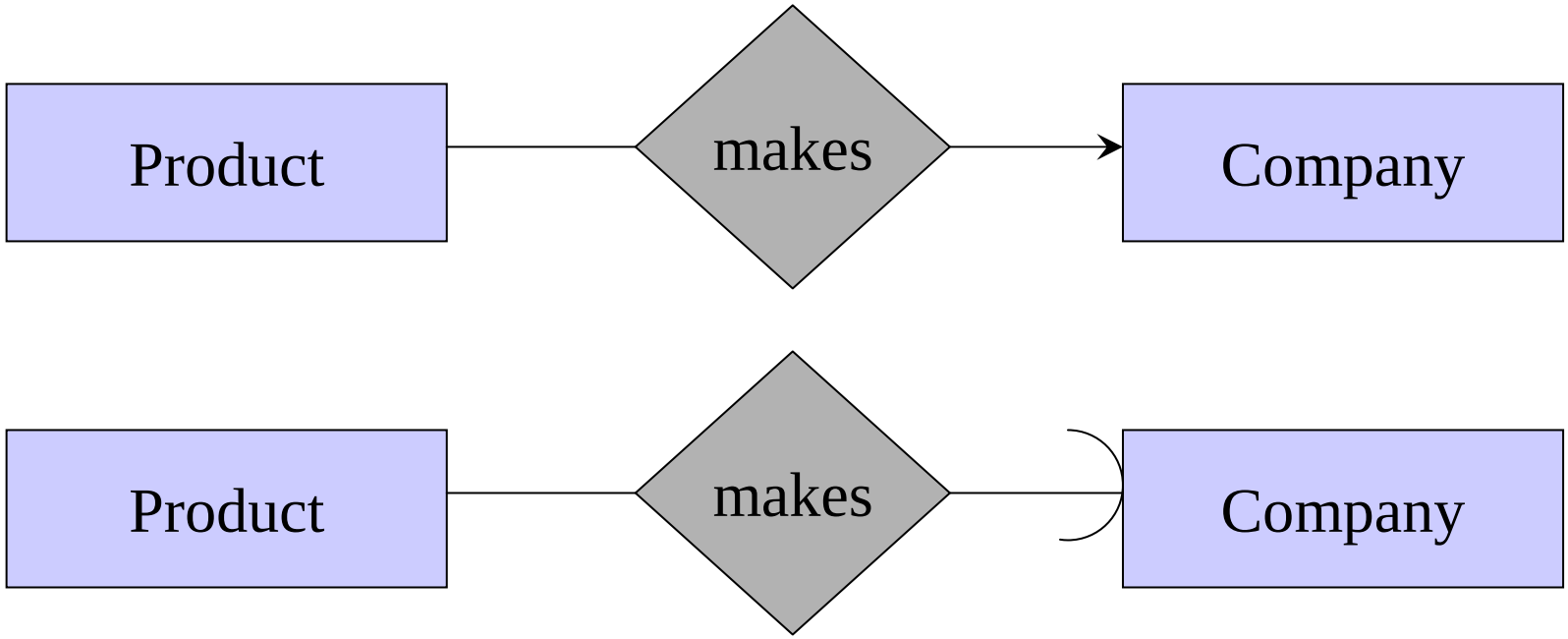
Ограничения ссылочной целостности

- **Ограничения на отдельные значения**
  - не более чем одно значение имеется в данной роли
- **Ограничения ссылочной целостности**:
  - ровно одно значение имеется в данной роли
  - явно требуют, чтобы ссылка существовала – указывала на существующий объект
- Если атрибут имеет обязательное значение
  - то это можно рассматривать как вид «ограничения ссылочной целостности».

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

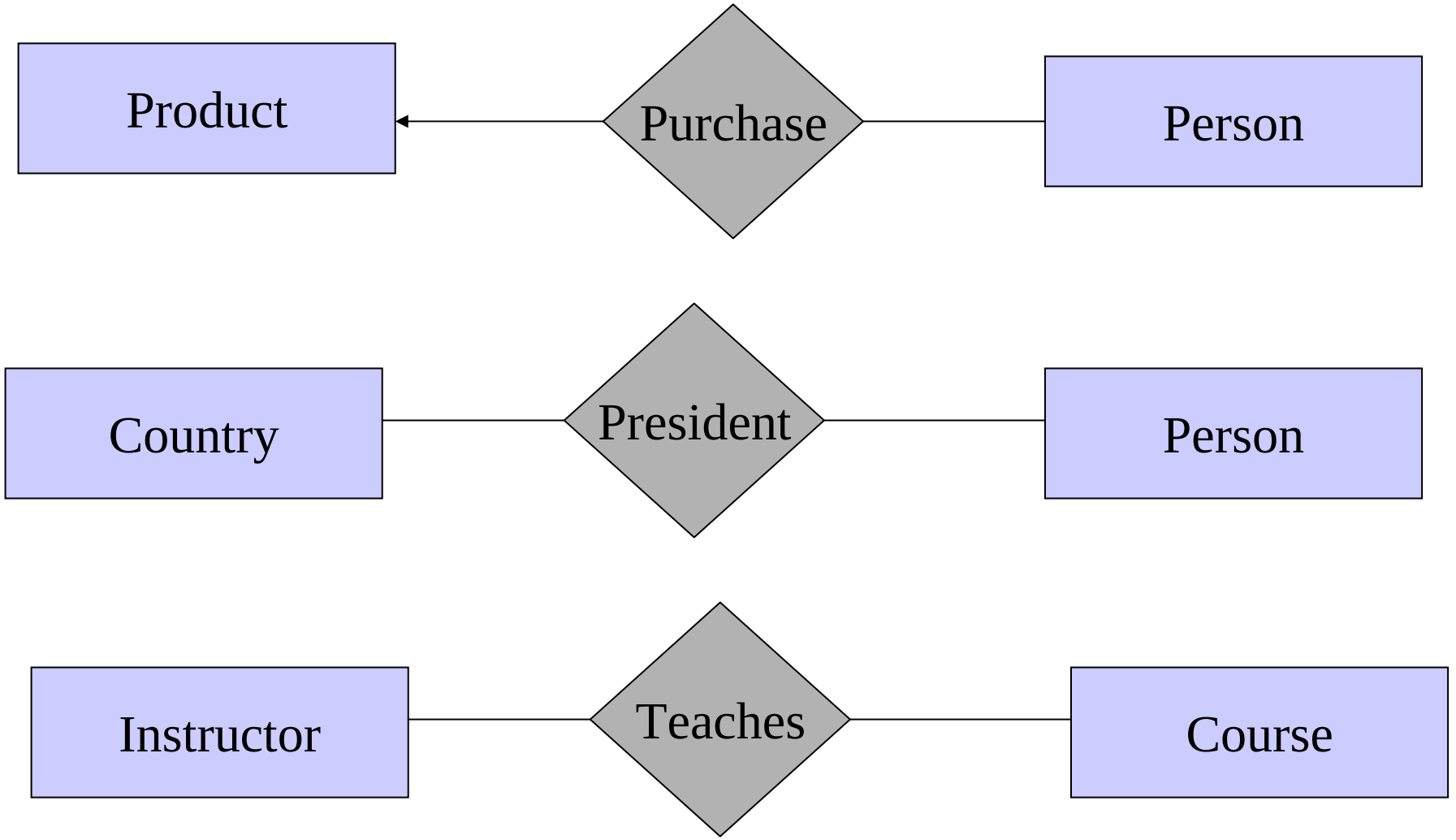


Другие виды ограничений

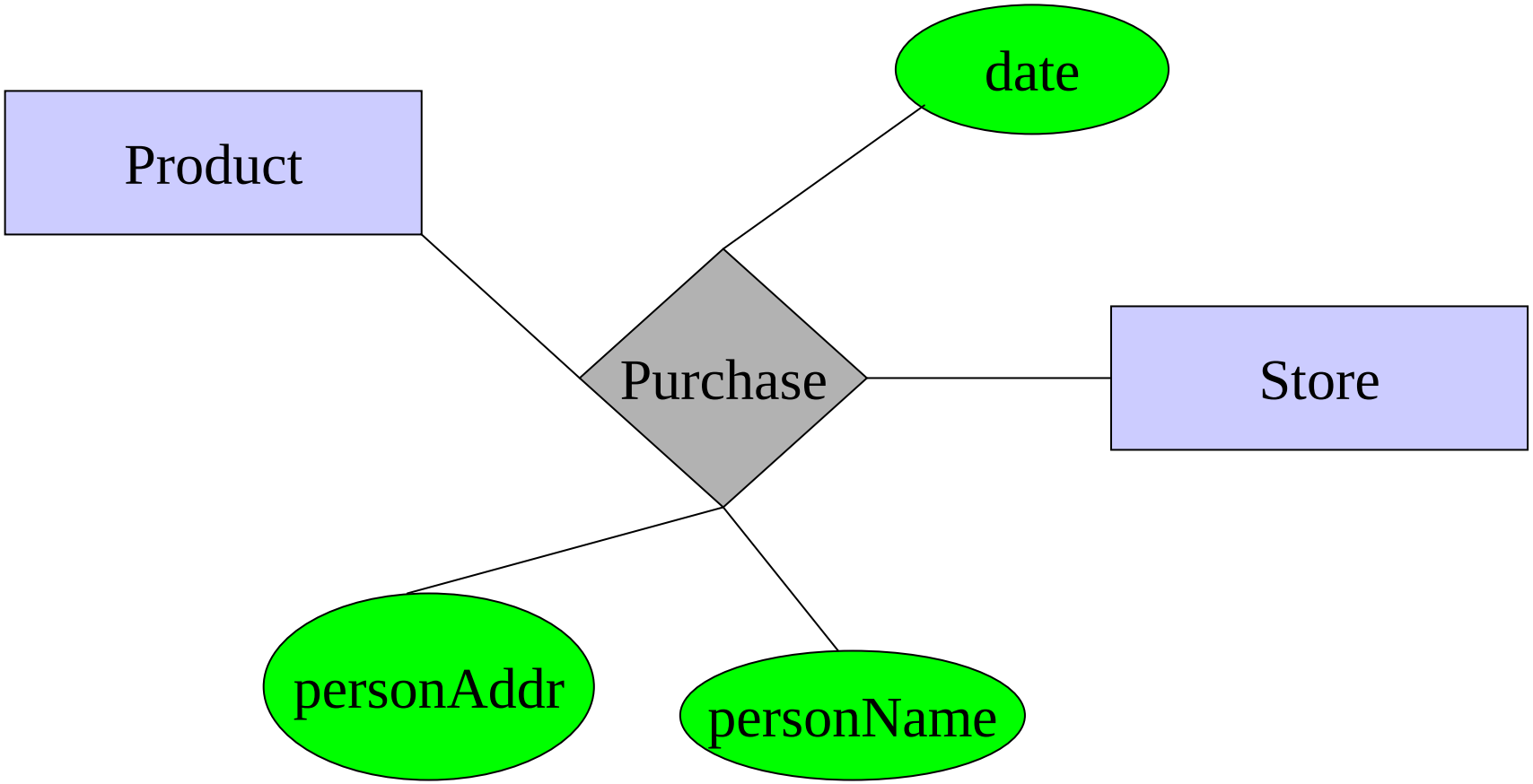
- ограничения области значений
- более общие ограничения
  - можно указать **комментариями**

Техники моделирования...

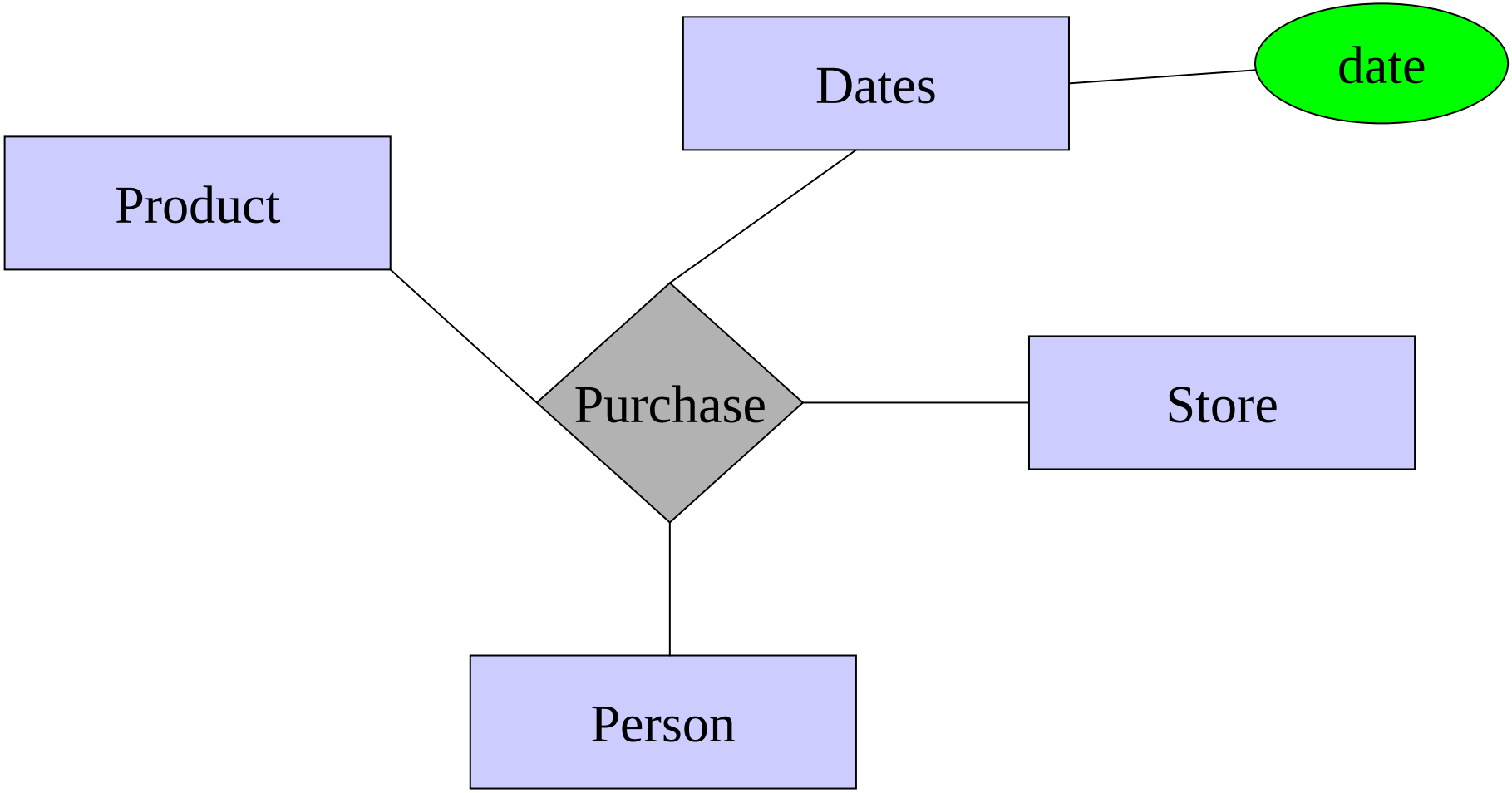
Принцип проектирования 1: **достоверность**, будь точен



Принцип проектирования 2: отсутствие избыточности



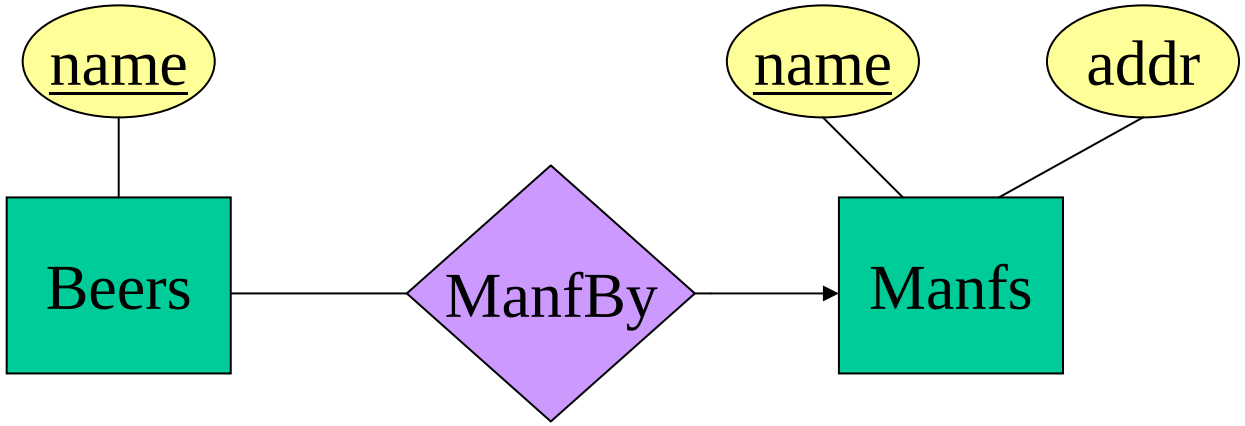
Принцип проектирования 3: адекватные сущности



Избегай избыточности

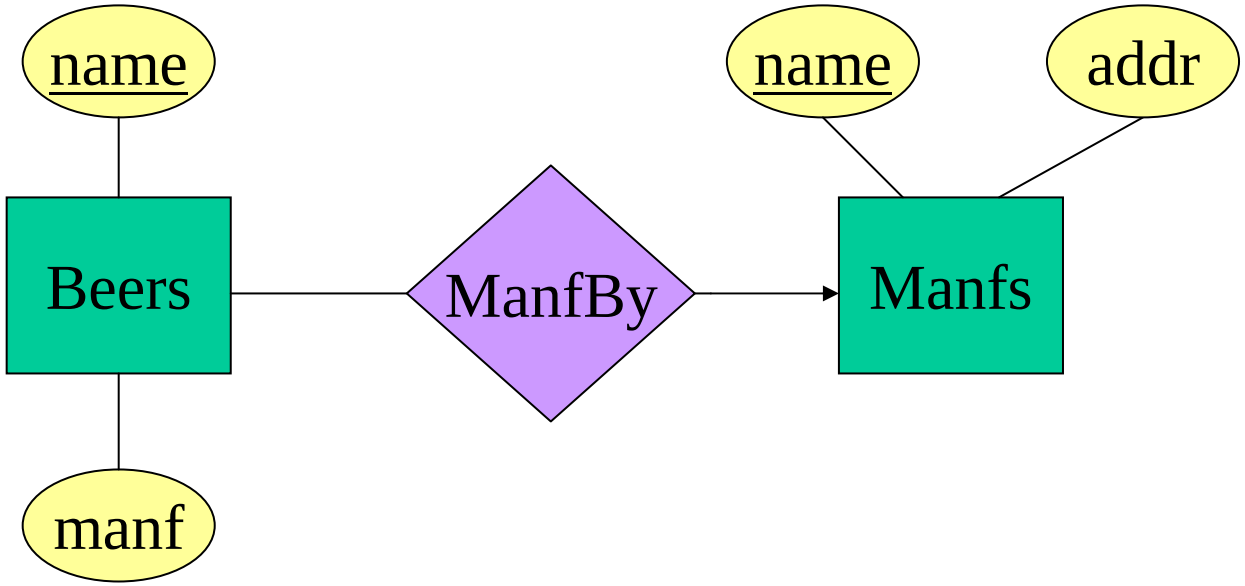
- Избыточность возникает, когда мы одно и тоже можем выразить/представить хотя бы **двумя различными способами.**
- Избыточность потребляет пространство и может привести к **нарушению согласованности данных**
  - Два экземпляра одного и того же факта могут стать противоречивыми, если мы изменим один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

Пример: хорошо



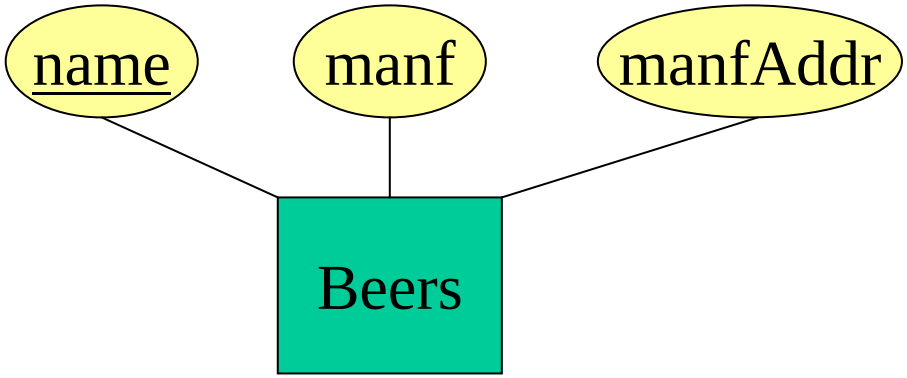
Производитель имеет ровно один адрес

Пример: плохо



Производитель пива указывается дважды:
как атрибут и как соответствующая сущность

Пример : плохо



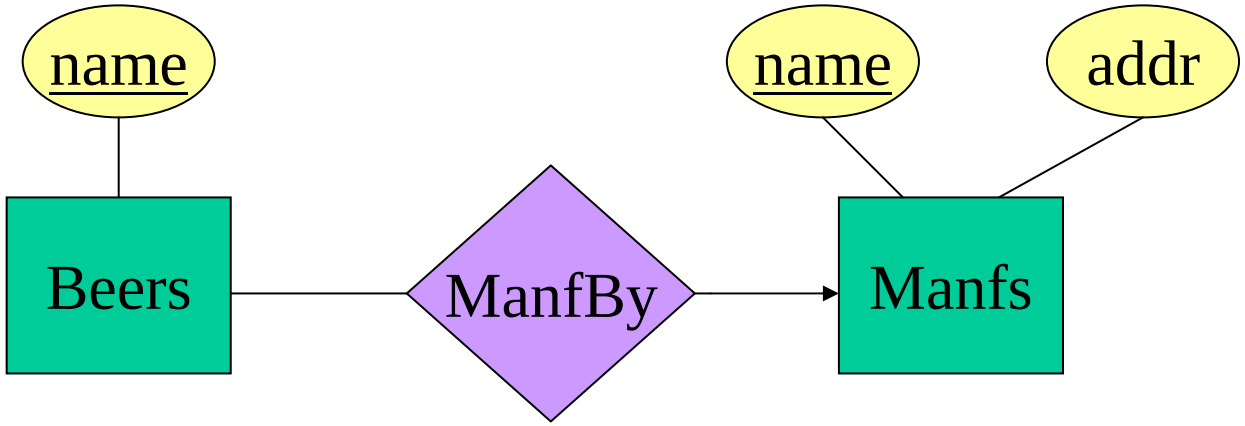
Повторяем адрес производителя каждый раз для каждого сорта пива.
Теряем адрес, если временно нет данных по сортам пива производителя

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

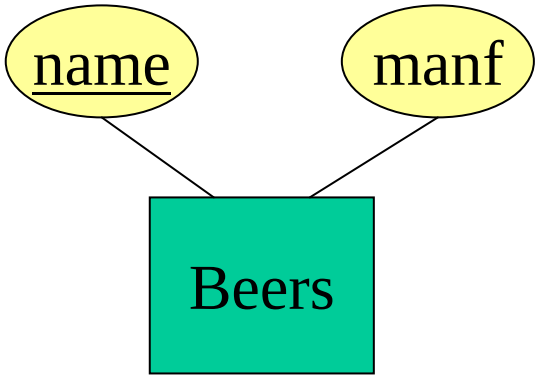
Пример : хорошо



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

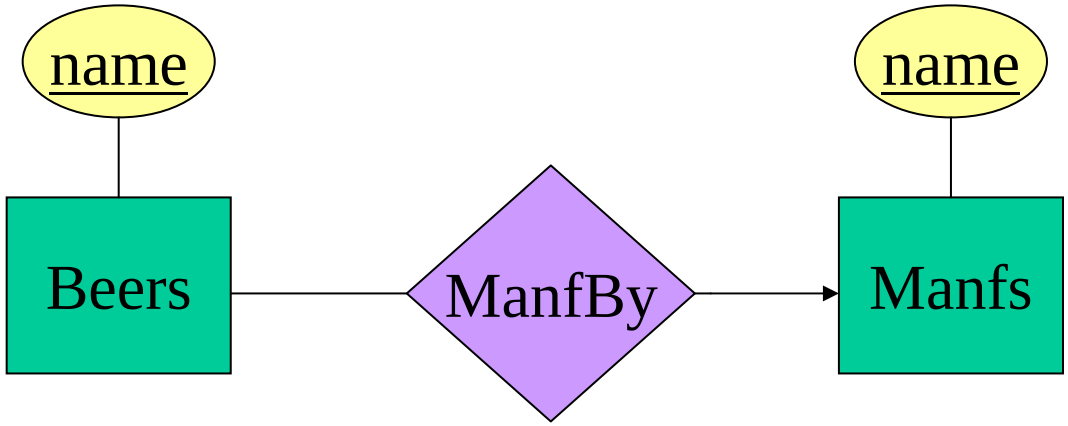
- *Beers* заслуживает того, чтобы быть сущностью, поскольку представляет “многие” в связи многие-к-одному с *ManfBy*

Пример: хорошо



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

Пример: плохо



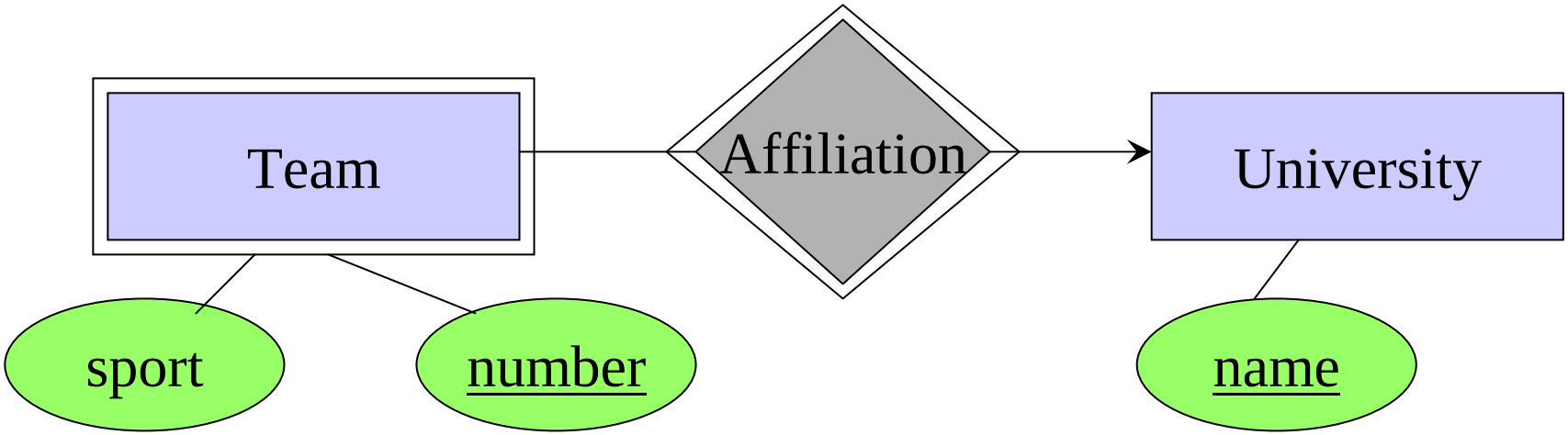
Поскольку фиксируем о производителе только имя и он не является концом “многие” в какой-либо связи, производитель не должен представляться множеством сущностей.

Слабые множества сущностей - 1

- 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

Слабые множества сущностей - 2



- недостаточно атрибутов, что составляющих сущность Team
- необходимо воспользоваться ключом сущности University через ассоциацию 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

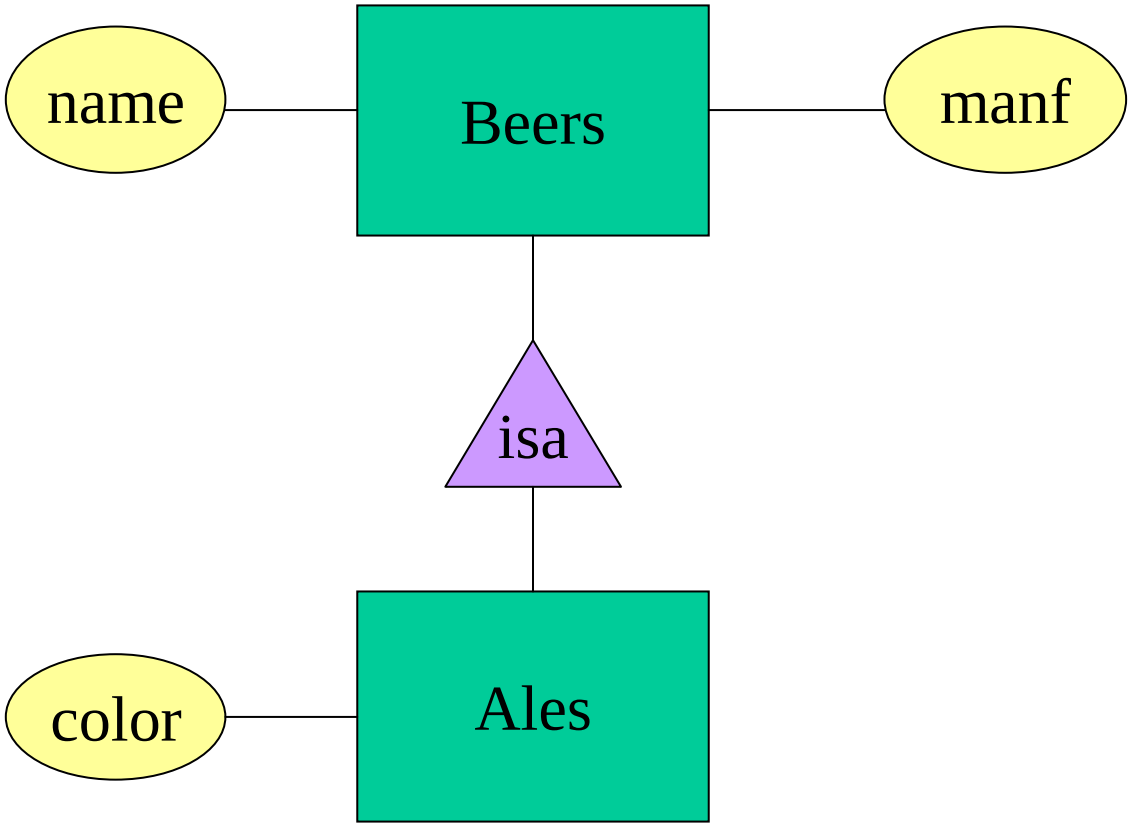
Подклассы в ER диаграммах

- **Подкласс** = специальный случай = **специализированные** сущности = больше свойств
- Подклассы образуют иерархии
  - нет множественного наследования
- **Isa** треугольник указывает связь между «классом» и «подклассом»
  - указывает на «суперкласс»
  - от «производного» к «базовому»

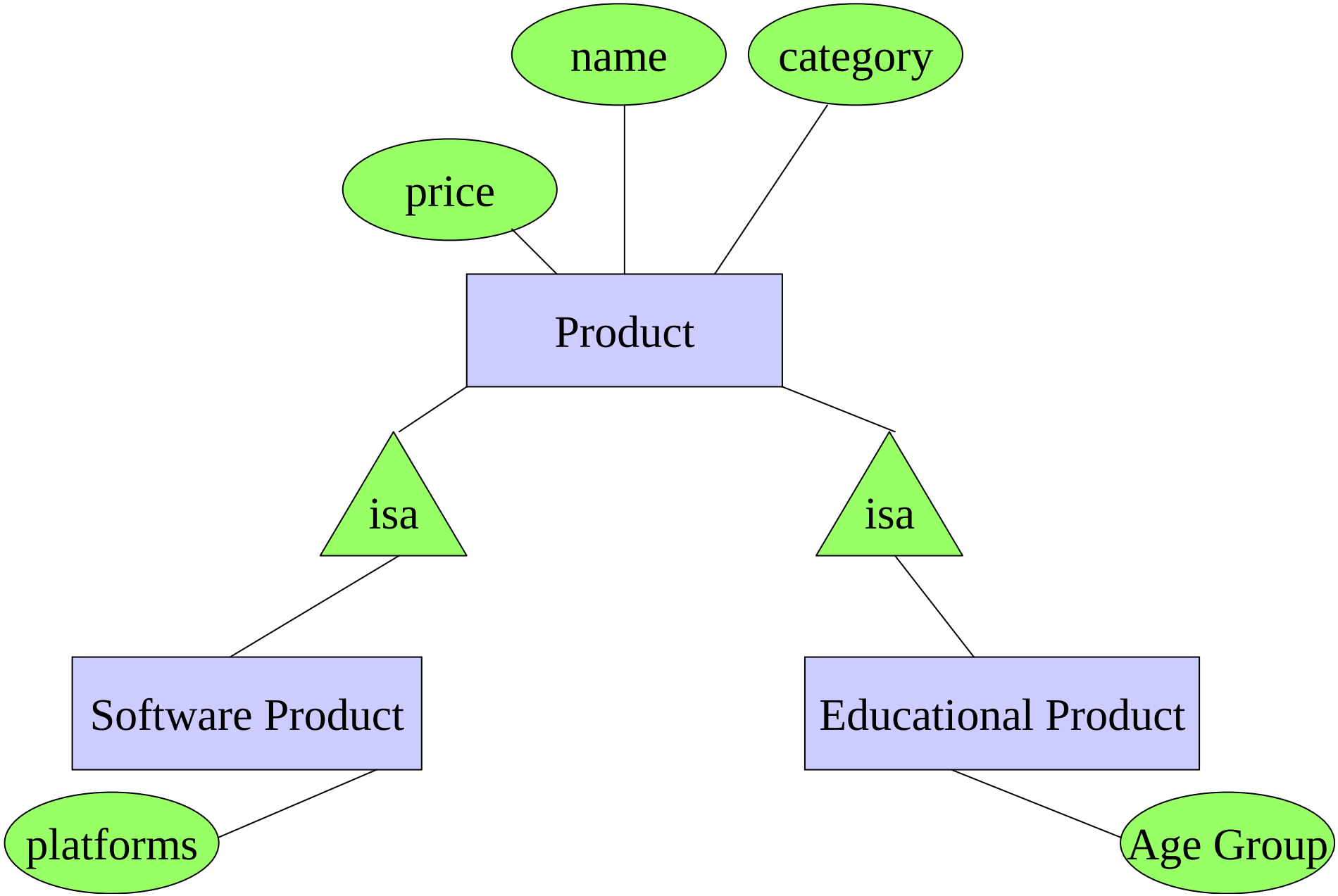
Пример

Эль – сорт пива

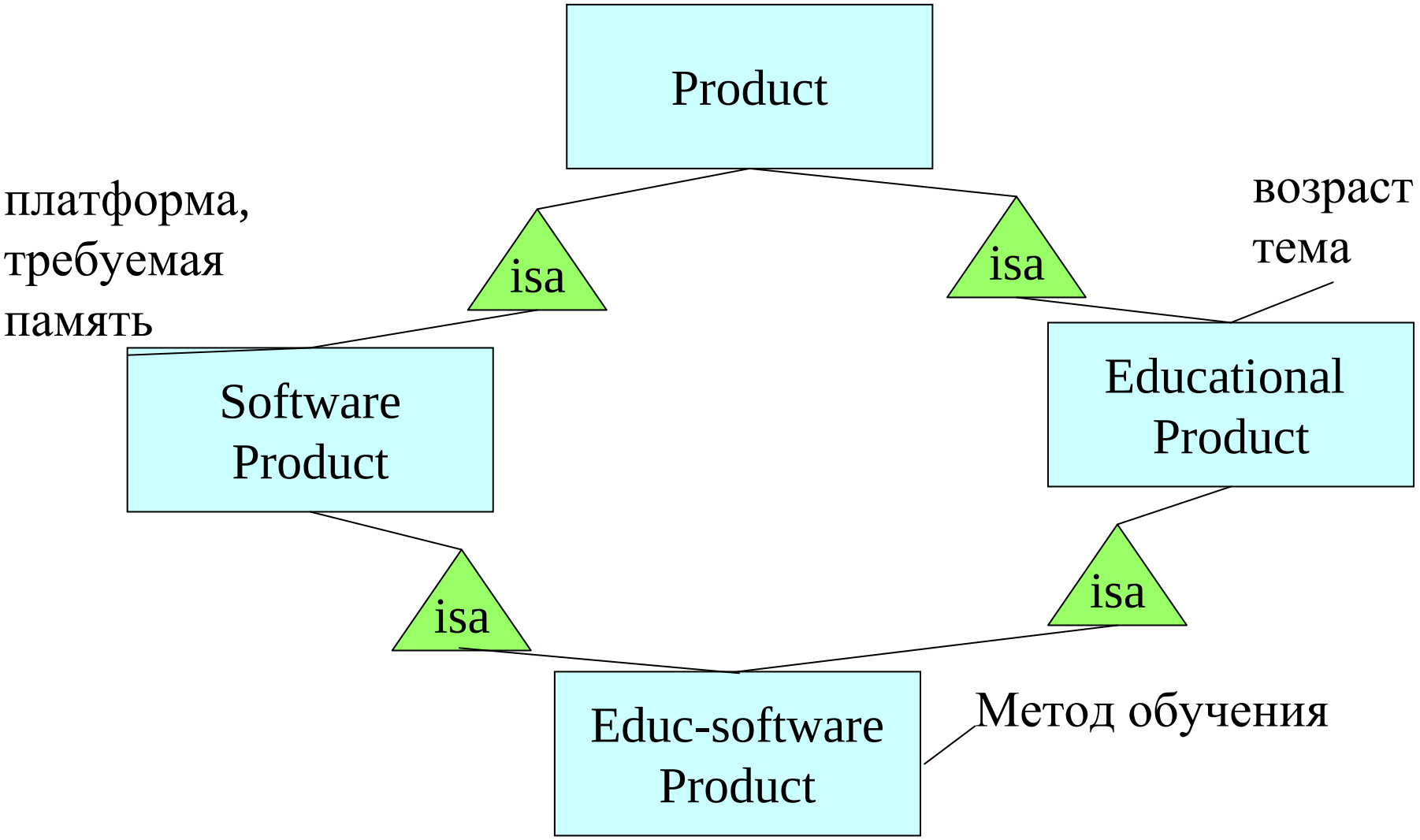
- Не все сорта пива - эль, но некоторые эль
- Пусть в дополнение ко всем свойствам пива (атрибуты и связи), эль имеет атрибут *color*



Подклассы ER диаграммах-1



Подклассы ER диаграммах -2



ER vs. объектно-ориентированные классы

- В объектно-ориентированном мире объекты относятся к одному классу
  - Подклассы наследуют свойства суперклассов
- ER сущности имеют компоненты/части во всех подклассах к которым они относятся
  - Следует учитывать при отображении ER схемы в отношения РМД

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 **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 неявных сущностей, "**замаскированных**" под атрибуты
  - устраняются **повторяющиеся** атрибуты или **группы** атрибутов
- 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, **isa** связей
- уточняемые степени связей
- разрешение связей «многие-ко-многим»
- определения потенциально допустимого множества значений атрибута сущности (**домена**)
  - ограничения целостности значений атрибутов
- определение **ограничений целостности** сущностей и связей
  - процедурные действия
- выявление **сильных связей** «один-ко-многим»
  - каскадные удаления

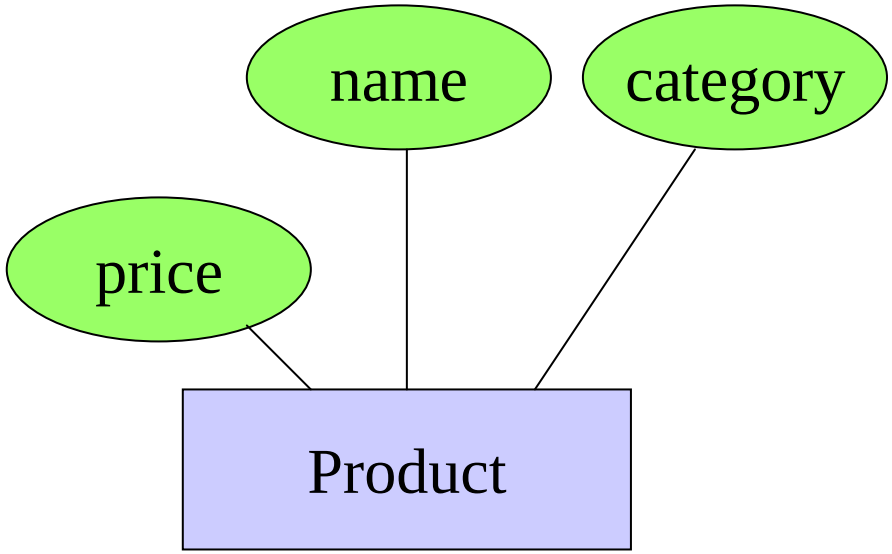
Трансляция ER модели в реляционную

Трансляция диаграмм в схему БД

- **Базовые варианты**
  - множество сущностей E **- отношение** с атрибутами E
  - связь R **- отношение с ключами связываемых** множеств сущностей + непосредственно атрибуты R
- **Специальные случаи**
  - связь один-к-многим
    - комбинирование двух отношений – **без отношения для связи**
  - трансляция слабых множеств сущностей
    - ключ включает **ключи поддерживающих** множеств сущностей
    - связи с **поддерживающими** множествами сущностей не формируют отношения
  - трансляция **isa** связей, подклассов

Базовые варианты

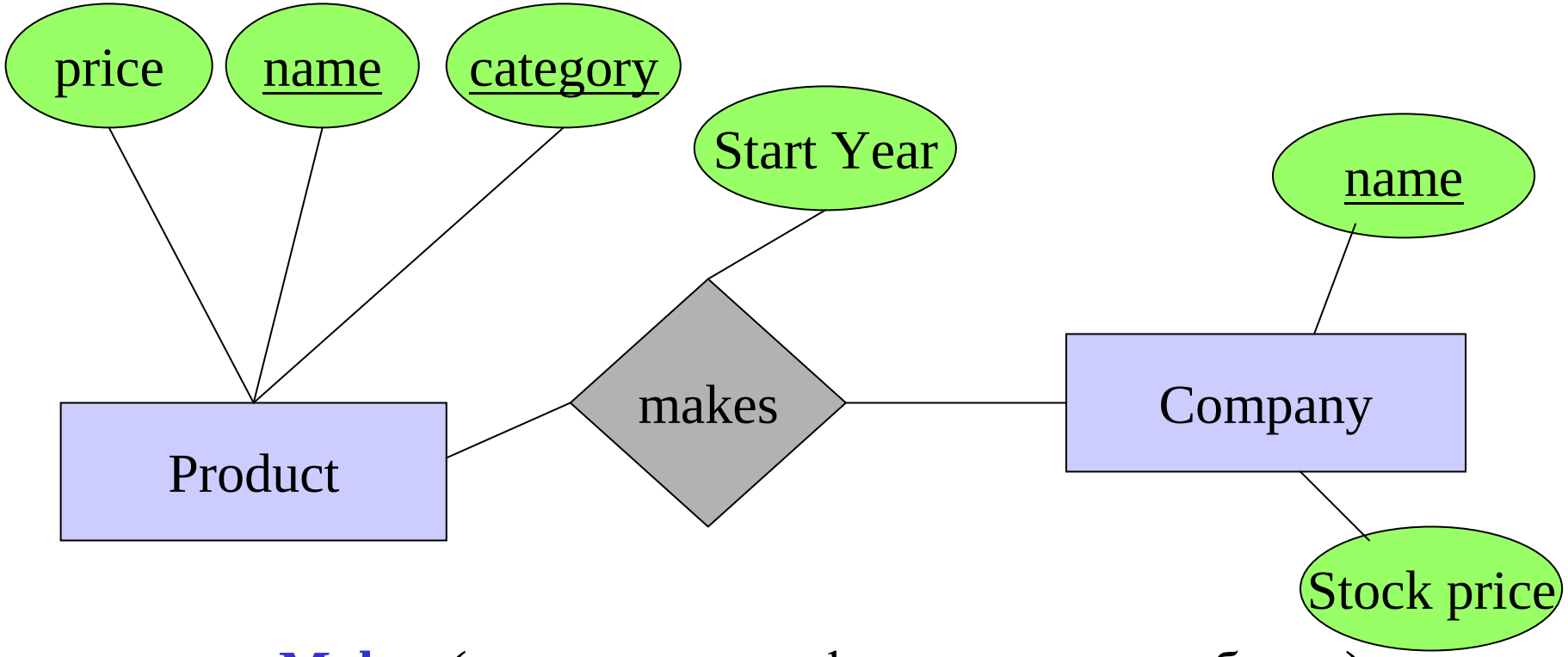
Множество сущностей в отношение



Product:

Name	Category	Price
кукла	игрушки	99.99

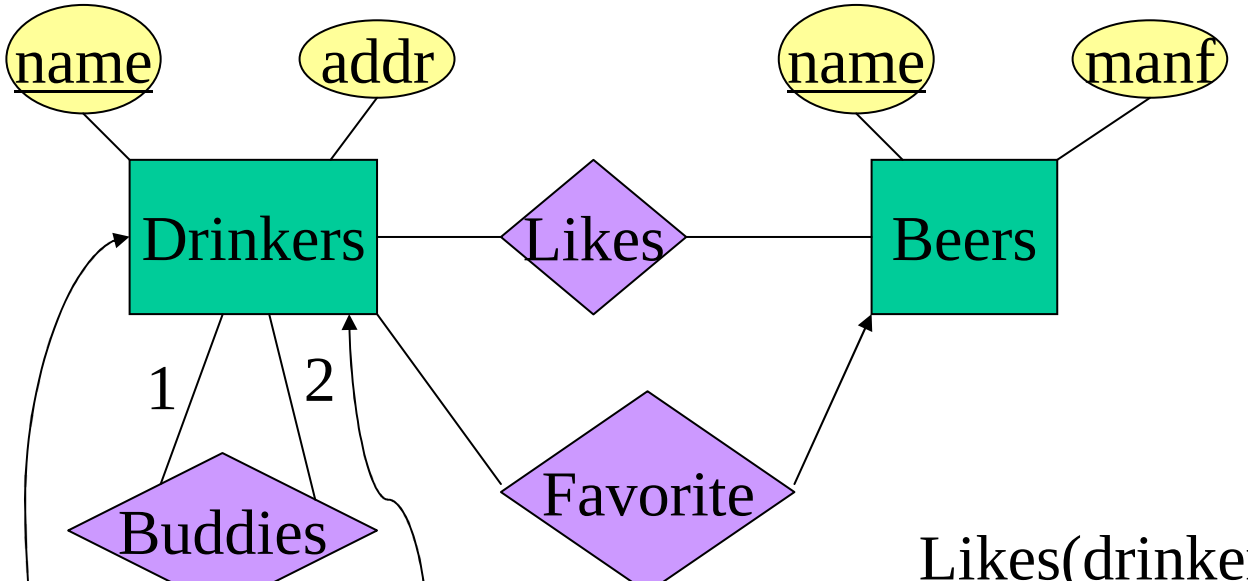
Связь в отношение - 1



отношение **Makes** (разрешаем конфликт имен атрибутов)

Product-name	Product-Category	Company-name	Starting-year
кукла	игрушки	ООО «Куклы»	1997

Связи в отношения - 2



Likes(drinker, beer)

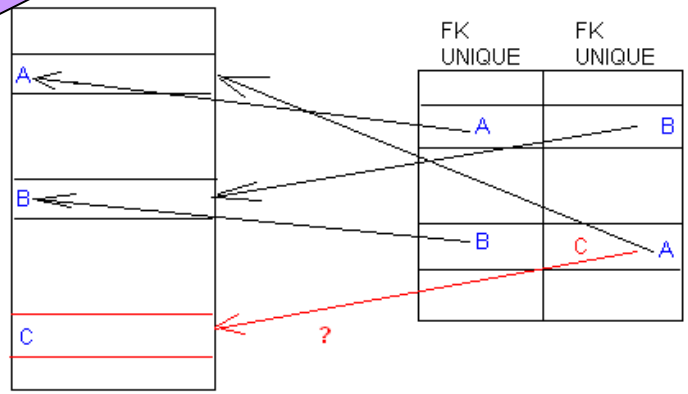
Favorite(drinker, beer)

Buddies(*name*1, *name*2)

Married(*husband*, *wife*)

unique(husband),**unique**(wife)

или FK Drinkers.spouse(1:m)?



Разные концептуальные связи – **одна реализация** кросс-таблицей

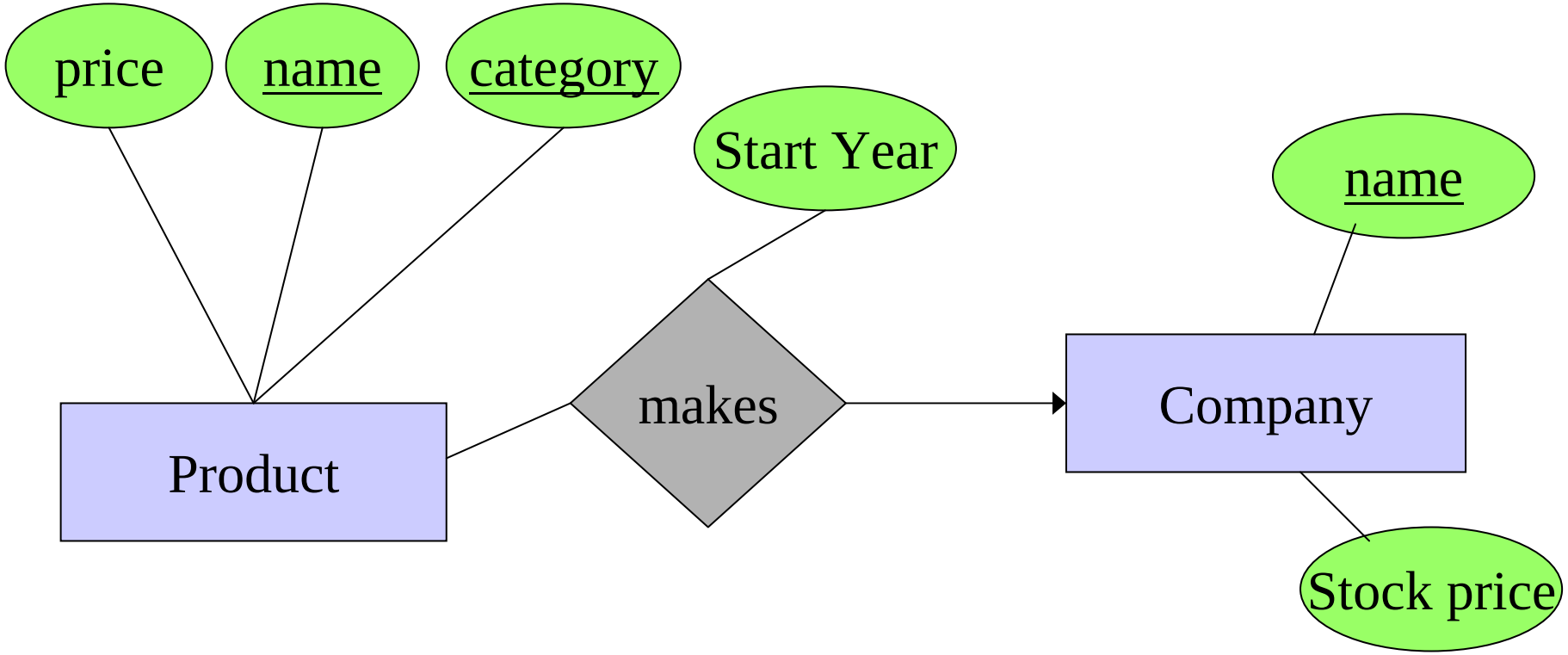
Специальные случаи:

1) отношения один-к-многим

2) слабые множества сущностей

3) isa связи

Комбинирование двух отношений

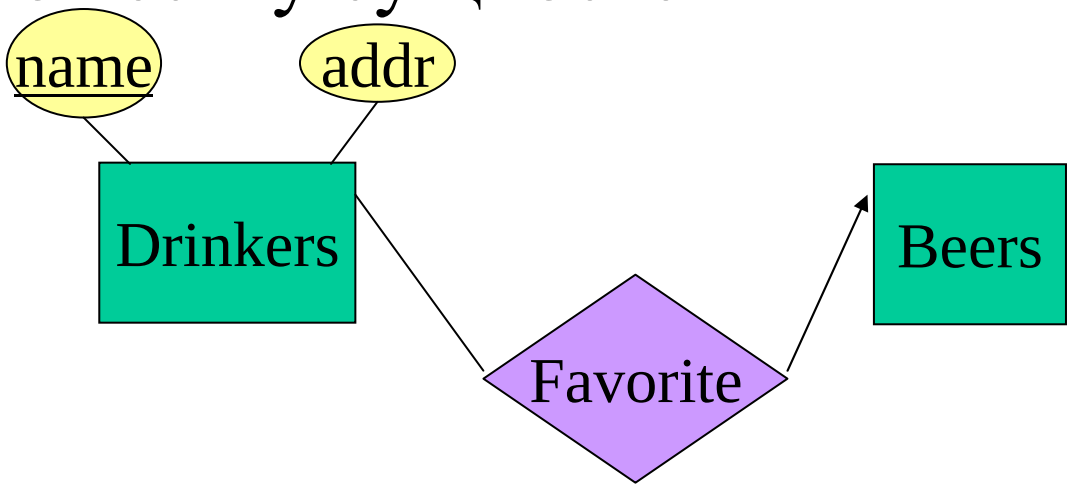


Не нужно отношение **Makes**. Просто модифицируем **Product**:

name	category	price	StartYear	companyName
кукла	игрушки	99.99	1997	ООО «Куклы»

Комбинирование отношений

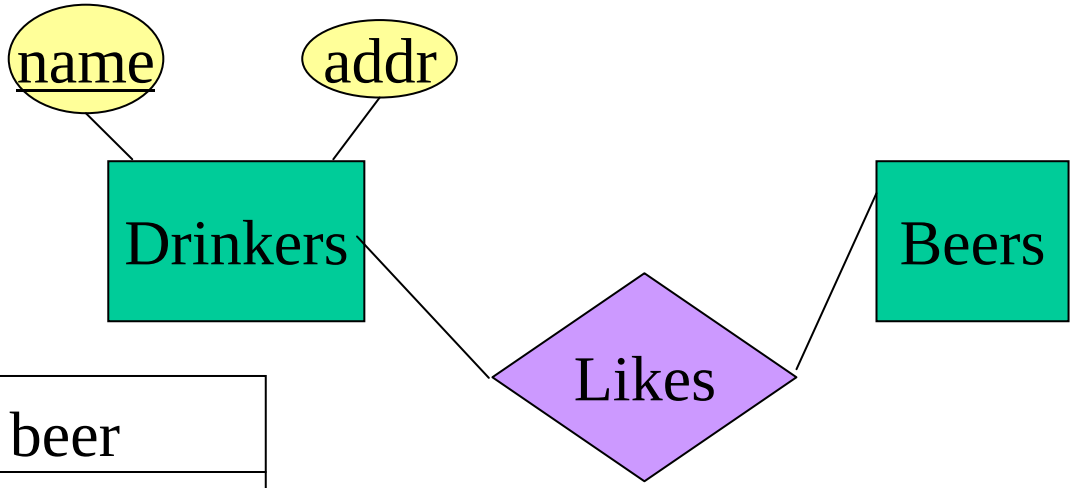
- Можно комбинировать отношение для множества сущностей *E* с отношением *R* для связи многие- к-одному «от» *E* к другому множеству сущностей



- Drinkers(name, addr) и Favorite(drinker, beer) можно скомбинировать, сделав **Drinker1(name, addr, favoriteBeer).**

Опасность связей многие-ко-многим

- Комбинирование Drinkers с Likes приводит к ошибке - **избыточность**



name	addr	beer
Алех	Южная 123	Бад
Алех	Южная 123	Миллер

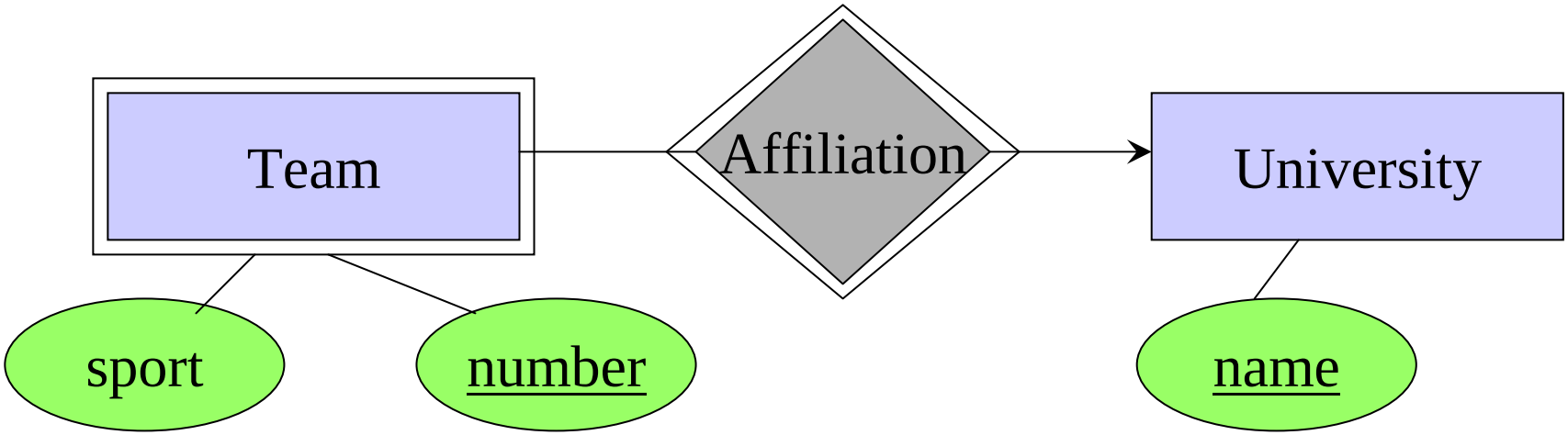
Избыточность

Слабые множества сущностей

- отношение для слабых множеств сущностей должно включать атрибуты, составляющие их ключ (**включая те, что относятся к другим** множествам сущностей), и собственные атрибуты, не являющиеся ключами.

- Связи слабых множеств сущностей **не формируют** отношение.

Трансляция слабых множеств сущностей

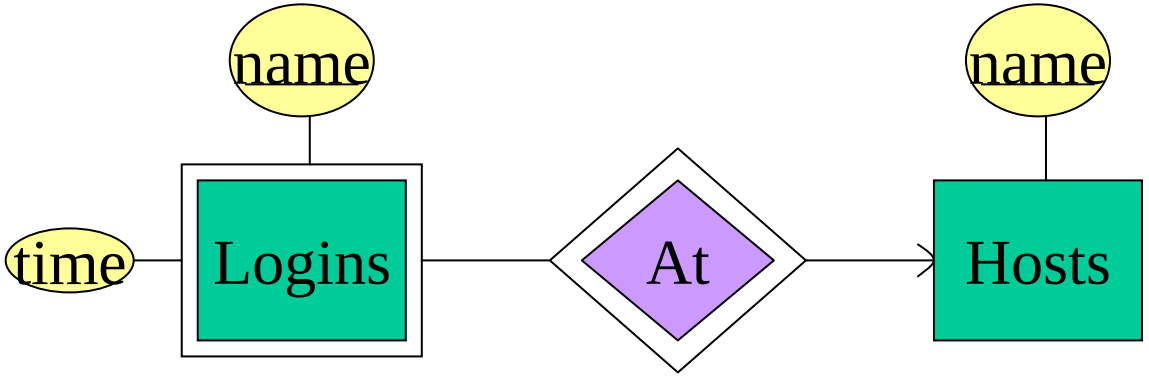


отношение Team:

Sport	Number	AffiliatedUniversity
плавание	15	МФТИ

- ассоциации слабых множеств сущностей **не формируют отношение**.
- нужны все атрибуты, что составляют ключ сущности Team
- не нужно отношение для Affiliation

Пример



Hosts(hostName)

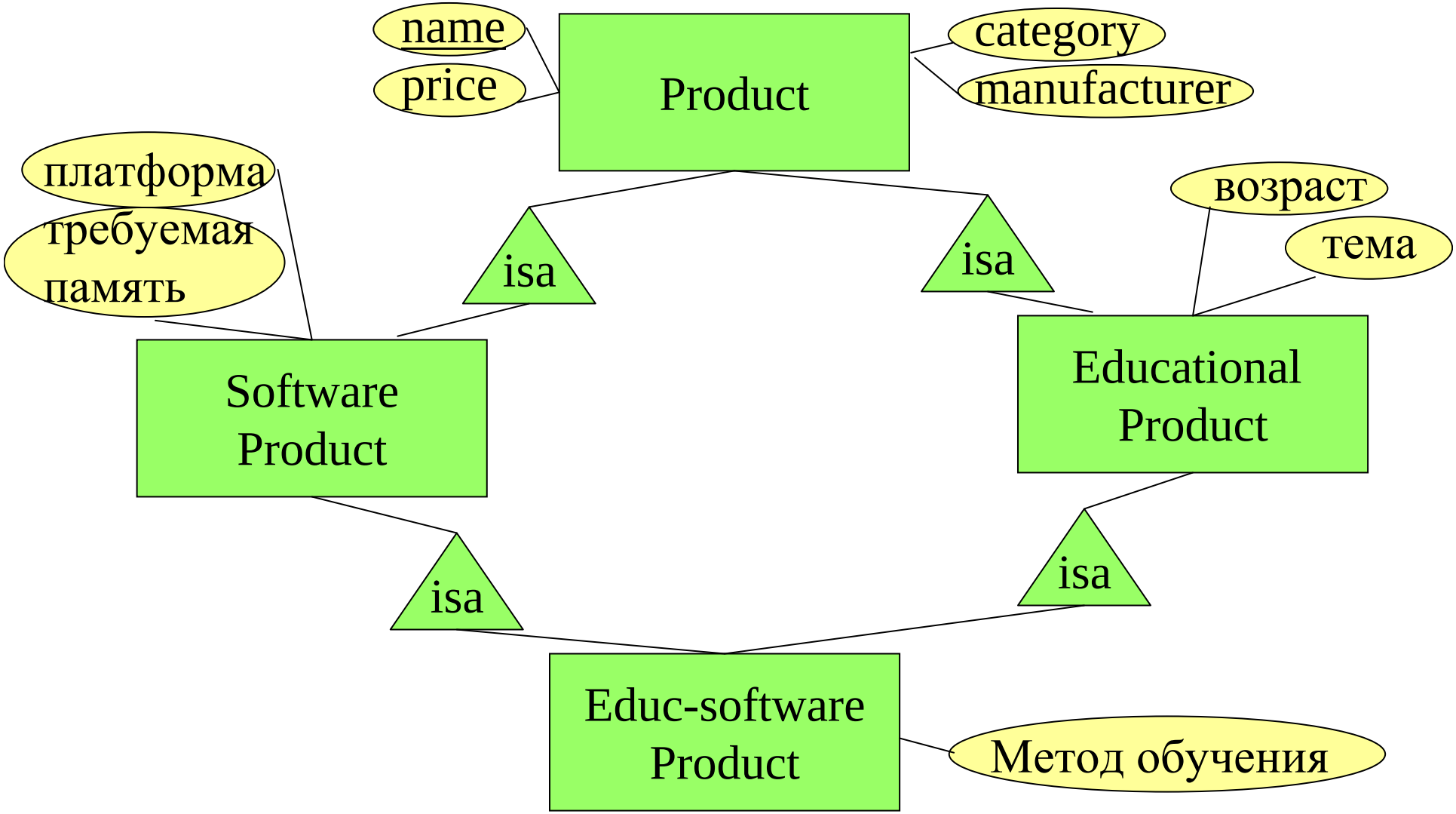
Logins(loginName, hostName, time)

~~At(loginName, hostName, hostName2)~~

Становится часть Logins

Должны быть одно и то же

Трансляция подклассов сущностей



Трансляция подклассов сущностей: правила

Три подхода:

1. *Объектно-ориентированный*: каждая сущность относится строго к одному классу; создаем отношение **для каждого класса со всеми его** атрибутами
2. *ER стиль* : создаем одно отношение для **каждого подкласса с ключом (ами) и атрибутами этого** подкласса; сущность представляется всеми отношениями к которым подкласс относится
3. *Использование NULL* : создать одно отношение; сущности принимаю **NULL значения для атрибутов, не относящихся к ним**

Вариант 1: OO подход

4 таблицы:

каждый объект может относиться только к одной таблице

Product(name, price, category, manufacturer)

EducationalProduct(name, price, category, manufacturer,
ageGroup, topic)

SoftwareProduct(name, price, category, manufacturer,
platforms, requiredMemory)

EducationalSoftwareProduct(name, price, category, manufacturer,
ageGroup, topic,
platforms, requiredMemory)

Все name должны быть различны

Вариант 2: ER подход

Product(name, price, category, manufacturer)

EducationalProduct(name, ageGroup, topic)

SoftwareProduct(name, platforms, requiredMemory)

Не нужно отношение EducationalSoftwareProduct ,
пока нет собственного атрибута:
EducationalSoftwareProduct(name, educationalMethod)

Те же самые name могут быть в нескольких отношениях

Вариант 3: использование NULL значений

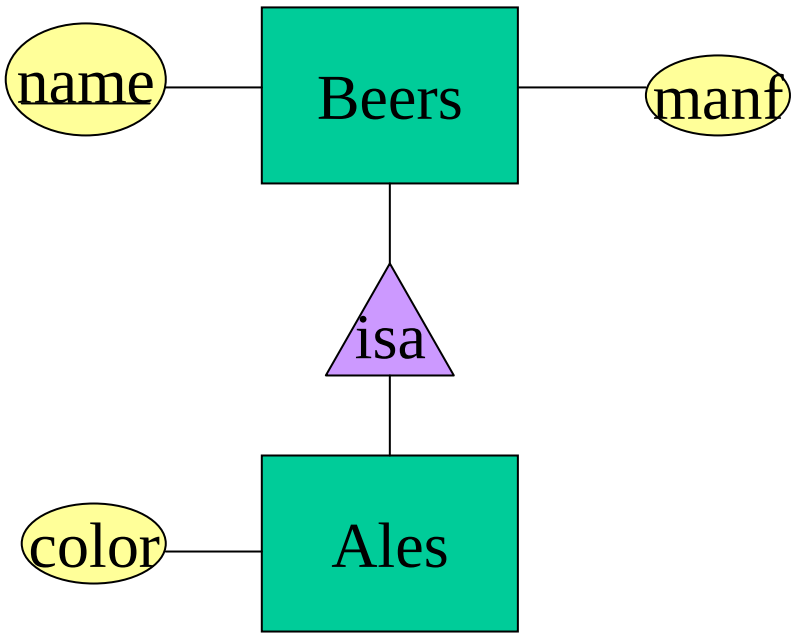
Формируем одну таблицу:

Product (name, price, category, manufacturer,
ageGroup, topic,
platforms, requiredMemory,
educationalMethod)

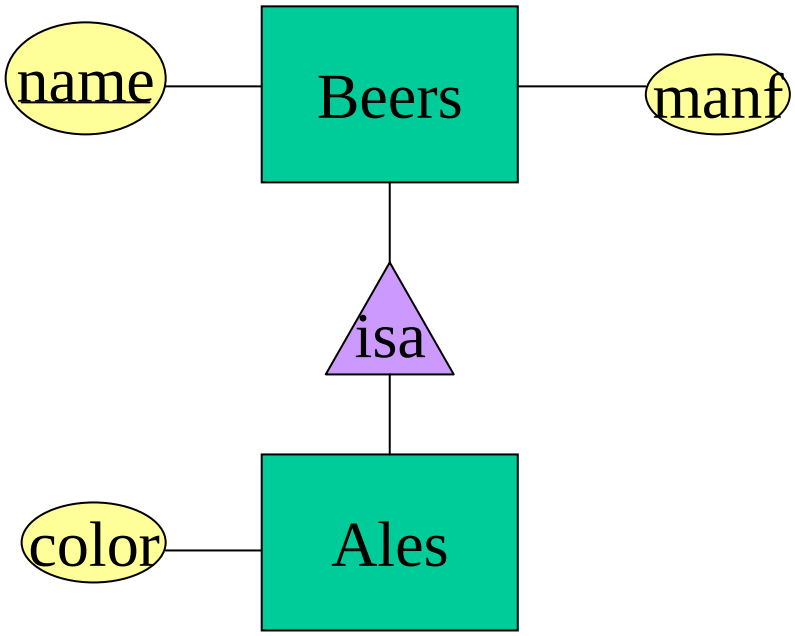
Некоторые значения в таблице будут NULL,
Это означает, что атрибут **не имеет смысла** для данного продукта

Слишком много смысловых значений для NULL

Пример



Объектно-ориентированный



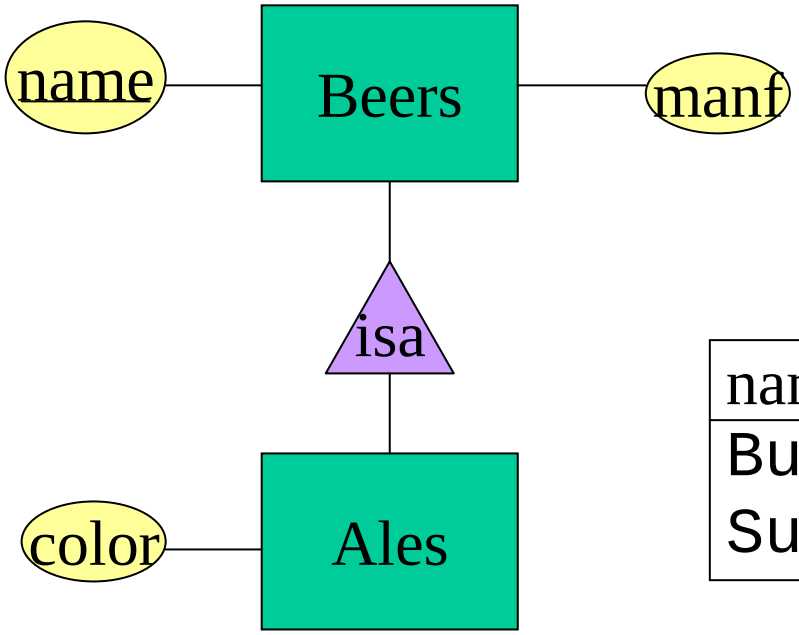
name	manf
Bud	Anheuser-Busch

Beers

name	manf	color
Summerbrew	Pete's	dark

Ales

ER стиль



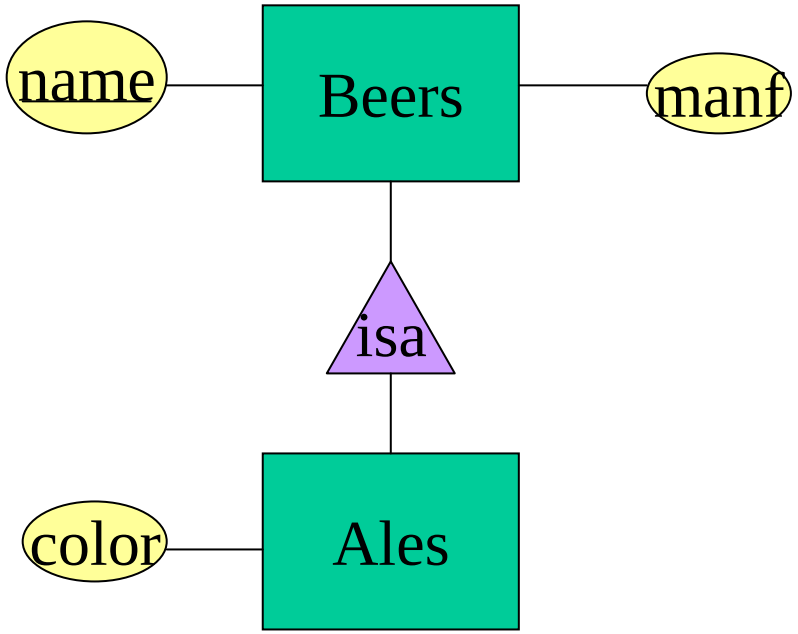
name	manf
Bud	Anheuser-Busch
Summerbrew	Pete's

Beers

name	color
Summerbrew	dark

Ales

Использование NULL значений



name	manf	color
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	dark

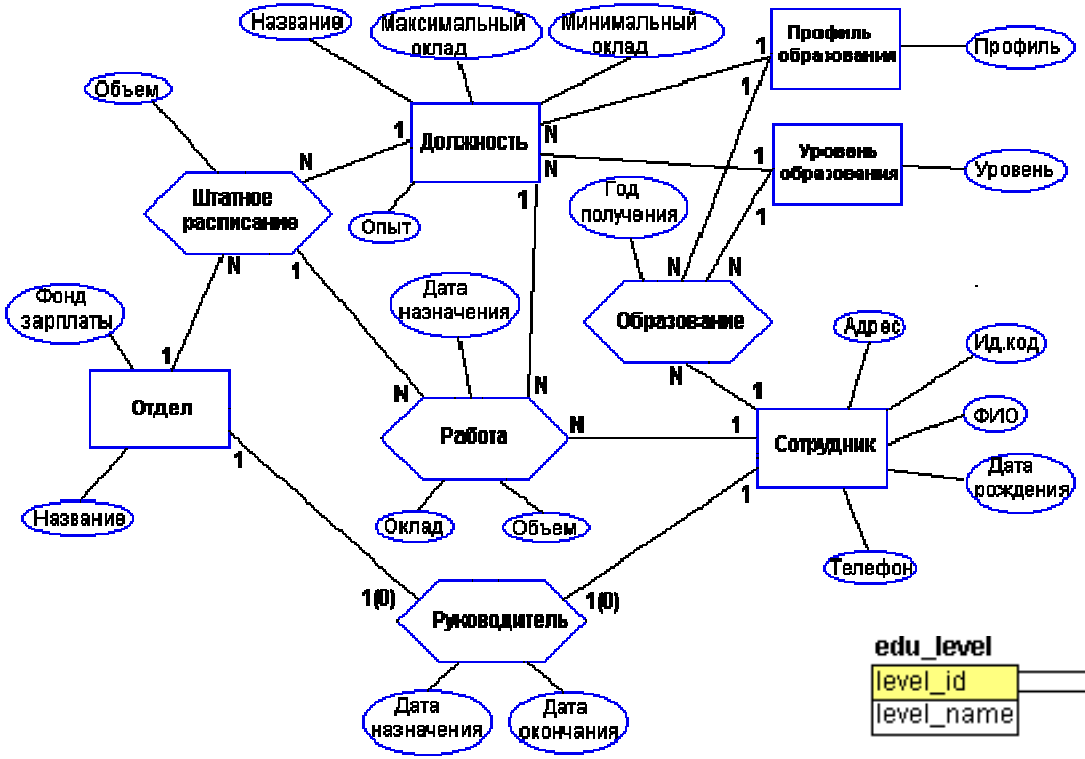
Beers

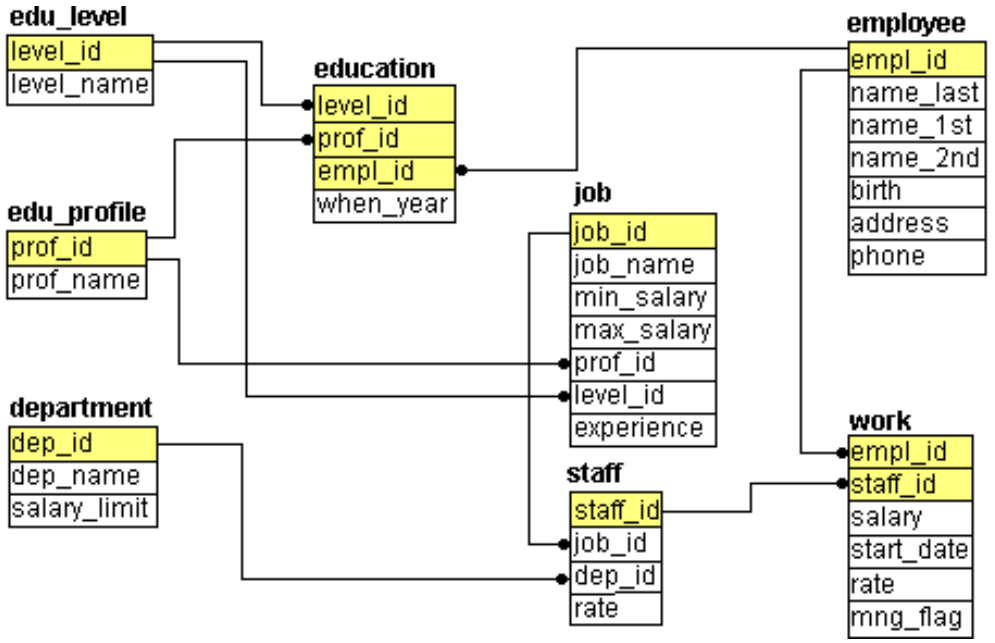
Сравнение

- O-O подход хорошо для запросов вида “найти все цвета элей, выпускаемых «Pete’s»”
  - Обратиться **только к** отношению Ales
- ER подход хорош для запросов вида “найти все сорта пива (включая эли) , выпускаемых «Pete’s»”
  - Обратиться **только к** отношению Beers
- Используя NULL значений приходится расходовать существенный объем памяти, если набирается изрядно количество атрибутов, принимающих NULL значения

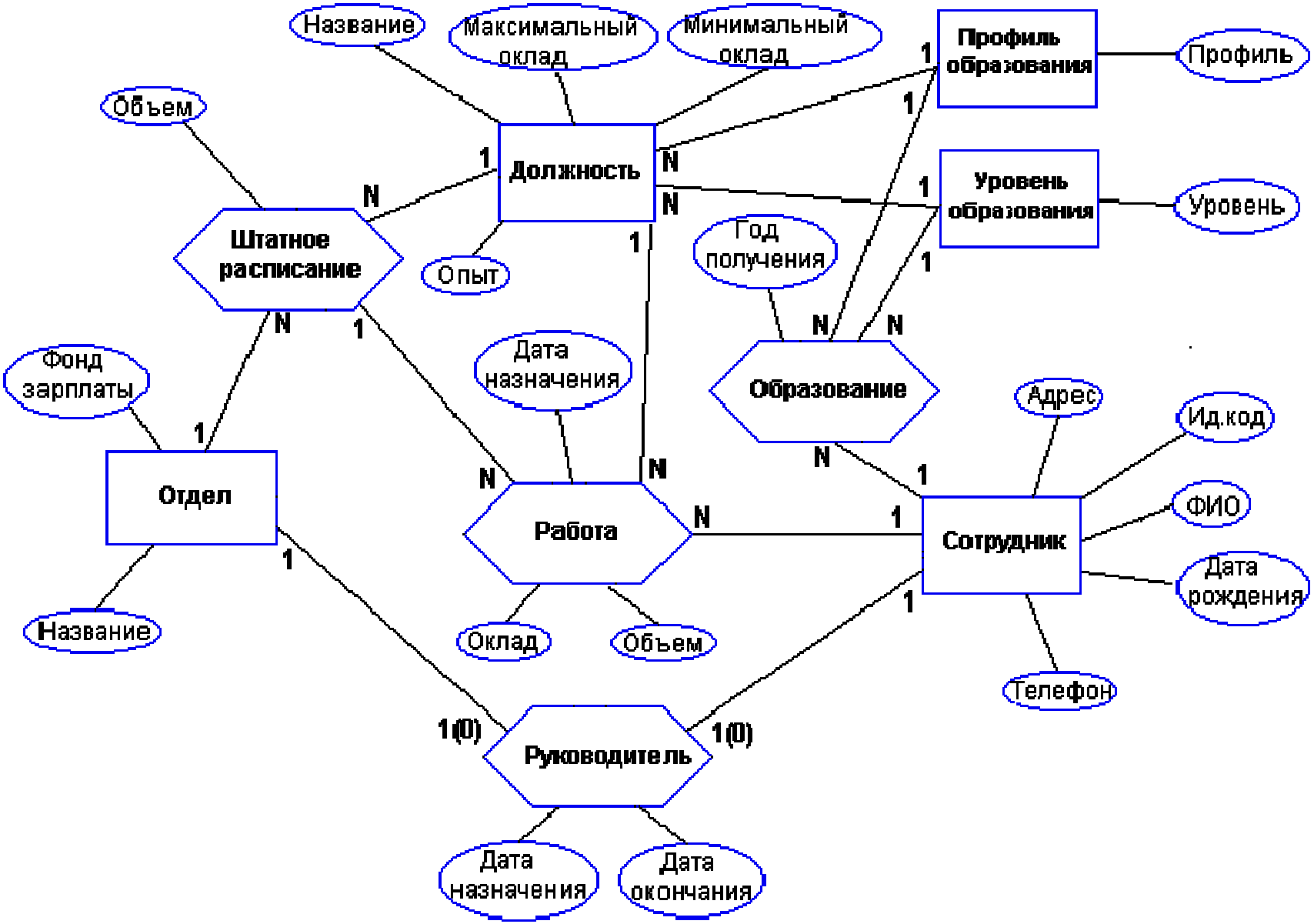
Запись схемы БД

Лаб.2 - концептуальная и ER диаграммы

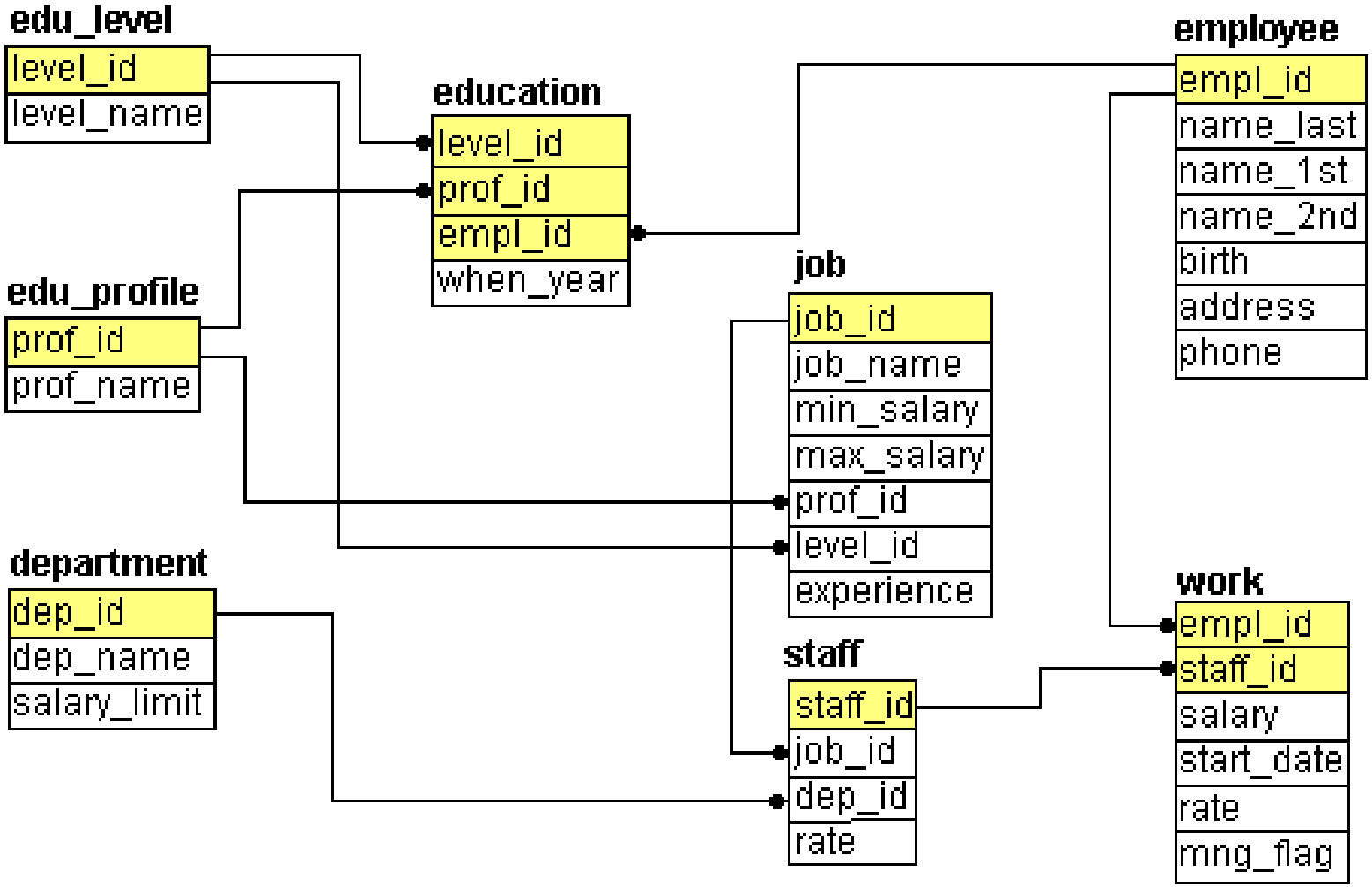




Лаб.2 - концептуальная диаграмма



Лаб.2 - ER-диаграмма



Лаб.2 – DDL скрипт

```
CREATE TABLE job (
  job_id INTEGER
    NOT NULL PRIMARY KEY,
  job_name VARCHAR(40),
  min_salary DECIMAL(7,2),
  max_salary DECIMAL(7,2),
  level_id INTEGER NOT NULL
    REFERENCES edu_level(level_id)
    ON DELETE CASCADE ON UPDATE RESTRICT,
  prof_id INTEGER NOT NULL
    REFERENCES edu_profile(prof_id)
    ON DELETE CASCADE ON UPDATE RESTRICT,
  experience SMALLINT
)
CREATE TABLE staff (
  staff_id INTEGER
    NOT NULL PRIMARY KEY,
  job_id INTEGER NOT NULL
    REFERENCES job(job_id)
    ON DELETE CASCADE ON UPDATE RESTRICT,
  dep_id INTEGER NOT NULL
    REFERENCES department(dep_id)
    ON DELETE CASCADE ON UPDATE RESTRICT,
  rate DECIMAL (6,2) NOT NULL
)
CREATE TABLE work (
  empl_id DECIMAL (10,0) NOT NULL
    REFERENCES employee(empl_id)
    ON DELETE CASCADE ON UPDATE RESTRICT,
  staff id INTEGER NOT NULL
```

Создание таблиц – CREATE TABLE

```
CREATE TABLE
   [ database_name.[ owner ] . | owner. ] table_name
   (
    { < column_definition >
     | column_name AS computed_column_expression
     | < table_constraint >
     } [ ,...n ]
   )
```

Создание таблиц– column_definition

< column_definition > ::=

column_name *data_type*

[DEFAULT *constant_expression* |

IDENTITY [(*seed* , *increment*)]

]

{ [< column_constraint >] } [...*n*]

Создание таблиц– column_constraint

< column_constraint > ::= [CONSTRAINT *constraint_name*]

{ [NULL | NOT NULL]
| [PRIMARY KEY | UNIQUE]
| [[FOREIGN KEY]
REFERENCES *ref_table* [**(** *ref_column* **)**]
[ON DELETE { CASCADE | SET NULL }]
[ON UPDATE { CASCADE | SET NULL }]
]
| CHECK **(** *logical_expression* **)**
}

Создание таблиц – table_constraint

< table_constraint > ::= [CONSTRAINT *constraint_name*]
{
{PRIMARY KEY | UNIQUE} { **(** *column* [**,**...*n*] **)** }
| FOREIGN KEY [**(** *column* [**,**...*n*] **)**]
REFERENCES *ref_table* [**(** *ref_column* [**,**...*n*] **)**]
[ON DELETE { CASCADE | SET NULL}]
[ON UPDATE { CASCADE | SET NULL }]
| CHECK **(***logical_expression* **)**
}

Пример

```
create table student

(studentno number(8) primary key,
givenname char(20),
surname char(20),
hons char(3) check (hons in ('cis','cs','ca','pc','cm','mcs')),
tutorid number(4),
yearno number(1) not null,

constraint year_fk
  foreign key (yearno) references year(yearno),
constraint super_fk
  foreign key (tutorid) references staff(staffid))
```

Alter - модификация схем отношений

- ```
  alter table student
     add (address char(20),
              default null);

  alter table student
     modify (name not null)
  ```
  ```

Оператор изменения схемы таблицы

<**alter table statement**> ::=

ALTER TABLE <table name> <alter table action>

<**alter table action**> ::=

<add column definition>

|<alter column definition>

|<drop column definition>

|<add table constraint definition>

|<drop table constraint definition>

Ограничения ссылочной целостности

- **Ограничения ссылочной целостности**:
  - ровно **одно значение имеется в данной роли**
  - явно требуют, чтобы ссылка существовала – **указывала на существующий** объект
- Если атрибут имеет обязательное значение
  - то это можно рассматривать как вид «ограничения ссылочной целостности».

Определение внешних ключей

- Используя ключевое слово REFERENCES
  1. внутри объявления атрибута, когда в ключе используется один атрибут
  2. как элемент схемы :

  FOREIGN KEY (<список атрибутов>)
  REFERENCES <отношение> (<список атрибутов>)
- Атрибуты, на которые ссылаемся, должны быть объявлены как PRIMARY KEY или UNIQUE
- Имена атрибутов первичного и внешнего ключа могут различаться, домены должны быть одинаковыми

Нарушение ограничений внешнего ключа

- Если имеется ограничение внешнего ключа на атрибуты отношения *R*, которое ссылается на первичный ключ отношения *S*, то возможны два вида нарушений:
  1. Вставка (insert) или модификация (update) *R* вводит значения отсутствующие в *S*.
  2. Удаление (delete) или модификация (update) S приводит к тому, что кортежи в *R* «повисают» - не имеют «родителя»

Предпринимаемые действия - 1

- Пусть R = Sells, а S = Beers.
- Вставка или модификация Sells, вводящие несуществующие сорта пива должны отклоняться
- Удалением или модификацией Beers сортов пива, которые имеются в кортежах Sells, можно управлять тремя способами

Предпринимаемые действия-2

- Три способа управления сортами пива, которые имеются в кортежах Sells, но внезапно исчезают:
  1. *По умолчанию*:
     - отклонить модификацию
  2. *Каскадные изменения*: те же самые изменения провести в Sells.
     - Удален сорт пива: удалить кортежи в Sells
     - Изменен сорт пива: изменить значения в Sells
  3. *Установить в NULL* :
     - установить сорт пива в NULL (неизвестно)

Выбор действия

- Когда объявляем внешний ключ, мы можем выбрать вид действия SET NULL или CASCADE независимо для удалений(delete) и модификаций (update),
- следуя спецификации:

ON [UPDATE, DELETE][SET NULL | CASCADE]

- можно указать два таких предложения,
- иначе, будет использовано действие по умолчанию (отклонить операцию)

БД примеров

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

  Beers(<u>name</u>, manf)

  Bars(<u>name</u>, addr, license)

  Drinkers(<u>name</u>, addr, phone)

  Likes(<u>drinker</u>, <u>beer</u>)

  Sells(<u>bar</u>, <u>beer</u>, price)

  Frequents(<u>drinker</u>, <u>bar</u>)

  – Подчеркнуты атрибуты первичного ключа

Пример

```
CREATE TABLE Sells (
   bar       CHAR(20),
   beer      CHAR(20),
   price     REAL,
   FOREIGN KEY(beer)
       REFERENCES Beers(name)
       ON DELETE SET NULL
       ON UPDATE CASCADE )
```

Пример: Каскадные изменения

- **ON DELETE/UPDATE CASCADE**
- Если мы удаляем сорт ‘Bud’ в кортеже отношения Beers
  - то удалить все кортежи в Sells, имеющие beer = ’Bud’
- Если мы изменяем кортеж с ‘Bud’, заменяя ’Bud’ на’Budweiser’,
  - то изменить все кортежи в Sells имеющие beer = ’Bud’ так, чтобы было beer = ’Budweiser’.

Пример: установление в NULL

- **ON DELETE/UPDATE SET NULL**
- Если мы удаляем кортеж с ‘Bud’ из Beers
  - то изменить все кортежи в Sells, имеющие beer = ’Bud’, так, чтобы beer = NULL.
- Если мы изменяем кортеж с ‘Bud’, заменяя ’Bud’ на ’Budweiser,
  - то изменить все кортежи в Sells, имеющие beer = ’Bud’, так, чтобы beer = NULL.

Составные части РМД

- **Структурная** (Structure)
  - Как хранятся?
    - Структура данных подсказывает, как данные **организованы**, как они хранения данных в БД.
    - РМД представляет метод, который должен использоваться для *хранения* всех данных в БД.
- **Целостная** (Integrity)
  - Какие **ограничения** накладываются данные?
    - Представляет способ, с помощью которого корректность и качество данных БД может быть обеспечено.
    - РМД представляет метод, который должен использоваться для *обеспечения корректности* данных в БД.
- **Манипуляционная** (Manipulation)
  - Какие операции используются для **манипулирования** данными?
    - Представляет возможности по доступу к данным хранимым в БД.
    - РМД представляет метод, который должен использоваться для *извлечения* всех данных из БД.

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

- 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

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

Целостность значений

- Добавить условие на значение некоторого атрибута
- К объявлению атрибута должно быть добавлено предложение

  CHECK(<условие>)
- в <условие> можно непосредственно указывать имя атрибута, но другие отношения или имена атрибутов могут быть использованы только в подзапросах

Пример

```
CREATE TABLE Sells (
   bar      CHAR(20),
   beer     CHAR(20)      CHECK ( beer IN
            (SELECT name FROM Beers)),
   price    REAL CHECK ( price <= 5.00 )
)
```

Моменты выполнения проверок

- Проверки значений атрибутов выполняются при модификации атрибута или вставке кортежа
  - **CHECK (price <= 150)**
    - проверять каждую новую цену, отклонять ее, если она более 150
  - **CHECK (beer IN (SELECT name FROM Beers))**
    - не проверяется, если сорт пива удаляется из Beers (в отличие от ограничений внешнего ключа)

Целостность кортежей

- CHECK (<условие>) можно указать как отдельный элемент определения отношения
- <условие> может ссылаться на любой атрибут отношения, но атрибуты других отношений могут быть использованы только в подзапросах
- Проверки условия выполняются при модификации или вставке кортежа

Пример: проверки целостности кортежей

- Только бар «Встреча» может продавать пиво дороже 150:

```
CREATE TABLE Sells (
    bar         CHAR(20),
    beer        CHAR(20),
    price       REAL,
    CHECK (bar = 'Встреча' OR
                    price <= 150)
)
```

Утверждения/assertion-ы

- Условия целостности отношений или представлений
- Определяются как :

  CREATE **ASSERTION** <имя>

  CHECK (<условие>);

- <условие> может ссылаться на любое отношение, на любой атрибут БД

Пример: assertion

- Условие на
  - Drinkers(name, addr, phone) и
  - Bars(name, addr, license):
- не может быть баров больше, чем потребителей пива :-)

```
CREATE ASSERTION FewBar CHECK (
   (SELECT COUNT(*) FROM Bars) <=
   (SELECT COUNT(*) FROM Drinkers)
)
```

Моменты выполнения assertion-ов

- В принципе все такие утверждения должны выполняться при любой модификации любого отношения БД
- Искусные СУБД могут оценить, что только определенные изменения могут вызвать нарушение утверждения (assertion-а)
  - Пример:
    никакие изменения в Beers,
    никакие вставки в Drinkers
    не могут нарушить утверждение FewBar

Нормализация базы данных

Формальный механизм создания «хорошей» БД

-устраняются повторяющиеся атрибуты или группы атрибутов
-устраняются атрибуты, зависящие только от части уникального идентификатора.
-устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор
- …

Ненормализованная форма (0NF)

0NF отношения может иметь:

- **Многозначные** атрибуты
  - Некоторые атрибуты имеют более одного значения
- **Составные** атрибуты
  - Некоторые атрибуты состоят из нескольких частей
- **Вложенные** отношения
  - Некоторые атрибуты содержат другие отношения

Пример 0NF

Emp-No	Emp-Name	Dept	Manager	Proj-id	Proj-Start-Date	Location	Weeks-on-Project
005	Smith	Marketing	Jones	A B C	12-93 6-94 09-94	Poole Plymouth Portsmouth	11 15 6
007	Bond	Accounts	Bloggs	B D	06-94 06-94	Plymouth Berlin	3 9
009	King	Info Systems	Hurne	C	09-94	Portsmouth	10
010	Holt	Accounts	Bloggs	A B D	12-93 06-94 06-94	Poole Belfast Hamburg	21 10 12

Proj-id, *Proj-Start-Date*, *Location* и *Weeks-on-Project* составляют группы **значений**

1NF

- “Отношение, в котором в пересечении любых строки и столбца имеется **только одно элементарное** значение”
- Отношение в 1NF не содержит
  - **многозначных** атрибутов
    - где атрибуты имеют более одного значения
  - **составных** атрибутов
    - где атрибуты состоят из нескольких частей
  - **вложенных** отношений
    - где атрибуты содержат другие отношения

Преобразование 0NF в 1NF

“Удаление повторяющихся групп значений”

- Начиная с 0NF отношения, которое содержит...
  - ключ, **идентифицирующий** множества атрибутов
  - и повторяющееся множества атрибутов
- Удалить повторяющиеся атрибуты и поместить их в **отдельное** отношение
- Включить **копию** исходного ключа в новое отношение

1NF –нет групп значений

Emp-No	Emp-Name	Dept	Manager	Proj-id	Proj-Start-Date	Location	Weeks-on-Project
005	Smith	Marketing	Jones	A B C	12-93 6-94 09-94	Poole Plymouth Portsmouth	11 15 6
007	Bond	Accounts	Bloggs	B D	06-94 06-94	Plymouth Berlin	3 9
009	King	Info Systems	Hume	C	09-94	Portsmouth	10
010	Holt	Accounts	Bloggs	A B D	12-93 06-94 06-94	Poole Belfast Hamburg	21 10 12

ключ **повторяющаяся** группа

Proj-id, *Proj-Start-Date*, *Location* и *Weeks-on-Project* составляют группы **значения**

1NF – удаление повторяющейся группы

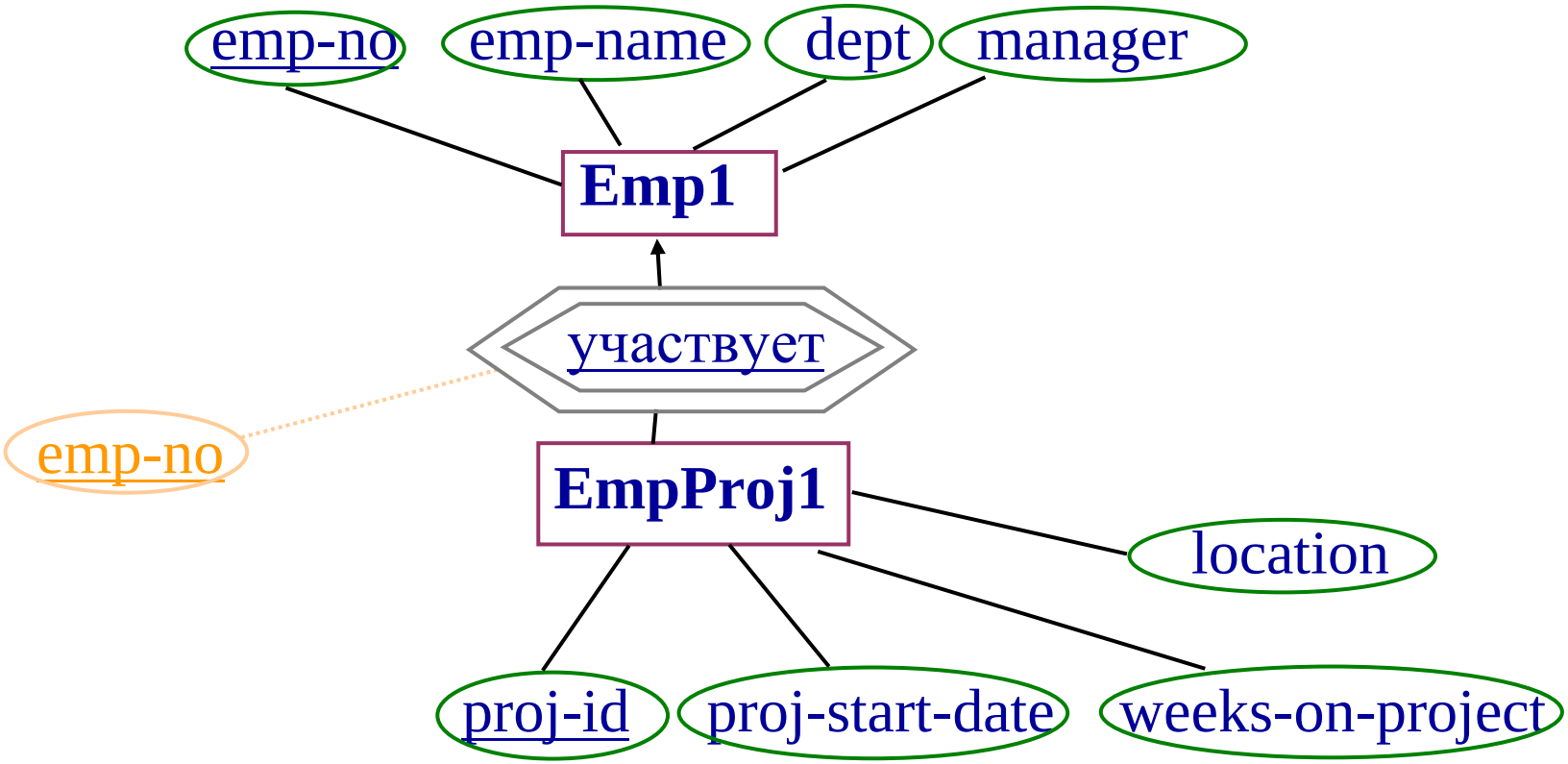
Emp-No	**Emp-Name**	**Dept**	**Manager**
005	Smith	Marketing	Jones
007	Bond	Accounts	Bloggs
009	King	Info Systems	Hume
010	Holt	Accounts	Bloggs

Emp1(emp-no, emp-name, dept, manager)

Emp-No	**Proj-id**	**Proj-Start-Date**	**Location**	**Weeks-on-Project**
005	A	12-93	Poole	11
005	B	6-94	Plymouth	15
005	C	09-94	Portsmouth	6
007	B	06-94	Plymouth	3
007	D	06-94	Berlin	9
009	C	09-94	Portsmouth	10
010	A	12-93	Poole	21
010	B	06-94	Belfast	10
010	D	06-94	Hamburg	12

EmpProj1(emp-no, proj-id, proj-start-date, location, weeks-on-project)

1NF


emp-no
emp-name
dept
manager
Emp1
участвует
emp-no
EmpProj1
location
proj-id
proj-start-date
weeks-on-project

2NF

- Отношение находится в 1NF
- каждый неключевой атрибут **полностью** зависит от первичного ключа
  - не содержит неполных (частичных) зависимостей
- Если ключей несколько, то полностью зависит от каждого из ключей

Частичные зависимости от ключа

- Атрибут зависит от части ключа

  (<u>A, B</u>, C, D)

  $AB \rightarrow C$

  $A \rightarrow D$

  – D частично зависит от ключа (AB)

- (<u>Emp-No, Proj-id</u>, Proj-Start-Date, Location …)

  Emp-No, Proj-id $\rightarrow$ Location

  но

  Proj-id $\rightarrow$ Proj-Start-Date

  зависит от части ключа

2NF – выявление ***частичных зависимостей***

EmpProj1

Emp-No	**Proj-id**	**Proj-Start-Date**	**Location**	**Weeks-on-Project**
005	A	12-93	Poole	11
005	B	6-94	Plymouth	15
005	C	09-94	Portsmouth	6
007	B	06-94	Plymouth	3
007	D	06-94	Berlin	9
009	C	09-94	Portsmouth	10
010	A	12-93	Poole	21
010	B	06-94	Belfast	10
010	D	06-94	Hamburg	12

Proj-Start-Date зависит **от части ключа**

Преобразование 1NF в 2NF

- Обнаружить все частичные зависимости
- Отбросить все отношения одноатрибутным ключом

Emp-No	**Emp-Name**	**Dept**	**Manager**
005	Smith	Marketing	Jones
007	Bond	Accounts	Bloggs
009	King	Info Systems	Hume
010	Holt	Accounts	Bloggs

Emp1

  – отношение **уже в 2NF**

- Если нет, создать новое отношение
  – удалить частично зависимые атрибуты
  – скопировать часть первичного ключа

2NF – удаление частичной зависимости

Emp-No	Proj-id	Location	Weeks-on-Project
005	A	Poole	11
005	B	Plymouth	15
005	C	Portsmouth	6
007	B	Plymouth	3
007	D	Berlin	9
009	C	Portsmouth	10
010	A	Poole	21
010	B	Belfast	10
010	D	Hamburg	12

EmpProj2

Удалили атрибут *proj-start-date*

Emp1(emp-no, emp-name, dept, manager)

EmpProj2(emp-no, proj-id, location, weeks-on-project)

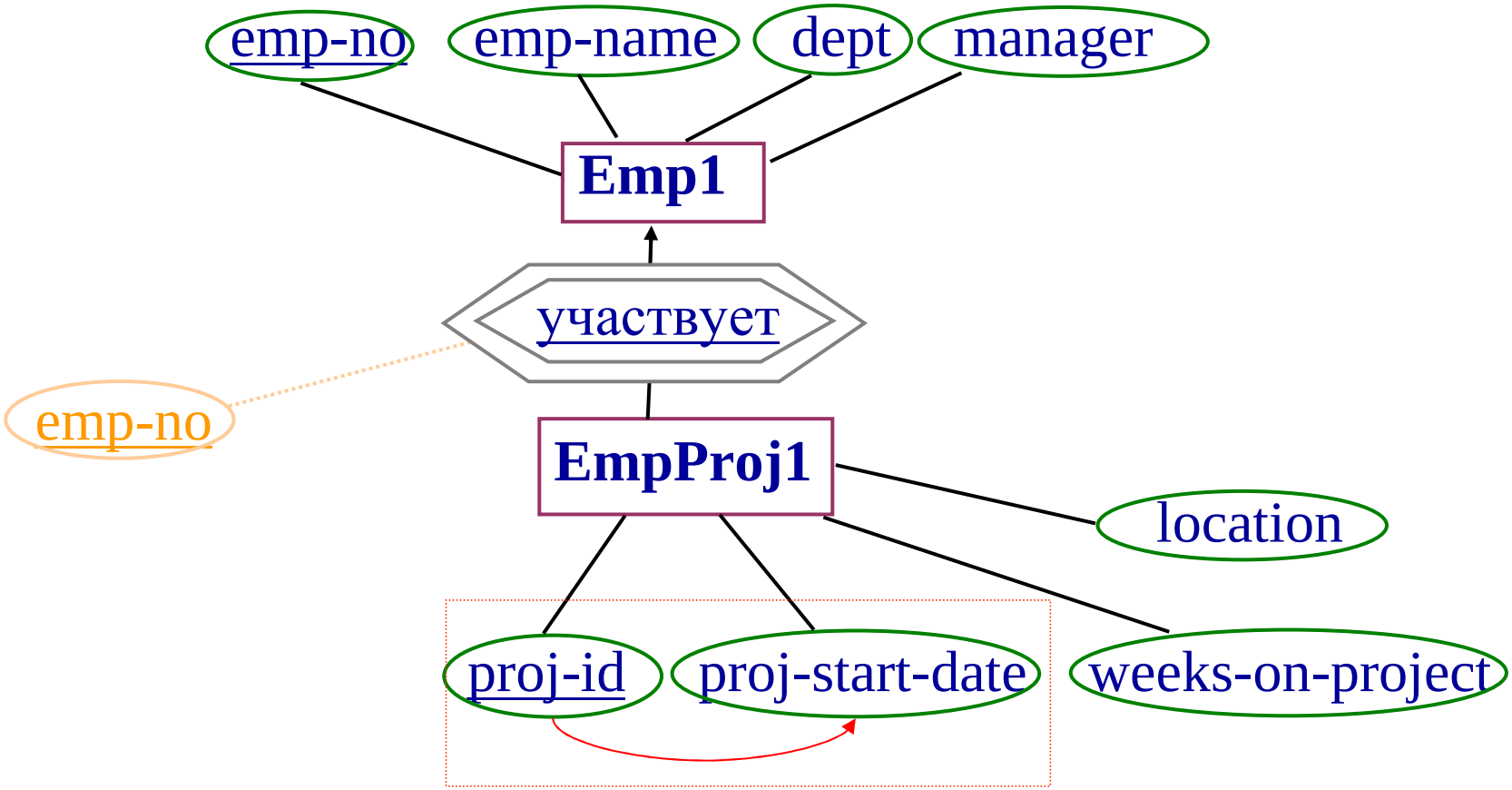
Proj2(proj-id, proj-start-date)

Proj-id	Proj-Start-Date
A	12-93
B	06-94
C	09-94
D	06-94

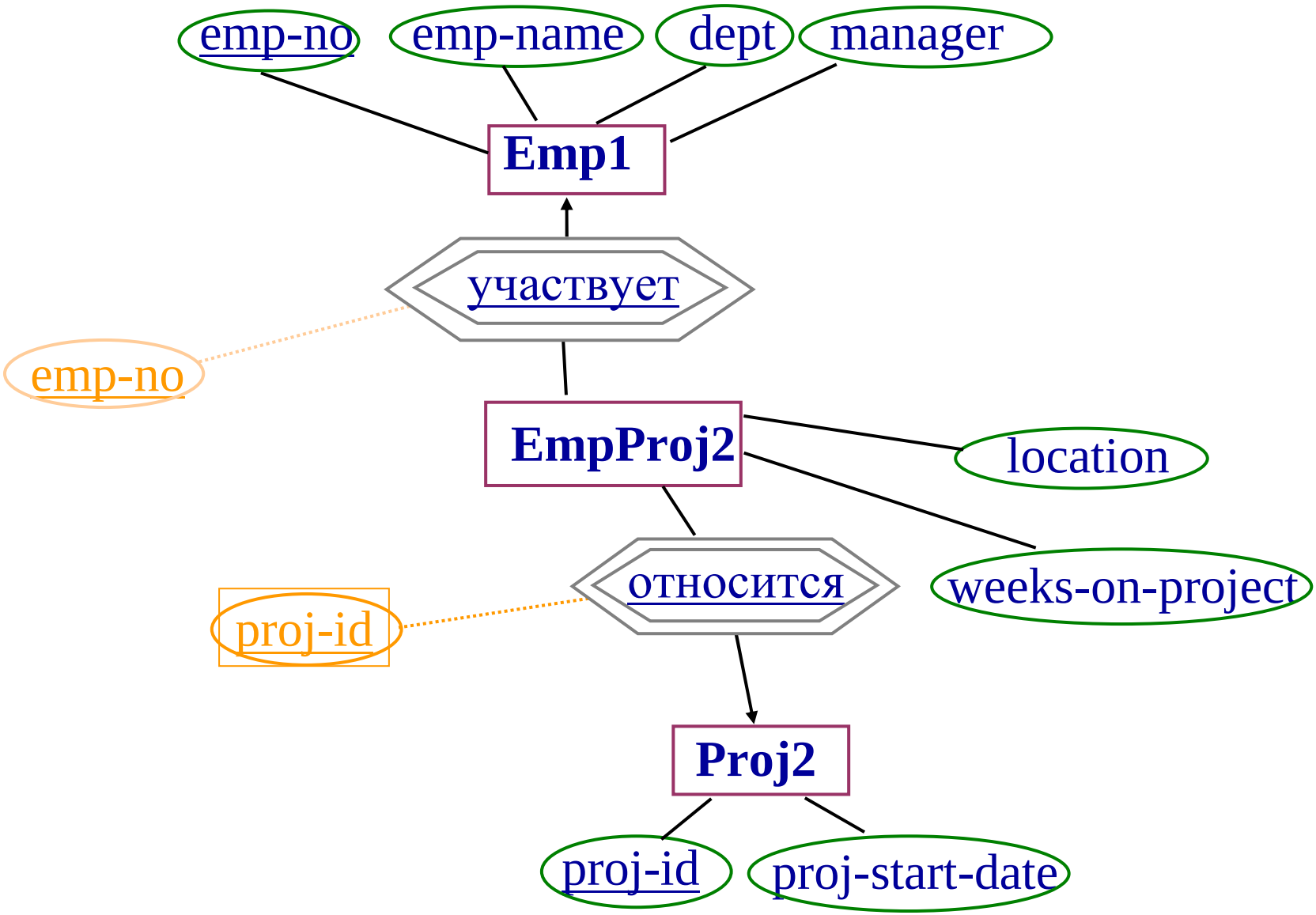
Proj2

Создали отношение с атрибутом *proj-start-date*, скопировав атрибут *proj-id*

1NF


emp-no
emp-name
dept
manager
Emp1
участвует
emp-no
EmpProj1
location
proj-id
proj-start-date
weeks-on-project

2NF


emp-no
emp-name
dept
manager
Emp1
участвует
emp-no
EmpProj2
location
относится
weeks-on-project
proj-id
Proj2
proj-id
proj-start-date

3NF

отношение R в 3-й нормальной форме:
Если в R имеется нетривиальная FD

$$A_1, A_2, ..., A_n \rightarrow B$$

тогда $\{A_1, A_2, ..., A_n\}$ – суперключ R,
или B – часть ключа

- Определение
  - “Отношение во 2NF и каждый неключевой атрибут нетранзитивно зависит от первичного ключа ”
    - от какого-либо ключа
  - “удалить транзитивные зависимости”

Транзитивная зависимость от ключа

- Атрибут зависит от атрибута, отличного от ключа

(A, B, C, D)

$AB \rightarrow C$

$C \rightarrow D$

- D транзитивно зависит от ключа (AB)
  - через C

3NF – выявление транзитивных зависимостей

Emp-No	**Emp-Name**	**Dept**	**Manager**
005	Smith	Marketing	Jones
007	Bond	Accounts	Bloggs
009	King	Info Systems	Hume
010	Holt	Accounts	Bloggs

Emp2

Manager зависит от **неключевого** атрибута

Преобразование 2NF в 3NF

- Обнаружить все транзитивные зависимости
- Отбросить отношения с двумя атрибутами

Proj-id	Proj-Start-Date
A	12-93
B	06-94
C	09-94
B	06-94
D	06-94

Proj2

- Для каждой транзитивной зависимости
  - Создать **новое** отношение
  - Удалить **транзитивно зависимый** атрибут
  - **Скопировать** определяющий атрибут

3NF – удаление транзитивной зависимости

Emp-No	Emp-Name	Dept
005	Smith	Marketing
007	Bond	Accounts
009	King	Info Systems
010	Holt	Accounts

Emp3

Удалили атрибут *manager*

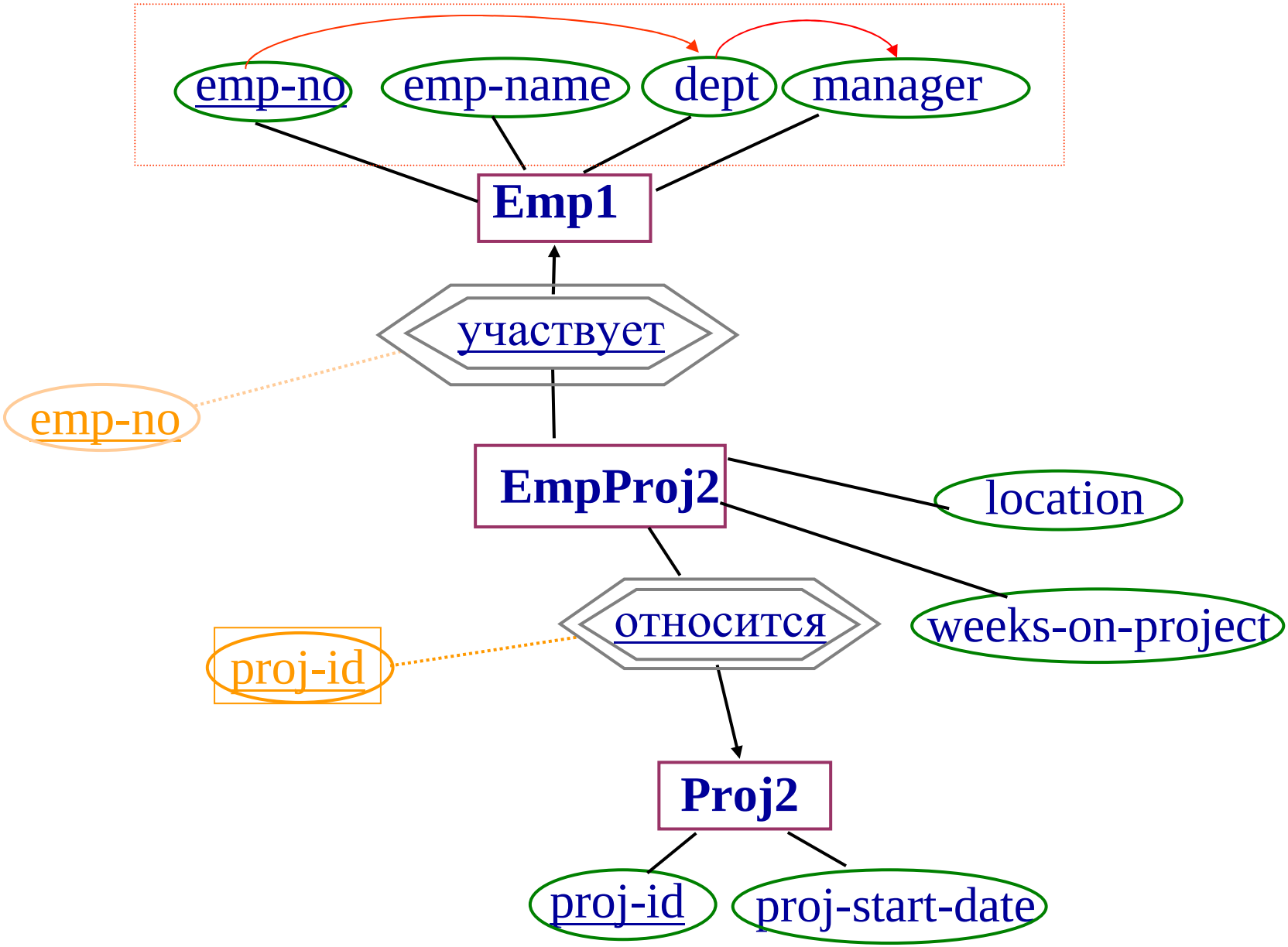
Emp3(emp-no, emp-name, dept)
Dept3(dept, manager)
EmpProj2(emp-no, proj-id, location, weeks-on-project)
Proj2(proj-id, proj-start-date)

Dept	Manager
Marketing	Jones
Accounts	Bloggs
Info Systems	Hume

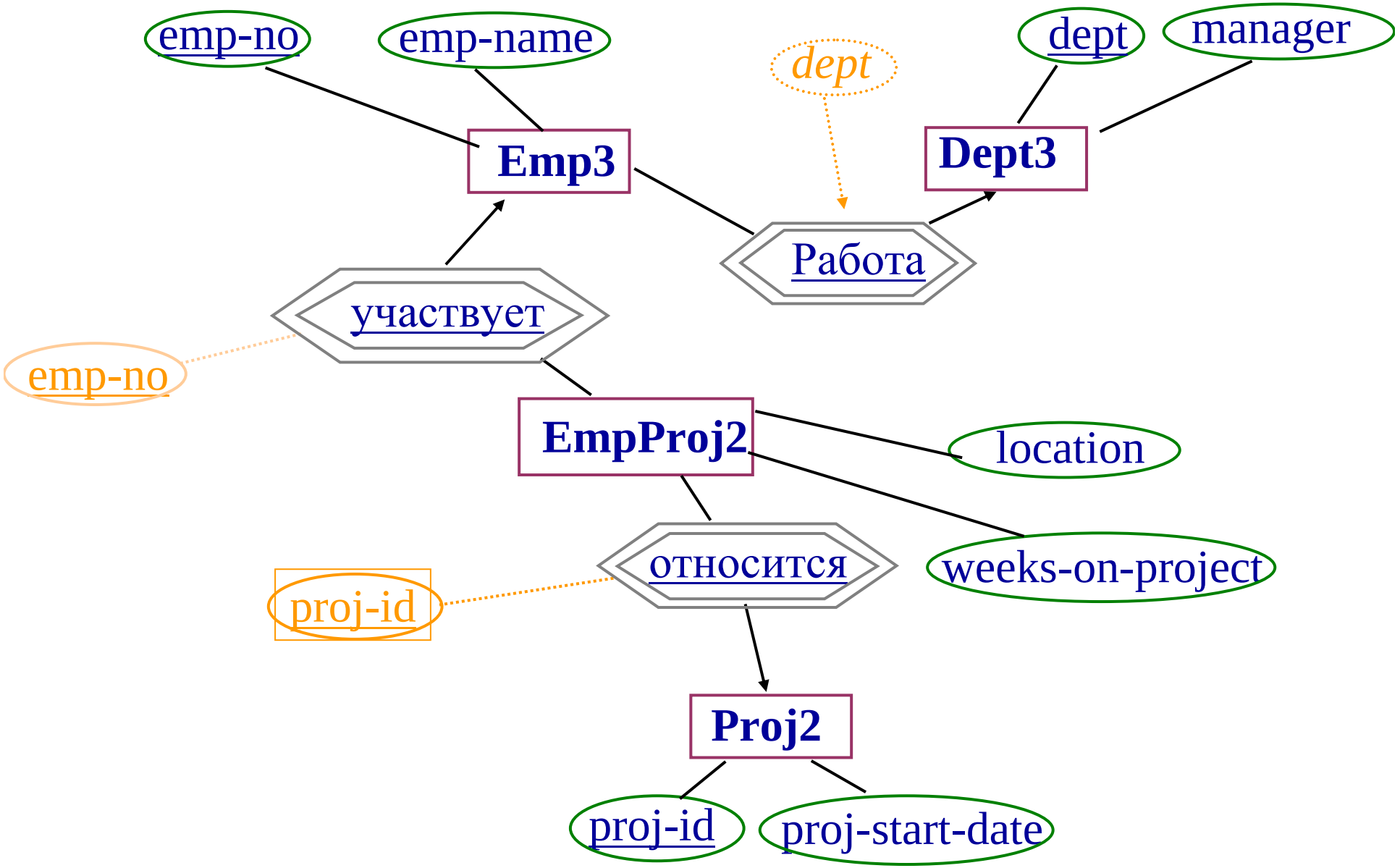
Dept3

Создали отношение с атрибутом *manager* и скопировать атрибут *dept*

2NF


emp-no
emp-name
dept
manager
Emp1
участвует
emp-no
EmpProj2
location
относится
weeks-on-project
proj-id
Proj2
proj-id
proj-start-date

3NF


emp-no
emp-name
dept
dept
manager
Emp3
Dept3
Работа
участвует
emp-no
EmpProj2
location
относится
weeks-on-project
proj-id
Proj2
proj-id
proj-start-date

3NF

Emp3

Emp-No	Emp-Name	Dept
005	Smith	Marketing
007	Bond	Accounts
009	King	Info Systems
010	Holt	Accounts

Proj3

Proj-id	Proj-Start-Date
A	12-93
B	06-94
C	09-94
D	06-94

Dept3

Dept	Manager
Marketing	Jones
Accounts	Bloggs
Info Systems	Hume

EmpProj3

Emp-No	Proj-id	Location	Weeks-on-Project
005	A	Poole	11
005	B	Plymouth	15
005	C	Portsmouth	6
007	B	Plymouth	3
007	D	Berlin	9
009	C	Portsmouth	10
010	A	Poole	21
010	B	Belfast	10
010	D	Hamburg	12

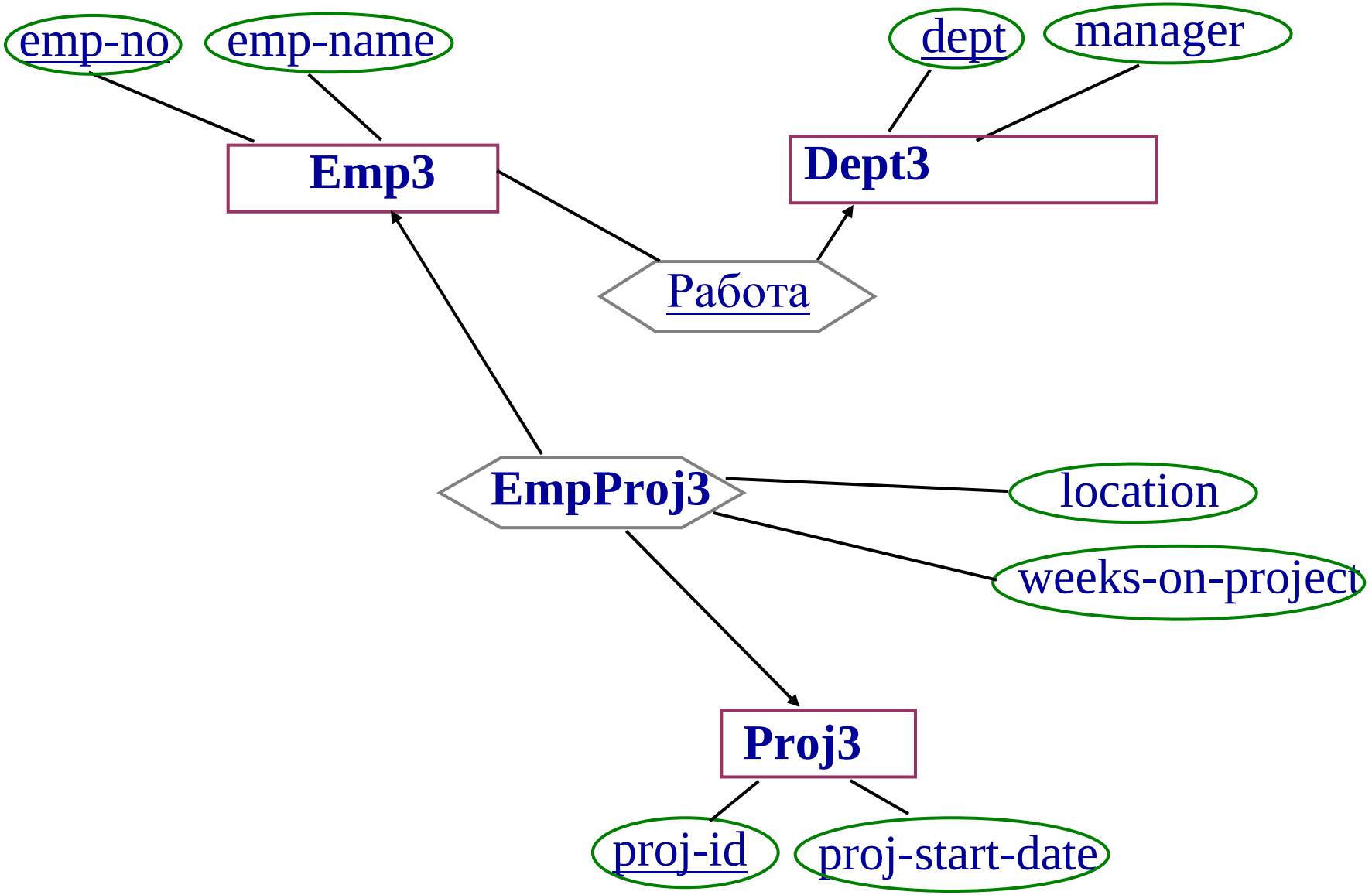
Процесс

1NF – удалили **повторяющиеся** группы

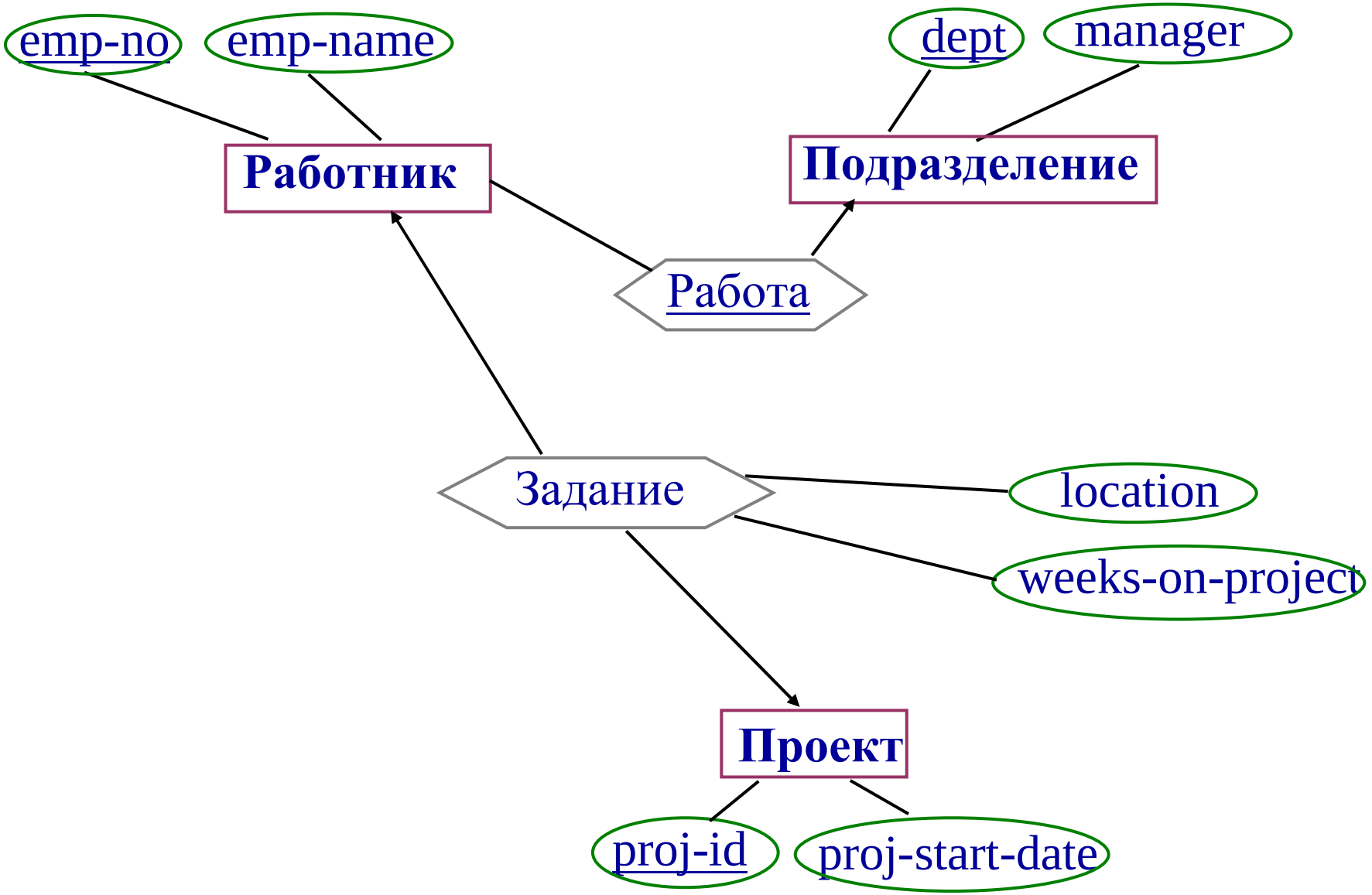
2NF - удалили **частичные** зависимости - декомпозиция

3NF - удалили **транзитивные** зависимости - декомпозиция

3NF


emp-no
emp-name
dept
manager
Emp3
Dept3
Работа
EmpProj3
location
weeks-on-project
Proj3
proj-id
proj-start-date

Концептуальная модель



Декомпозиция отношений

Пусть *R* отношение с атрибутами $A_1, A_2, \ldots A_n$

Создать два отношения *R1* и *R2* с атрибутами

$B_1, B_2, \ldots B_m$ $C_1, C_2, \ldots C_l$

Такими что:

$B_1, B_2, \ldots B_m \cup C_1, C_2, \ldots C_l = A_1, A_2, \ldots A_n$

и

-- *R1* – проекция *R* на $B_1, B_2, \ldots B_m$

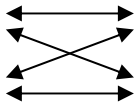
-- *R2* – проекция *R* на $C_1, C_2, \ldots C_l$

Могут быть проблемы

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
DoubleClick	29.99	Camera

Декомпозируем на : **Name, Category** and **Price, Category**

Name	Category
Gizmo	Gadget
OneClick	Camera
DoubleClick	Camera



Price	Category
19.99	Gadget
24.99	Camera
29.99	Camera

Не можем восстановить исходную информацию восстановить полностью и без избыточности

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
OneClick	**29.99**	**Camera**
DoubleClick	**24.99**	**Camera**
DoubleClick	29.99	Camera

Соединения - декомпозиция

- если мы декомпозируем отношение, тогда соединение частей с помощью натурального соединения (natural join), может дать больше кортежей, чем было исходно, получаем ложные кортежей
- цель: использовать декомпозиции, после которых мы не приводят к этому результату

Манипулирование данными

Модификация данных

- Команды модификации не возвращают результат, они соответствующим образом изменяют БД.
- Три вида модификаций:
  1. ***Insert*** вставка кортежей
  2. ***Delete*** удаление кортежей
  3. ***Update*** изменение значения(й) кортежей

Вставка

- Чтобы вставить один кортеж:

  INSERT INTO <отношение>

  VALUES (<список значений>)

- Пример: добавить Likes(drinker, beer) факт, что Алес любит Bud.

```
INSERT INTO Likes
VALUES('Алекс', 'Bud')
```

Указание атрибутов в **INSERT**

- Можно добавить к отношению список атрибутов, потому что
  1. Забыли стандартный порядок столбцов, указанный в **CREATE TABLE**
  2. Не имеет данных для всех атрибутов и хотим, чтобы СУБД заполнила не указанные поля **NULL** значениями или значениями по умолчанию

Вставка нескольких кортежей

- В отношение можно вставить весь результат запроса:

INSERT INTO <relation> **(** <subquery> **)**;

- Используя Frequents(drinker, bar) заполнить новое отношение PotBuddies(name) потенциальными приятелями Тима, то есть посетителями баров, посещаемых Тимой.

Заполнение таблицы

```
INSERT INTO PotBuddies
(SELECT d2.drinker
 FROM Frequents d1, Frequents d2
 WHERE d1.drinker = 'Алекс' AND
   d2.drinker <> 'Алекс' AND
   d1.bar = d2.bar
)
```

Пары посетителей, где первый Алекс, второй кто-то другой, но оба посетители одного и того же бара

Удаление

- Удаление кортежей, удовлетворяющих <условию> из некоторого <отношения>:

DELETE FROM <отношение>
WHERE <условие>

Пример

- Из Likes(drinker, beer) the удалить факт, что Алекс любит Bud:

```
DELETE FROM Likes
WHERE drinker = ‘Алекс’ AND
    beer = ‘Bud’;
```

Пример: удалить несколько кортежей

- Из Beers(name, manf) удалить все сорта пива, для которых имеется другой сорт пива с тем же производителем

```
DELETE FROM Beers b
WHERE EXISTS (
    SELECT name FROM Beers
    WHERE manf = b.manf AND
        name <> b.name);
```

Сорта типа с тем же производителем, но наименованием отличным от наименования пива, представляемого кортежем b

Пример: Удалить все кортежи

- «Очистить» отношение Likes:

```
DELETE FROM Likes;
```

- Не путать с

```
DROP TABLE Likes;
```

Семантика удаления

- Процесс удаления имеет **две стадии**:
  1. Отметить все кортежи, для которых удовлетворяется условие WHERE в исходном отношении
  2. Удалить отмеченные записи

Изменение

- Изменить какой-то атрибут в некотором кортеже отношения:

UPDATE <отношение>
SET <список присваиваний атрибутам>
WHERE <условие на кортежи>

Пример

- Изменить номер телефона Алекса на 555-12-12:

```
UPDATE Drinkers
SET phone = ‘555-12-12’
WHERE name = ‘Алекс’;
```

Пример: изменение нескольких кортежей

- Установить 100 как максимальную цену пива:

```
UPDATE Sells
SET price = 100.00
WHERE price > 100.00;
```

```
update table
set column = expression
[where search-condition]
```

Все