Тема 4.
Инфологическое (концептуальное) моделирование предметной области

Лекция 4

Вопросы лекции:

1. Этапы проектирования БД
2. Концептуальное проектирование базы данных
3. Создание диаграммы «сущность-связь»
4. Пример разработки ER-модели

1. Этапы проектирования БД

Этапы проектирования БД

- Проектирование реляционной базы данных в терминах отношений на основе механизма **нормализации** представляет собой очень сложный и неудобный для проектировщика процесс.

Семантические модели данных

- Потребности проектировщиков баз данных в удобных и мощных средствах моделирования предметной области вызвали к жизни направление **семантических моделей данных**. При том, что любая развитая семантическая модель данных, как и реляционная модель, включает структурную, манипуляционную и целостную части, **главным назначением семантических моделей является обеспечение возможности выражения семантики данных**.

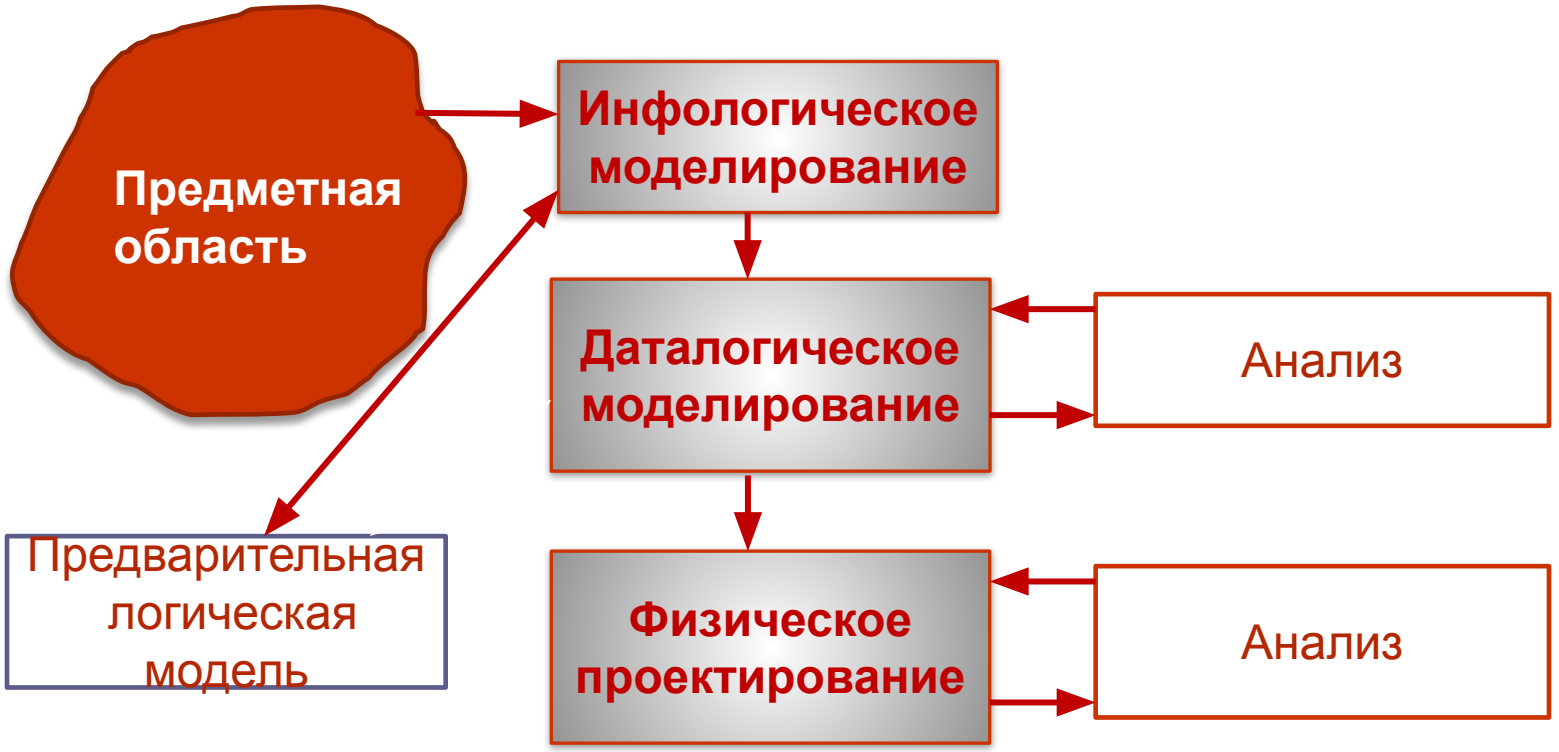
Семантические модели данных

- Семантическое моделирование используется на первой стадии проектирования базы данных. При этом в терминах семантической модели строится **концептуальная схема базы данных**, которая затем автоматически преобразуется к **реляционной** (или какой-либо другой) **схеме**. Этот процесс выполняется под управлением методик, в которых достаточно четко оговорены все этапы.

Методология проектирования

- **Методология проектирования** - структурированный подход, предусматривающий использование специализированных процедур, технических приемов, инструментов, документации и нацеленный на поддержку и упрощение процесса проектирования.
- Методология проектирования предусматривает разбиение всего процесса на несколько **стадий**. На каждой стадии разработчику предлагается набор технических приемов, позволяющих решать задачи, стоящие перед ним на данной стадии разработки. Кроме того, методология предлагает методы планирования, координации, управления, оценки хода разработки проекта, а также структурированный подход к анализу и моделированию всего набора предъявляемых к базе данных требований.

Стадии проектирования БД



Стадии проектирования БД

- **Инфологическое проектирование**

  Инфологическая модель (или семантическая или концептуальная модель) – формализованное представление предметной области (без привязки к СУБД, типам данных, программным средствам и т.п.)

- **Даталогическое проектирование**

  Даталогическая модель – привязка к конкретному типу СУБД (например, реляционной СУБД); Конечная цель – описание структуры БД с учетом особенностей модели данных используемой СУБД.

- **Физическое проектирование** – проектирование физической структуры БД (выборы носителей, определение размеров физических блоков, буферизация и др.)

Стадии проектирования БД

Концептуальное проектирование

- **Концептуальное проектирование базы данных (инфологическое)** - процедура конструирования информационной модели предприятия, не зависящей от каких-либо физических условий реализации.

Стадия концептуального проектирования базы данных начинается с создания концептуальной модели данных предприятия, полностью независимой от любых деталей реализации. К последним относятся: выбранный тип СУБД, состав программ приложения, используемый язык программирования, конкретная вычислительная платформа и любые другие физические особенности реализации.

Логическое проектирование

- **Логическое проектирование базы данных** - процесс конструирования информационной модели предприятия с отображением логических связей между элементами данных безотносительно к среде хранения.

Стадия логического проектирования базы данных заключается в преобразовании концептуальной модели данных в логическую модель данных предприятия **с учетом выбранного типа СУБД** (например, предполагается использование некоторой реляционной СУБД). **Логическая модель данных** является источником информации для физического проектирования. Она предоставляет разработчику физической модели данных средства проведения всестороннего анализа различных аспектов работы с данными, что имеет исключительно важное значение для выбора действительно эффективного проектного решения.

Физическое проектирование

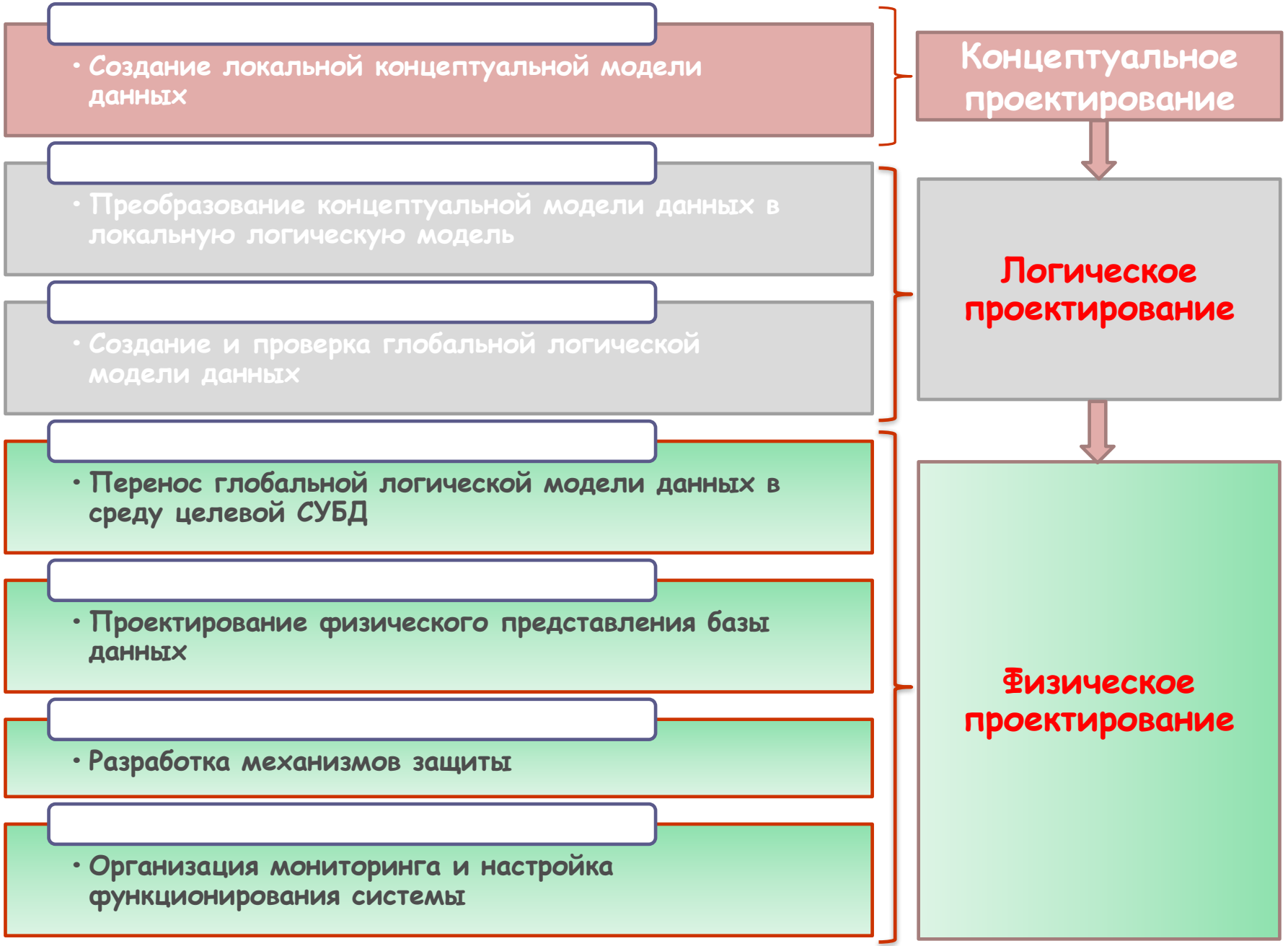
- **Физическое проектирование базы данных** - процесс создания описания конкретной реализации базы данных, размещаемой во внешней памяти. Предусматривает описание структуры хранения данных и методов доступа, предназначенных для осуществления наиболее эффективного доступа к информации.

Стадия **физического проектирования** базы данных предусматривает принятие разработчиком окончательного решения о способах реализации создаваемой базы. Поэтому физическое проектирование обязательно производится с учетом всех особенностей используемой СУБД. Между фазами физического и логического проектирования всегда имеется определенная обратная связь, поскольку решения, принятые на этапе физического проектирования с целью повышения производительности разрабатываемой системы, могут потребовать некоторого пересмотра логической модели данных.

Технологическая сеть проектирования БД

- На основе отдельных технологических операций строится **технологическая сеть проектирования** (ТСП), под которой понимается взаимосвязанная по входам и выходам последовательность тех нологических операций проектирования, выполнение которых приводит к достижению требуемого результата – созданию проекта БД

Технологическая сеть проектирования БД



2. Концептуальное проектирование базы данных

Концептуальное проектирование

Этап 1. Создание локальной концептуальной модели данных на основе представления о предметной области каждого из типов пользователей

- **Цель:** Создание локальной концептуальной модели данных предприятия на основе представления о предметной области каждого отдельного типа пользователей.

Первый этап проектирования базы данных состоит в разработке **концептуальных моделей данных для каждого из существующих типов пользователей** создаваемого приложения. Представление пользователя включает в себя данные, необходимые конкретному пользователю для принятия решений или выполнения не которого задания.

Концептуальное проектирование

- Обычно **представление пользователя** отражает некоторую **функциональную область** в общем поле деятельности предприятия - например, производство, маркетинг, сбыт, управление кадрами или складской учет.
- Пользователь может быть как отдельным работником, так и группой лиц, которые будут непосредственно работать с создаваемым приложением.
- Суть в ТОМ, ЧТО **в любом случае выполнение требуемых пользователю действий должно обеспечиваться создаваемой системой.**

Концептуальная модель

- **Описывает** функциональные и информационные потребности бизнеса
- **Основывается** на текущих потребностях, но может отражать будущие потребности
- **Обращается** к потребностям бизнеса, но не рассматривает их реализацию.

Концептуальное проектирование

- Определить характеристики представлений пользователей можно с помощью различных методов. Например, рекомендуется провести опросы потенциальных пользователей, изучить деловые процедуры, существующие отчеты и формы и/или провести обследование работы предприятия.
- Локальной концептуальной моделью данных будем называть концептуальную модель данных, которая отражает представление о предметной области приложения соответствующего пользователя.

Концептуальное проектирование

- Каждая локальная концептуальная модель данных включает следующее:
  - ✔ типы сущностей
  - ✔ типы связей
  - ✔ атрибуты
  - ✔ домены атрибутов
  - ✔ потенциальные ключи
  - ✔ первичные ключи.

Концептуальное проектирование базы данных

Этап 1. Создание локальной концептуальной модели данных исходя из представлений о предметной области каждого из типов пользователей.

- **Этап1.1.** Определение типов сущностей.
- **Этап1.2.** Определение типов связей.
- **Этап1.3.** Определение атрибутов и связывание их с типами сущностей и связей.
- **Этап1.4.** Определение доменов атрибутов.
- **Этап1.5.** Определение атрибутов, являющихся потенциальными и первичными ключами.
- **Этап1.6.** Создание диаграммы "сущность-связь".
- **Этап1.7.** Обсуждение локальных концептуальных моделей данных с конечными пользователями.

Этап 1.1. Определение типов сущностей

- **Цель:** Определение основных типов сущностей, присутствующих в представлении данного пользователя о предметной области приложения.

Первый этап в построении локальной концептуальной модели данных состоит в **определении основных объектов**, которые могут интересовать пользователя. Эти объекты являются **типами сущностей**, входящих в модель.

Этап 1.1. Определение типов сущностей

Один из методов идентификации сущностей состоит в **изучении спецификаций** по выполнению конкретных функций пользователя на данном предприятии. Из этих спецификаций следует извлечь все используемые в них **существительные или сочетания существительного и прилагательного** (например, "личный номер", "фамилия работника", "номер объекта недвижимости", "адрес объекта недвижимости", "арендная плата", "количество комнат").

Этап 1.1. Определение типов сущностей

Затем среди них выбираются **самые крупные объекты** (люди, города) или представляющие интерес концепции и исключаются все существительные, которые просто определяют другие объекты.

Например, свойства "личный номер" и "фамилия работника" могут быть объединены в сводном объекте под названием "работник", тогда как свойства "номер объекта недвижимости", "адрес объекта недвижимости", "арендная плата" и "количество комнат" можно объединить в сущности под названием "объект недвижимости".

Этап 1.1. Определение типов сущностей

Альтернативный способ идентификации сущностей состоит в **поиске объектов, которые существуют независимо от других.**
Например, объект "работник" (staff) безусловно является сущностью.

1.1 Определение типов сущностей

В некоторых случаях выделение сущностей бывает затруднено из-за способа, посредством которого они представлены в спецификациях. Зачастую пользователи, излагая свои мысли, используют **примеры или аналогии**. Вместо того чтобы вести разговор о некотором обобщенном работнике, они могут просто упомянуть одно или несколько имен. Бывает также, что пользователи заменяют имена работников или название предприятия выполняемыми ими обязанностями или оказываемыми услугами. В этом случае они могут упоминать либо должность работника, либо выполняемые им функции - например, "руководитель", "ответственный", "контролер" или "заместитель"…

Этап 1.1. Определение типов сущностей

Пользователи часто используют синонимы или омонимы. **Синонимами** называются слова, сходные по смыслу, но различные по звучанию и написанию, - например, "**отделение**" и "**филиал**". **Омонимы** это слова, одинаковые по написанию и звучанию, но имеющие различные смысловые значения, причем реальное значение в каждом конкретном случае можно установить только по контексту. Так, слово "**программа**" может обозначать курс обучения, предстоящую серию последовательных событий, план предстоящей работы и даже последовательность телепередач.

Этап 1.2. Определение типов связей

- **Цель:** Определение важнейших типов связей, существующих между сущностями, выделенными на предыдущем этапе.

После выделения сущностей следующим этапом разработки будет установление всех существующих между ними связей. Одним из методов определения сущностей является выборка всех существительных, присутствующих в спецификациях на проект. Аналогичный подход можно использовать и при определении существующих связей, однако в этом случае выбираются все выражения, в которых содержатся глаголы.

Например:

- ✔ Подразделение имеет персонал.
- ✔ Персонал занимается объектами недвижимости.
- ✔ Арендатор просматривает сведения об объектах недвижимости, сдаваемых в аренду.

Этап 1.2. Определение типов связей.

Следует выделить только те связи между сущностями, которые необходимы для удовлетворения требований к проекту. Так, в предыдущем примере были выделены связи "персонал занимается объектами недвижимости" и "арендатор просматривает сведения об объектах недвижимости, сдаваемых в аренду". Может возникнуть желание включить в модель и связь между персоналом и арендатором (например, "персонал обслуживает арендатора"). Хотя эта связь является вполне допустимой, НО, если в спецификациях на проект нет ни одного указания на то, что она должна быть отображена в модели, её явно обозначать не следует (она, например, будет осуществляться через объекты недвижимости).

Этап 1.2. Определение типов связей.

В большинстве случаев связи являются парными - другими словами, связи **существуют только между двумя сущностями.**

Однако, следует проявлять осторожность и тщательно проверять наличие в проекте комплексных связи, объединяющих более двух сущностей различных типов, а также рекурсивных связей, существующих между сущностями одного и того же типа.

Особое внимание следует уделять проверке того, были ли выделены все связи, явно или неявно присутствующее в спецификациях на проект.

Этап 1.2. Определение типов связей.

В принципе, каждую из возможных пар сущностей было бы полезно проверить на наличие между ними некоторой связи, однако в крупных системах, включающих сотни типов сущностей, эта задача может оказаться чрезвычайно трудоемкой.

Но вообще отказываться от выполнения подобных проверок неразумно, к тому же ответственность за печальные последствия этого отказа придется нести как аналитикам, так и проектировщикам.

Так или иначе, все пропущенные связи будут обязательно выявлены позже, при проведении проверки возможности выполнения транзакций, необходимых пользователям (Этап 2.4).

Этап 1.2. Определение типов связей

Определение кардинальности связей и ограничений, накладываемых на его участников. Установив связи, которые будут иметь место в создаваемой модели, необходимо определить кардинальность каждой из них. Каждая связь может иметь кардинальность либо "один к одному" (1:1), либо "один ко многим" (1:M), либо "многие ко многим" (M:M). Если известны конкретные значения кардинальности или хотя бы верхний или нижний предел этих значений, то эту информацию обязательно нужно зафиксировать в документации.

Этап 1.3. Определение атрибутов и связывание

- **Цель:** Связывание атрибутов с соответствующими типами сущностей или связей.

На этом этапе предлагаемой методологии необходимо выявить все данные, описывающие сущности и связи, выделенные в создаваемой модели базы данных.

Воспользуемся тем же методом, который применялся нами для идентификации сущностей: выберем все существительные и содержащие их фразы, присутствующие в спецификациях на проект. Выбранное существительное представляет атрибут в том случае, если оно описывает свойство, качество, идентификатор или характеристику некоторой сущности или связи.

Этап 1.3. Определение атрибутов и связывание

Важно отметить, что каждый атрибут может быть либо простым, либо составным. **Составные атрибуты** представляют собой **набор простых атрибутов**. Например, атрибут "Адрес" может быть простым и представлять все элементы адреса как единое значение: "115 Durnbarton Road, Partick, Glasgow, G11 6YG". В другом варианте этот же атрибут может быть представлен как составной, т.е. состоящий из серии простых атрибутов, содержащих различные элементы адреса. В этом случае то же самое значение может быть разделено на такие атрибуты, как "Улица" (115 Dumbarton Road), "Район" (Partick), "Город" (Glasgow) и "Почтовый код" (Gll 6YG).

Этап 1.3. Определение атрибутов и связывание

Выбор способа представления адреса в виде простого или составного атрибута определяется требованиями, предъявляемыми к приложению пользователем. Если пользователь не нуждается в доступе к отдельным элементам адреса, то его целесообразно представить как простой атрибут. Но, если пользователю потребуется независимый доступ к отдельным элементам адреса, то атрибут "Адрес" следует сделать составным, образованным из необходимого количества простых атрибутов (например, для проверки по справочникам и унификации написания).

На данном этапе важно идентифицировать все простые атрибуты, которые должны быть представлены в концептуальной модели базы данных, включая и те, которые впоследствии будут использованы для создания составных атрибутов.

Этап 1.3. Определение атрибутов и связывание

Атрибуты, значения которых могут быть установлены с помощью значений других атрибутов, называются **производными**, или **вычисляемыми**.

Примерами производных атрибутов являются следующие:

- ✔ количество работников данного отделения предприятия;
- ✔ возраст работника;
- ✔ общая сумма зарплаты всего персонала данного отделения предприятия;
- ✔ количество объектов недвижимости, которыми занимается персонал данного отделения предприятия.

Этап 1.3 Определение атрибутов и связывание

Очень часто подобные атрибуты вообще не отображаются в концептуальной модели данных. Однако в некоторых случаях может иметь место риск удаления или модификации атрибута или атрибутов, значения которых используются для вычисления значения производного атрибута. В этом случае **производный атрибут должен быть представлен в модели данных, что позволит предупредить нежелательную потерю информации.** Однако, если производный атрибут показан в модели данных, следует непременно указать, что он является именно производным. Способ представления производных атрибутов устанавливается на этапе физического проектирования базы данных.

Этап 1.4. Определение доменов атрибутов

- **Цель** Определение доменов для всех атрибутов, присутствующих в каждой локальной концептуальной модели данных.

Задача этого этапа построения локальной концептуальной модели данных состоит в определении доменов атрибутов для всех атрибутов, присутствующих в модели.

- **Доменом** называется некоторый пул значений, элементы которого выбираются для присвоения значений одному или более атрибутам.

Этап 1.4. Определение доменов атрибутов

- Примеры доменов

✔ **Домен атрибута, включающий допустимые номера отделений предприятия.** Он состоит из трехсимвольных строк, в которых первый символ является буквой, а остальных два - цифрами, задающими числа в диапазоне 1-99.

✔ **Домен атрибута «Пол».** Допустимыми значениями для атрибута "Пол" сущности "Работник" являются "М" и "Ж". Домен этого атрибута состоит из двух строк длиной в один символ, имеющих указанные значения.

Этап 1.5. Определение первичных ключей

- **Цель:** Определение всех потенциальных ключей для каждого типа сущности и, если таких ключей окажется несколько, выбор среди них первичного ключа.

На этом этапе для каждой сущности устанавливается **потенциальный ключ** (или ключи), после чего осуществляется выбор **первичного ключа.**

- **Потенциальным ключом** называется атрибут или минимальный набор атрибутов заданной сущности, позволяющий уникальным образом идентифицировать каждый ее экземпляр.

Для некоторых сущностей возможно наличие нескольких потенциальных ключей. В этом случае среди них нужно выбрать один ключ, который будет называться первичным ключом. Все остальные потенциальные ключи будут называться **альтернативными ключами.**

1.5 Определение первичных ключей

При выборе первичного ключа среди нескольких потенциальных руководствуйтесь приведенными ниже рекомендациями.

- ✔ Используйте **потенциальный ключ с минимальным набором атрибутов.**
- ✔ Используйте тот потенциальный ключ, **вероятность изменения значений которого минимальна.**
- ✔ Выбирайте тот потенциальный ключ, который имеет **минимальную вероятность потери уникальности значений в будущем.**
- ✔ Используйте потенциальный ключ, значения которого имеют **минимальную длину** (в случае текстовых атрибутов).
- ✔ Остановите свой выбор на **потенциальном ключе, с которым будет проще всего работать** (с точки зрения пользователя).

Этап 1.6. Создание диаграммы «сущность-связь»

- Большинство современных подходов к проектированию баз данных (главным образом, реляционных) основано на использовании разновидностей **ER-модели**. Рассмотрим основные понятия ER-модели.

3. Создание диаграммы «сущность-связь» (ER-модели)

Моделирование структуры базы данных при помощи **алгоритма нормализации** имеет серьезные недостатки:

- Первоначальное размещение всех атрибутов **в одном отношении** является очень неестественной операцией. Интуитивно разработчик сразу проектирует несколько отношений в соответствии с обнаруженными сущностями.
- **Невозможно** сразу **определить полный список атрибутов.** Пользователи имеют привычку называть разными именами одни и те же вещи или наоборот, называть одними именами разные вещи.
- Для проведения процедуры нормализации **необходимо выделить зависимости атрибутов**, что тоже очень нелегко, т.к. необходимо явно выписать все зависимости, даже те, которые являются очевидными.

Семантическое моделирование БД

- В реальном проектировании структуры базы данных применяются метод **семантического моделирования.** Семантическое моделирование представляет собой моделирование структуры данных, опираясь на **смысл этих данных.** В качестве инструмента семантического моделирования используются различные варианты диаграмм **сущность-связь** (ER - Entity-Relationship).

Модель Entity-Relationship (Сущность-Связи)

Модель сущность-связь (ER-модель) (англ. entity-relationship model, ERM) — модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит **преобразование ER-модели в конкретную схему базы данных** на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

Основные понятия ER-модели Entity-Relationship (Сущность-Связи)

На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Модель была предложена **Питером Ченом** (Chen) в 1976 г. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в системах CASE, поддерживающих автоматизированное проектирование реляционных баз данных.

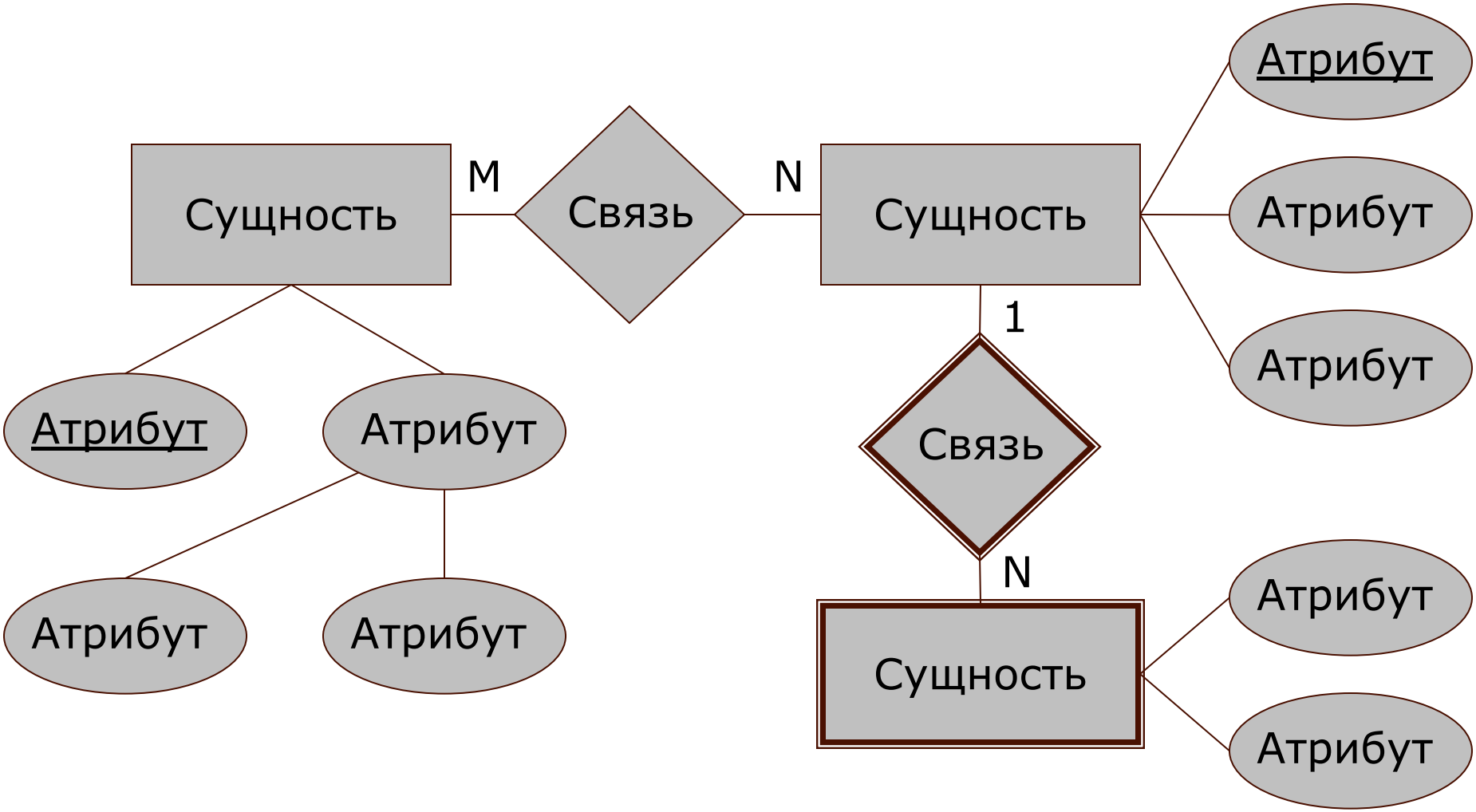
Нотации (графические диаграммы), используемые для визуализации ER-моделей:

- Нотация Питера Чена
- Crow's Foot
- IDEF1X

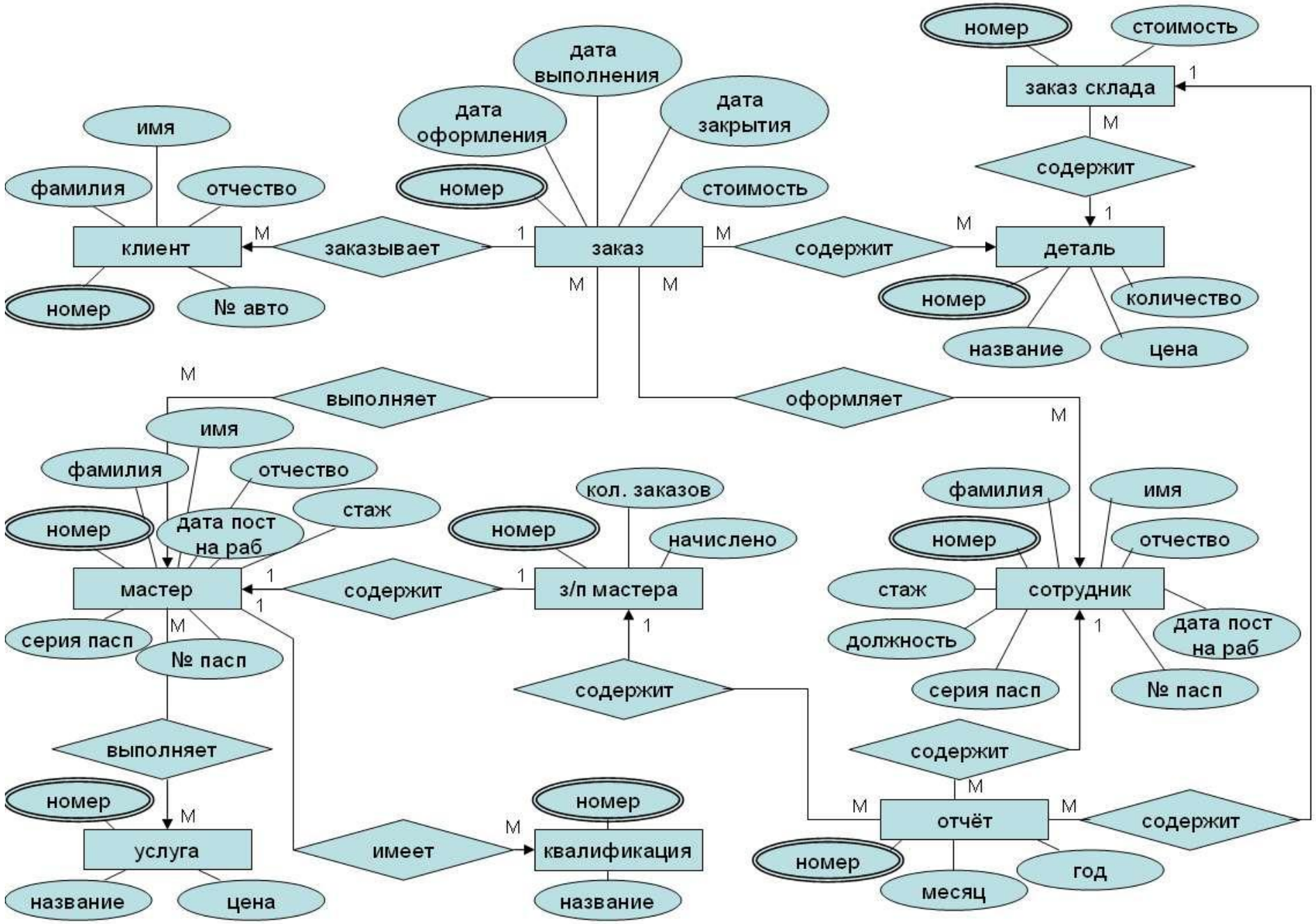
Диаграмма «Сущность-связь» в нотации Питера Чена

Множества **сущностей** изображаются в виде **прямоугольников**, множества отношений изображаются в виде **ромбов**. Если сущность участвует в отношении, они связаны линией. Если отношение не является обязательным, то линия пунктирная. **Атрибуты** изображаются в виде **овалов** и связываются линией с одним отношением или с одной сущностью.

Диаграмма «Сущность-связь» в нотации Питера Чена (метамодель)



Пример диаграммы «Сущность-связь» в нотации Чена

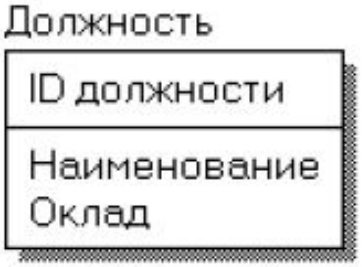


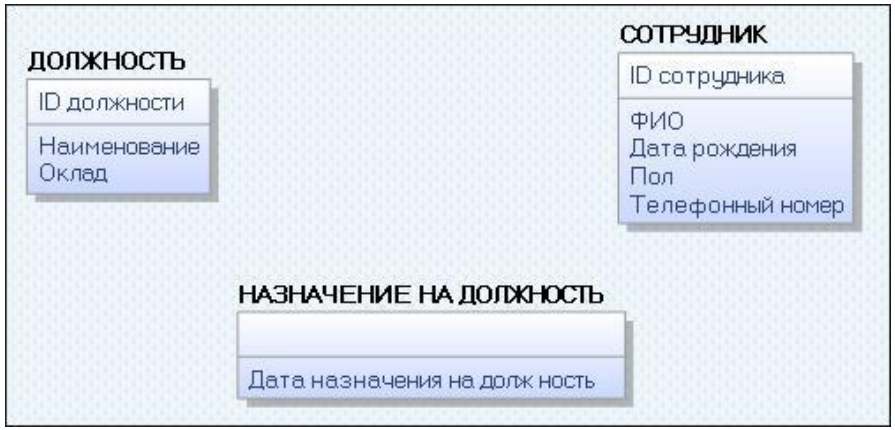
Основные понятия модели Entity-Relationship (Сущность-Связи)

- Основными понятиями ER-модели являются **сущность, связь и атрибут.**
- **Сущность - это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.** В диаграммах ER-модели сущность представляется в виде **прямоугольника**, содержащего **имя сущности**. При этом имя сущности - это имя типа, а не некоторого конкретного экземпляра этого типа.
- Каждый **экземпляр сущности** должен быть отличим от любого другого экземпляра той же сущности.

Сущность, атрибут

- **Сущность** – это объект, который может быть идентифицирован некоторым способом, отличающим его от других объектов. Каждая сущность обладает набором атрибутов.
- **Атрибут** - отдельная характеристика сущности.
- **Сущность состоит из экземпляров**, каждый из которых должен отличаться от другого экземпляра. Пример: сущность – «Город», экземпляры сущности «Город» – Москва, Пенза,





Сущности с атрибутами

Нотация IDEF1X

- Сущность обладает одним или несколькими **атрибутами**, которые являются либо **собственными** для сущности, либо наследуются через другое отношение (от **PK «родителя»** передается **FK** в **«сущность-потомок»**).
- Атрибуты однозначно идентифицируют каждый экземпляр сущности.
- Каждый атрибут идентифицируется уникальным именем.
- Атрибуты изображаются в виде списка их имен внутри блока ассоциированной сущности, причем каждый атрибут занимают отдельную строку.
- Определяющие первичный ключ атрибуты размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

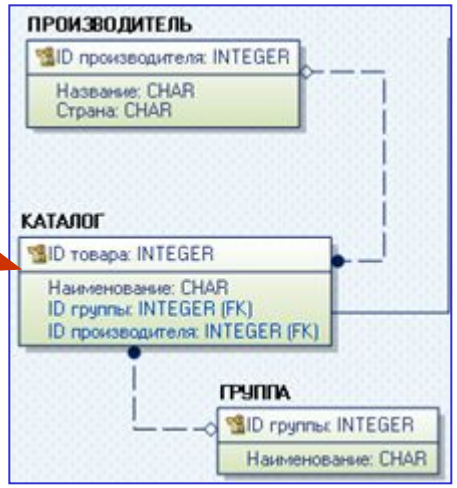
Ключ

- **Ключ** - минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.
- **Первичный ключ** сущности позволяет идентифицировать ее экземпляры, а внешний – экземпляры сущности, которая находится в связи с данной сущностью.

Типы сущностей

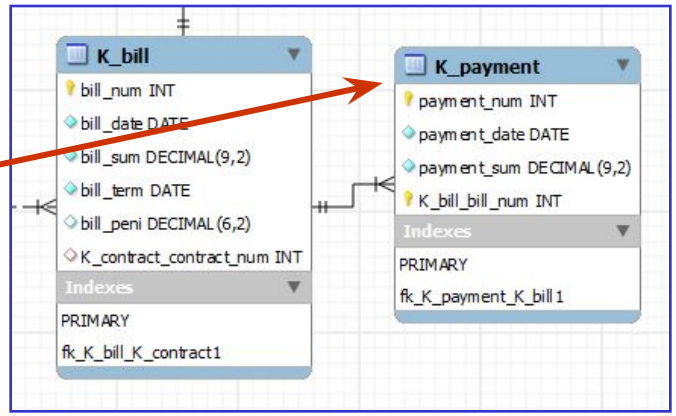
Независимая сущность

Для определения экземпляра сущности нет необходимости ссылаться на другие сущности.

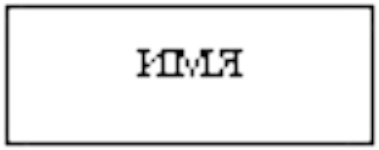


Зависимая сущность

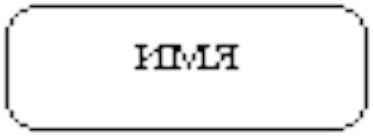
Для определения экземпляра такой сущности необходимо сослаться на экземпляр независимой сущности, с которой связана зависимая сущность.



Обозначение сущностей в нотации IDEF1X



Независимая сущность



Зависимая сущность

- Сущность является **"независимой"**, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Пример: «Сотрудник».
- Сущность называется **"зависимой"**, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Пример: «Участие в проектах».

Связь

Связь - это логическая ассоциация, устанавливаемая между сущностями.

Связь определяет **количество экземпляров в связанной сущности, с которыми ассоциирован экземпляр рассматриваемой сущности.**

Связи бывают следующих типов:

- **один к одному**
- **один ко многим**

Примеры:

- Связь один к одному:

  «Страна» - «Столица»

- Связь один ко многим:

  «Группа» - «Студент»

- Связь многие ко многим:

  «Сотрудник» - «Проект»

Связь «один-ко-многим»: Отделы – Сотрудники

Таблица «Сотрудники» (Emp)

Табельный номер	*ФИО сотрудника*	*Отдел*
023	Волкова Елена Павловна	2
113	Белов Сергей Юрьевич	1
101	Рогов Сергей Михайлович	2
056	Панина Анна Алексеевна	1
…	…	…
098	Фролов Юрий Вадимович	9

Таблица «Отделы» (Departs)

Номер отдела	*Название отдела*
1	Информационный отдел
2	Администрация
3	Отдел кадров
…	…
9	Проектный отдел

«Номер отдела» – первичный ключ в таблице «Отделы»

«Отдел» – внешний ключ в таблице «Сотрудники» к таблице «Отделы»

Связь «многие-со-многими»: Сотрудники- Проекты

- Связи "many-to-many". Иногда бывает необходимо связывать сущности таким образом, что с обоих концов связи могут присутствовать несколько экземпляров сущности. Например, сотрудники консалтинговой компании участвуют в проектах. При этом один сотрудник может участвовать в нескольких проектах и в одном проекте могут участвовать несколько сотрудников. Для этого вводится разновидность связи **"многие-со-многими"**.
- Оформляются через «развязочные таблицы», например: «участие в проектах» (сущность из двух атрибутов: код сотрудника (FK), код проекта (FK)).

Связь «многие-со-многими»: Сотрудники- Проекты

Таблица «Сотрудники»

ФИО	*Номер*
Волкова Е.П.	023
Белов С.Ю.	113
Рогов С.М.	101
Панина А.А.	056
Фролов Ю.В.	098
...	...

Таблица «Участие»

Участник	*Роль*	*Проект*
113	исполнитель	23/Н
101	руководитель	18-К
056	исполнитель	18-К
101	консультант	09/Р
098	руководитель	23/Н
...	...	...

Таблица «Проекты»

Шифр	*Название проекта*
23/Н	АИС "Налог"-2
18-К	ИПС "Жители"
09/Р	ГИС "Город"

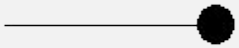
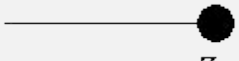
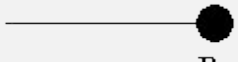
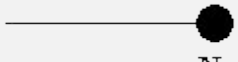
В таблице «Участие»:

«Участник» – внешний ключ к таблице «Сотрудники»

«Проект» – внешний ключ к таблице «Проекты»

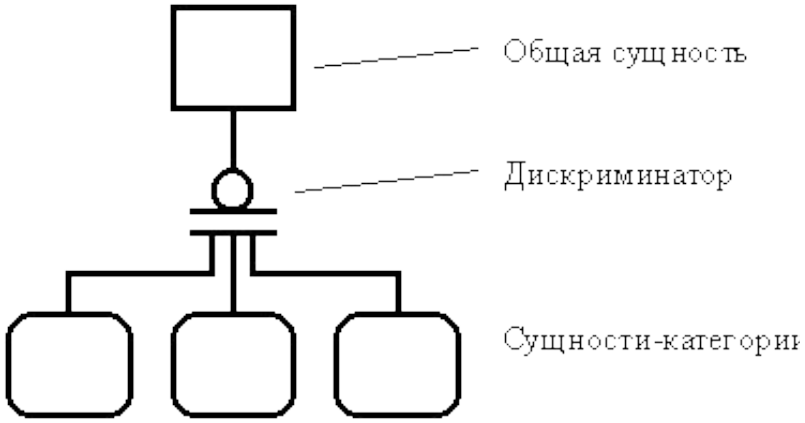
Виды связей

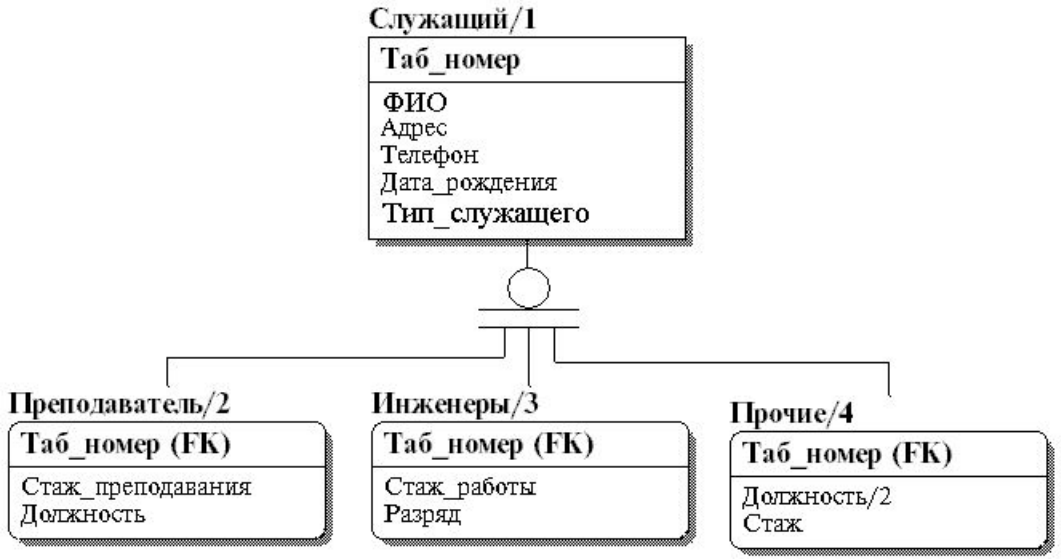
Отношение дополнительно определяется с помощью указания мощности: какое количество экземпляров сущности-потомка может существовать для сущности-родителя.

	1, 1
	0, M
	0, 1
	1, M
	точно N (N-произвольное число)

Виды связей

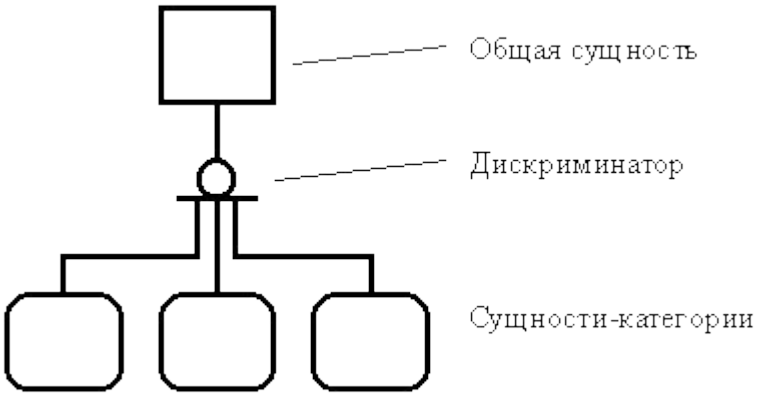
В IDEF1X вводится понятие "отношение категоризации", по смыслу эквивалентное иерархической связи. Отношение полной категоризации (сущности-категории составляют полное множество потомков родительской сущности) обозначается.

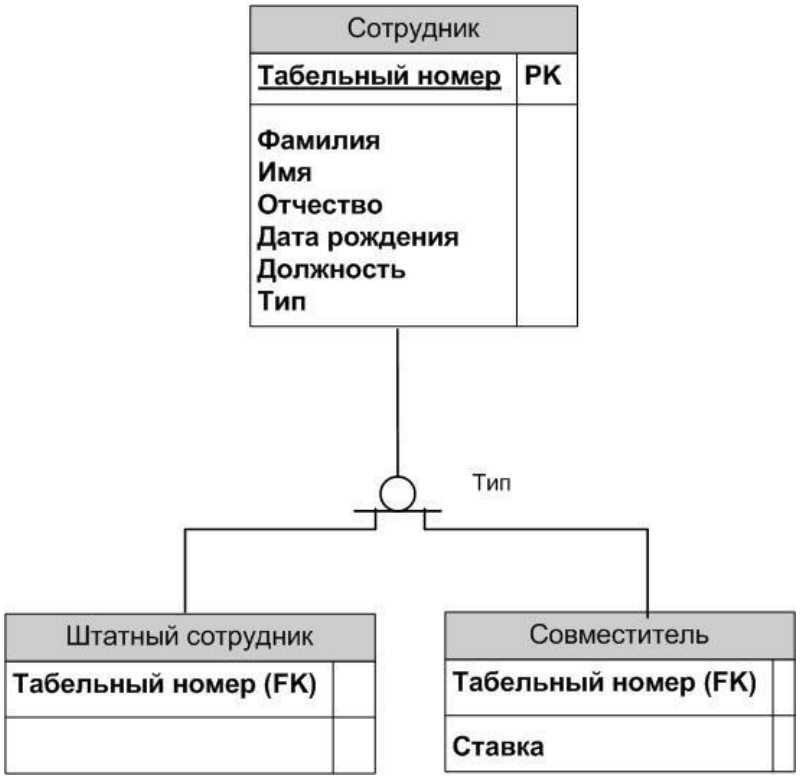




Виды связей

В IDEF1X также может существовать отношение неполной категоризации (сущности-категории составляют неполное множество потомков общей сущности):





Виды связей

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Элемент диаграммы	Обозначает
----------	неидентифицирующая связь
_____	идентифицирующая связь

Виды связей

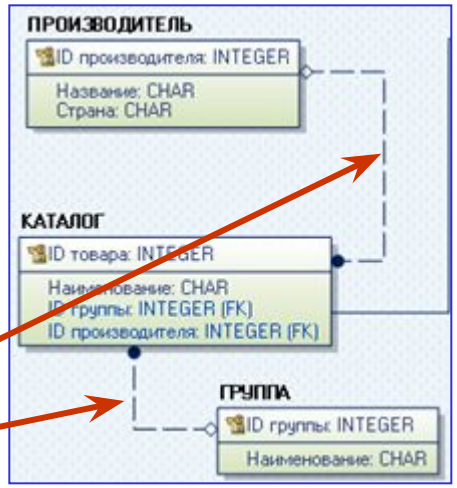
При **идентифицирующей связи** миграция ключевых атрибутов родительской сущности (сущности со стороны связи 1 осуществляется всегда **в область ключевых атрибутов дочерней сущности** (сущности со стороны связи M). **Мигрирующие ключи** помечены символами (FK) – это аббревиатура слов **Foreign Key** (чуждый, посторонний ключ). Идентифицирующая связь между сущностями указывается тогда, когда экземпляр дочерней сущности не может существовать без экземпляра родительской сущности. При этом ключ связи FK **(внешний ключ, ссылающийся на родительскую сущность)** в составе дочерней сущности не может принимать значение **Null**. Можно сказать, что идентифицирующая связь является особым случаем **обязательной связи.**

Виды связей

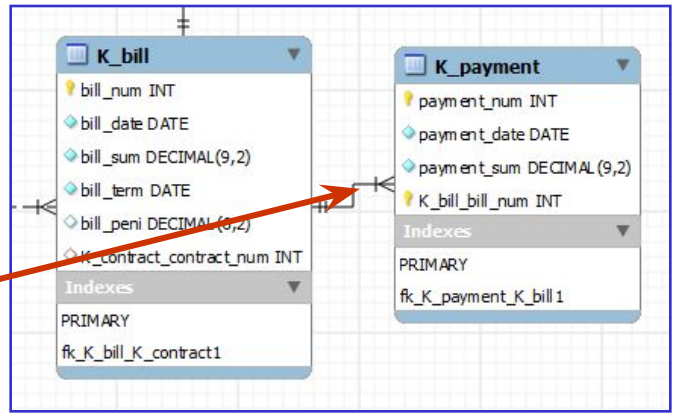
- При **неидентифицирующей связи** миграция ключевых атрибутов родительской сущности всегда осуществляется в область **неключевых атрибутов дочерней сущности**. При этом ключ связи **FK** (внешний ключ, ссылающийся на родительскую сущность) в составе дочерней сущности может принимать любые значения (в пределах заданных типов данных), в том числе и значение **Null**.

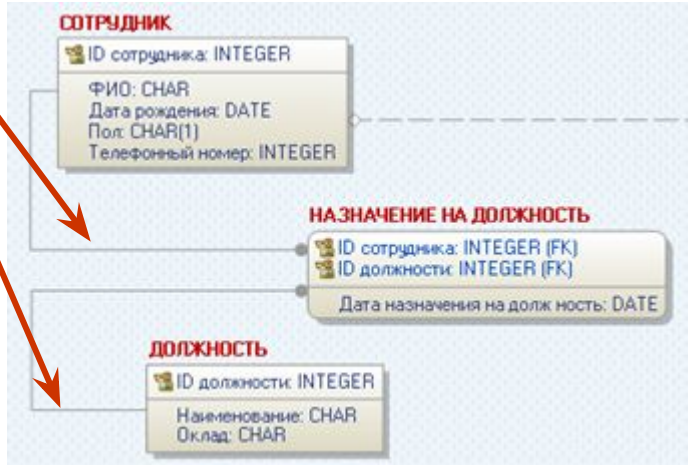
Идентифицирующие и неидентифицирующие связи

неидентифицирующие связи

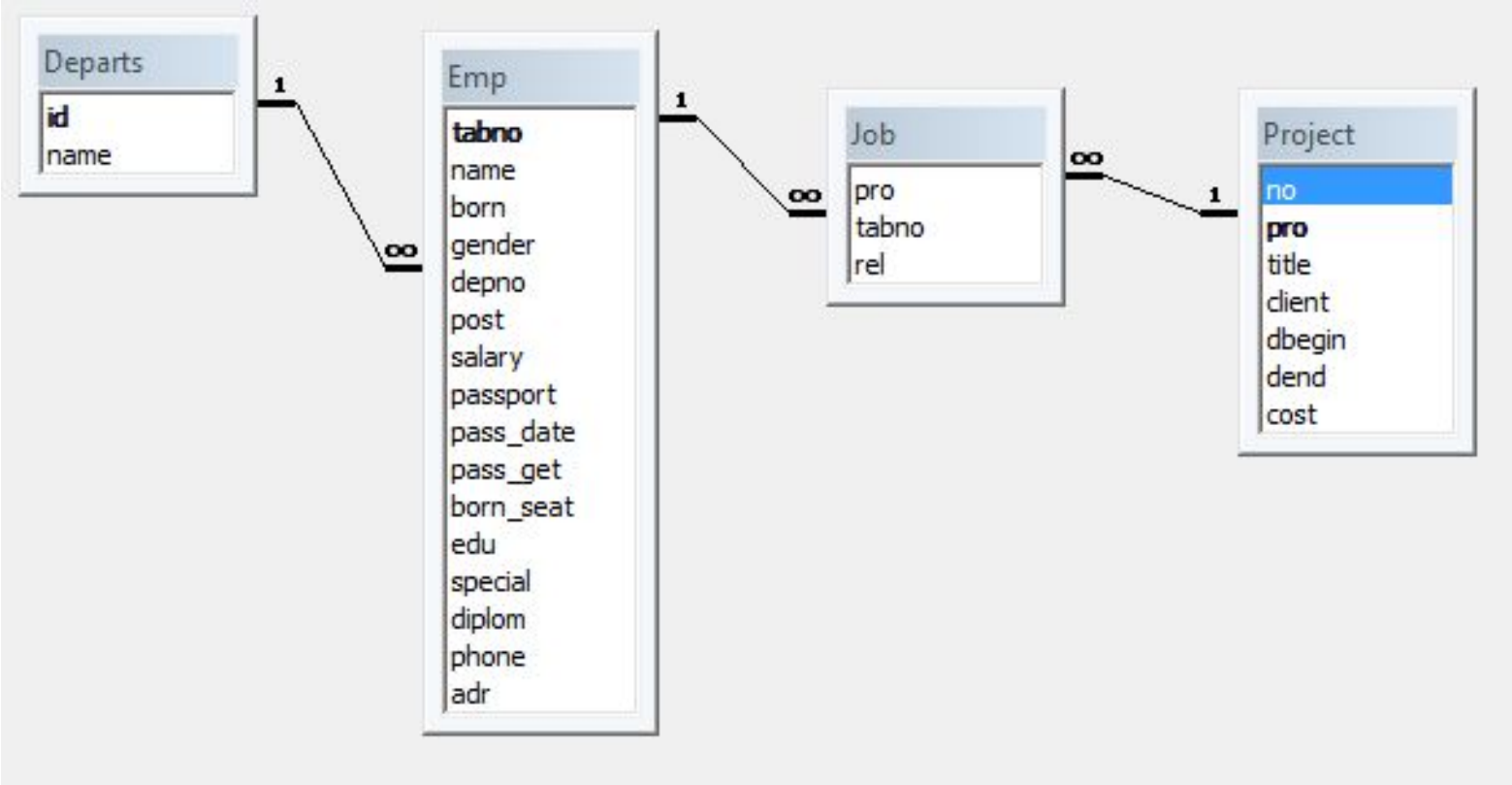


идентифицирующие связи





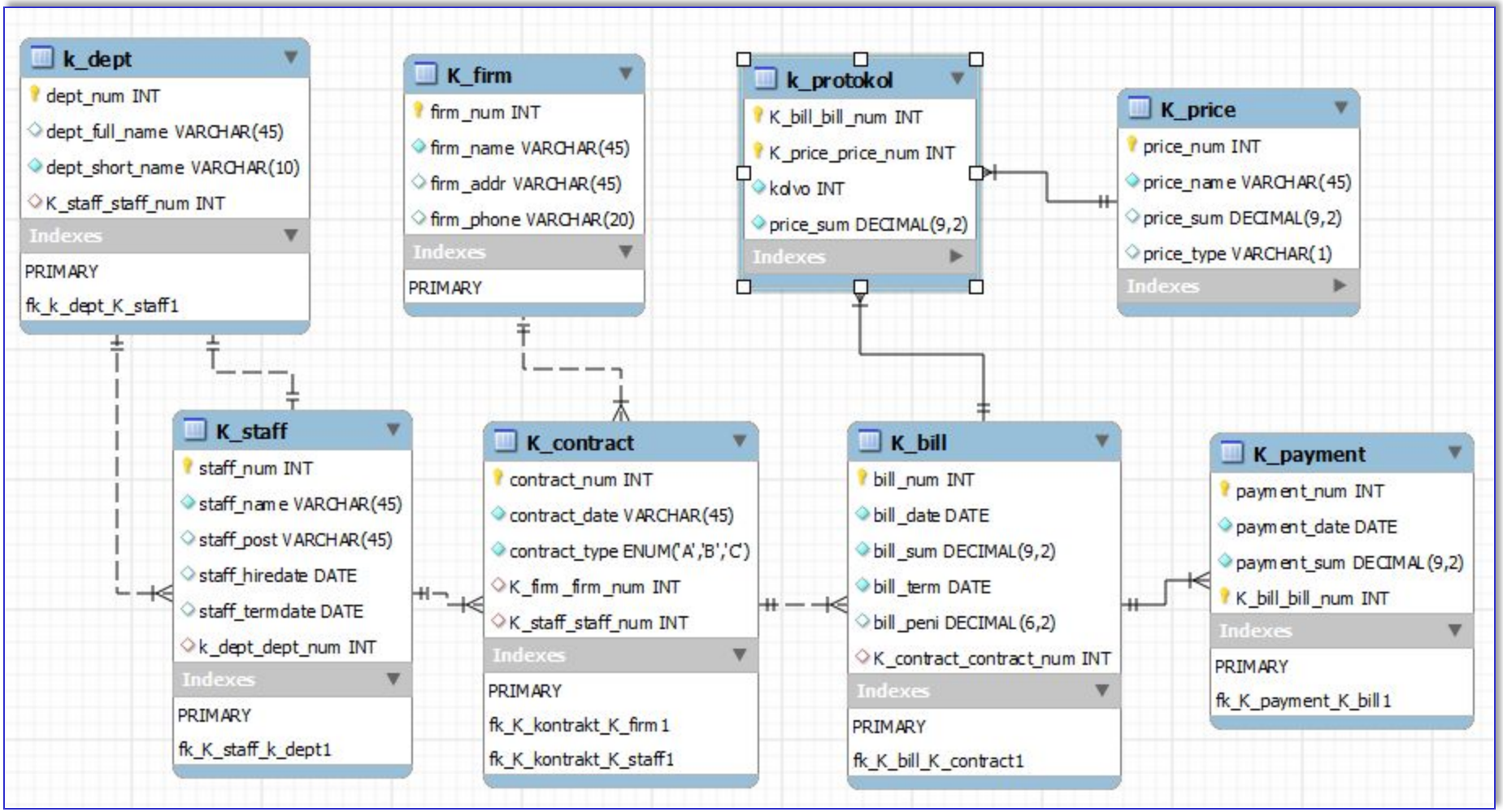
Пример БД: «Проектная организация»



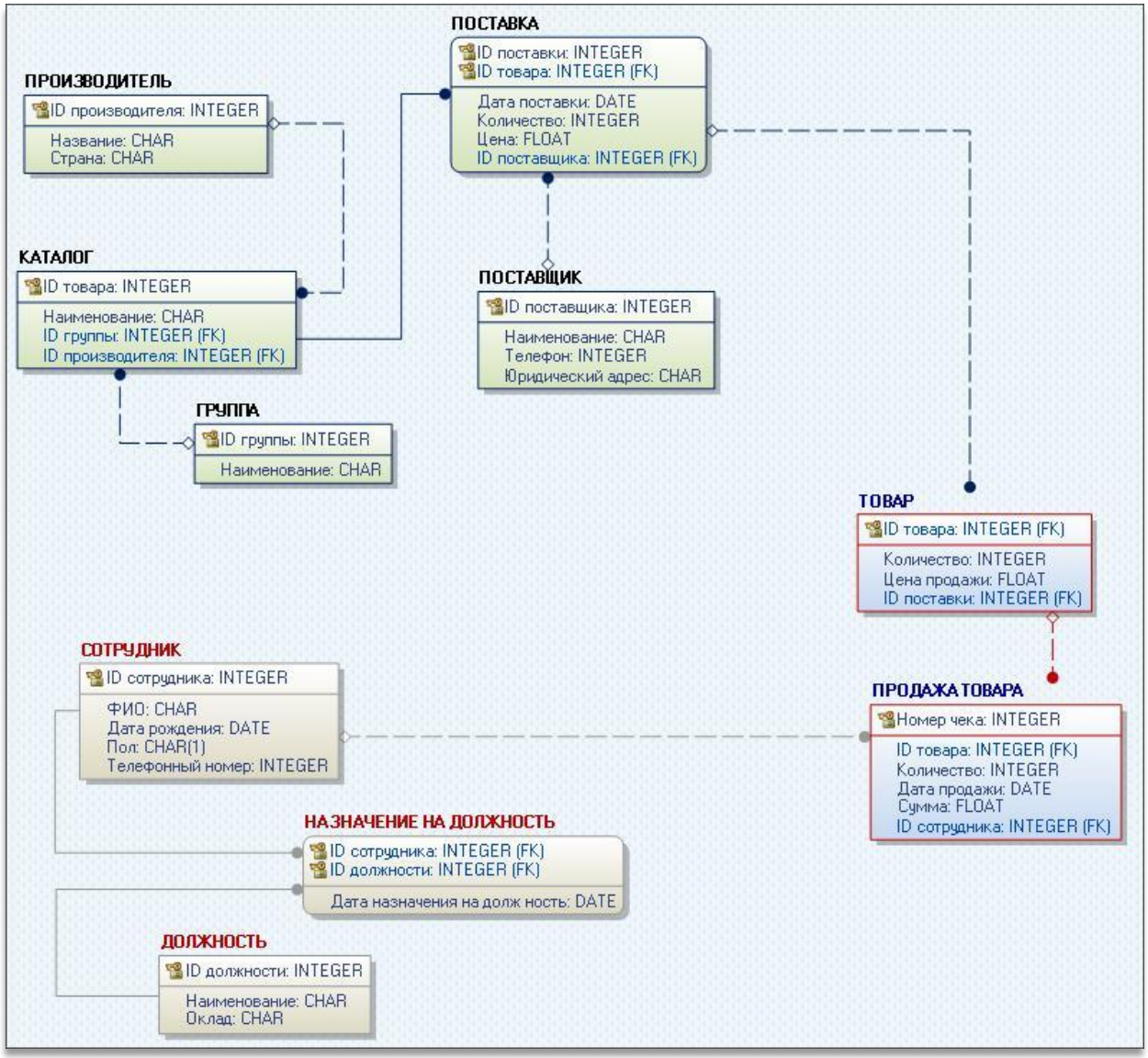
Departs – отделы, **Project** – проекты,

Emp – сотрудники, **Job** – участие в проектах.

Пример ER-диаграммы в MySQL WORKBENCH



Пример ER-диаграммы в CA ERwin Data Modeler (ERwin)



Нормальные формы ER-схем

- Как и в реляционных схемах баз данных, в ER-схемах вводится понятие **нормальных форм**, причем их смысл соответствует смыслу реляционных нормальных форм.
- Заметим, что формулировки нормальных форм ER-схем делают более понятным смысл нормализации реляционных схем. Приведем краткие и неформальные определения трех первых нормальных форм.
- **В первой нормальной форме ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.**

Нормальные формы ER-схем

- Во второй нормальной форме устраняются атрибуты, зависящие только от части уникального идентификатора. Эти атрибуты должны определять отдельную сущность.
- В третьей нормальной форме устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.
- При правильном проектировании все СУЩНОСТИ должны быть по крайней мере в третьей нормальной форме.

Получение реляционной схемы из ER-схемы

- **Шаг 1.** Каждая простая сущность превращается в таблицу. Имя сущности становится именем таблицы.
- **Шаг 2.** Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.
- **Шаг 3.** Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы.

Получение реляционной схемы из ER-схемы

- **Шаг 4.** Связи многие-к-одному (и один-к-одному) становятся внешними ключами. Т. е. делается копия уникального идентификатора с конца связи "один", и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.
- **Шаг 5.** Индексы создаются для первичного ключа (уникальный индекс) и внешних ключей.

4. Пример разработки ER-модели

Пример разработки ER-модели

При разработке ER-моделей проектировщик БД должен получить следующую информацию о предметной области:

- ✔ **Список сущностей** предметной области.
- ✔ **Список атрибутов** сущностей.
- ✔ **Описание взаимосвязей** между сущностями.

- 🖌 **ER-диаграммы** удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

Пример разработки ER-модели

Предположим, что перед нами стоит задача **разработать информационную систему управления заказами для оптовой торговой фирмы**. В первую очередь мы должны изучить предметную область и процессы, происходящие в ней. Для этого мы опрашиваем сотрудников фирмы, читаем документацию, изучаем формы заказов, накладных и т.п.

Менеджер по продажам считает, что проектируемая система должна выполнять следующие действия:

- ✔ Хранить информацию о покупателях.
- ✔ Печатать накладные на отпущенные товары.
- ✔ Следить за наличием товаров на складе

Пример разработки ER-модели

Выделим все **существительные** в этих предложениях - это будут **потенциальные кандидаты на сущности и атрибуты**, и проанализируем их (непонятные термины будем выделять знаком вопроса):

- ✔ **Покупатель** - явный кандидат на сущность
- ✔ **Накладная** - явный кандидат на сущность
- ✔ **Товар** - явный кандидат на сущность
- ✔ **(?)Склад** - а вообще, сколько складов имеет фирма? Если несколько, то это будет кандидатом на новую сущность.
- ✔ **(?)Наличие товара** – это, скорее всего, атрибут, но атрибут какой сущности?

Пример разработки ER-модели

- Сразу возникает очевидная связь между сущностями - "покупатели могут покупать много товаров" и "товары могут продаваться многим покупателям". Первый вариант диаграммы выглядит так:

Пример разработки ER-модели

После уточнения информации стало известно, что:

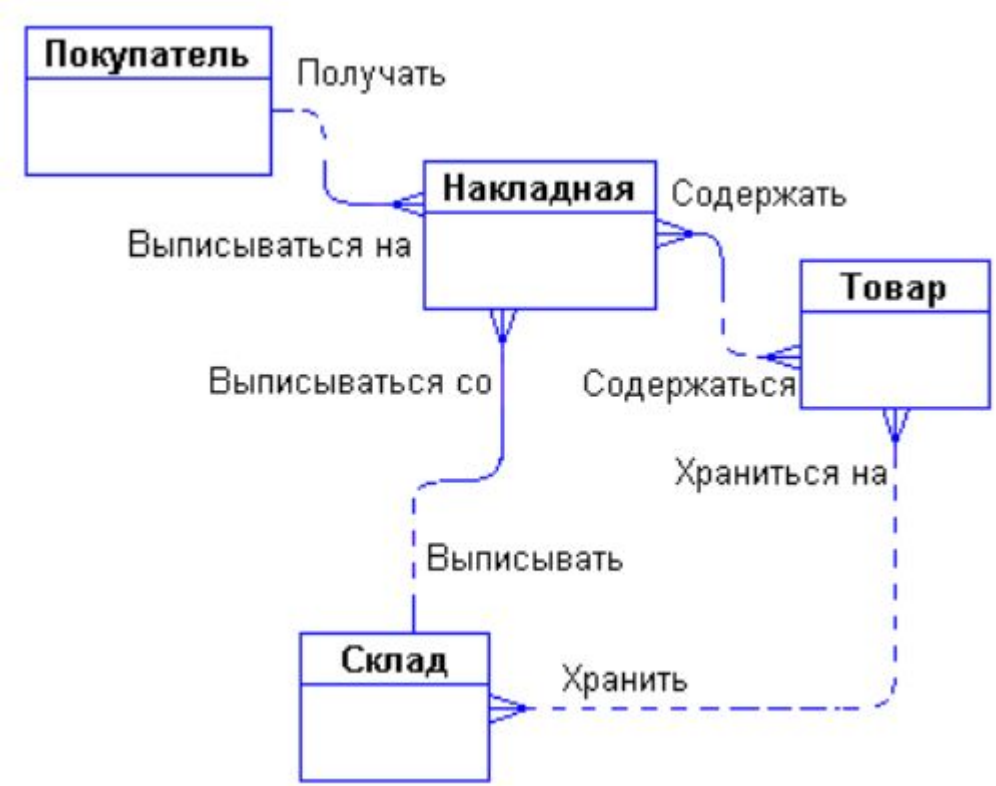
- Фирма имеет несколько складов. Причем, каждый товар может храниться на нескольких складах и быть проданным с любого склада.
- Покупатели покупают товары, получая при этом накладные, в которые внесены данные о количестве и цене купленного товара. Каждый покупатель может получить несколько накладных. Каждая накладная обязана выписываться на одного покупателя.

Пример разработки ER-модели

- Каждая накладная обязана содержать несколько товаров (не бывает пустых накладных).
- Каждый товар, в свою очередь, может быть продан нескольким покупателям через несколько накладных.
- Каждая накладная должна быть выписана с определенного склада, и с любого склада может быть выписано много накладных.

Пример разработки ER-модели

Таким образом, после уточнения, диаграмма будет выглядеть следующим образом:



Пример разработки ER-модели

Перейдем к атрибутам сущностей. Беседуя с сотрудниками фирмы, мы выяснили следующее:

- ✔ **Каждый покупатель** является юридическим лицом и имеет наименование, адрес, банковские реквизиты.
- ✔ **Каждый товар** имеет наименование, цену, а также характеризуется единицами измерения.
- ✔ **Каждая накладная** имеет уникальный номер, дату выписки, список товаров с количествами и ценами, а также общую сумму накладной. Накладная выписывается с определенного склада и на определенного покупателя.
- ✔ **Каждый склад** имеет свое наименование.

Пример разработки ER-модели

Снова выпишем все существительные, которые будут потенциальными атрибутами, и проанализируем их:

- ✔ **Юридическое лицо** - термин риторический, фирма не работает с физическими лицами. Не обращаем внимания.
- ✔ **Наименование покупателя** - явная характеристика покупателя.
- ✔ **Адрес** - явная характеристика покупателя.
- ✔ **Банковские реквизиты** - явная характеристика покупателя.
- ✔ **Наименование товара** - явная характеристика

Пример разработки ER-модели

- ✔ (?)**Цена товара** - похоже, что это характеристика товара. Отличается ли эта характеристика от цены в накладной?
- ✔ **Единица измерения** - явная характеристика товара.
- ✔ **Номер накладной** - явная уникальная характеристика накладной.
- ✔ **Дата накладной** - явная характеристика накладной.
- ✔ (?)**Список товаров в накладной** - список не может быть атрибутом. Вероятно, нужно выделить этот список в отдельную сущность.

Пример разработки ER-модели

- ✔ **(?)Количество товара в накладной** - это явная характеристика, но характеристика чего? Это характеристика не просто "товара", а "товара в накладной".
- ✔ **(?)Цена товара в накладной** - опять же это должна быть не просто характеристика товара, а характеристика товара в накладной. Но цена товара уже встречалась выше - это одно и то же?
- ✔ **Сумма накладной** - явная характеристика накладной. Эта характеристика не является независимой. Сумма накладной равна сумме стоимостей всех товаров, входящих в накладную.
- ✔ **Наименование склада** - явная характеристика склада.

Пример разработки ER-модели

В ходе дополнительной беседы с менеджером удалось прояснить различные понятия цен. Оказалось, что каждый товар имеет некоторую **текущую цену**. Эта цена, по которой товар продается в данный момент. Естественно, что эта цена может меняться со временем. **Цена одного и того же товара в разных накладных, выписанных в разное время, может быть различной.** Таким образом, **имеется две цены - цена товара в накладной и текущая цена товара.**

Пример разработки ER-модели

Сущности "Накладная" и "Товар" связаны друг с другом отношением типа **много-ко-многим**. Такая связь должна быть расщеплена **на две связи типа один-ко-многим**. Для этого требуется дополнительная сущность. Этой сущностью и будет сущность **"Список товаров в накладной"**. Связь ее с сущностями "Накладная" и "Товар" характеризуется следующими фразами - **"каждая накладная обязана иметь несколько записей из списка товаров в накладной"**, **"каждая запись из списка товаров в накладной обязана включаться ровно в одну накладную"**, **"каждый товар может включаться в несколько записей из списка товаров в накладной"**, **"каждая запись из списка товаров в накладной обязана быть связана ровно с одним товаром"**.

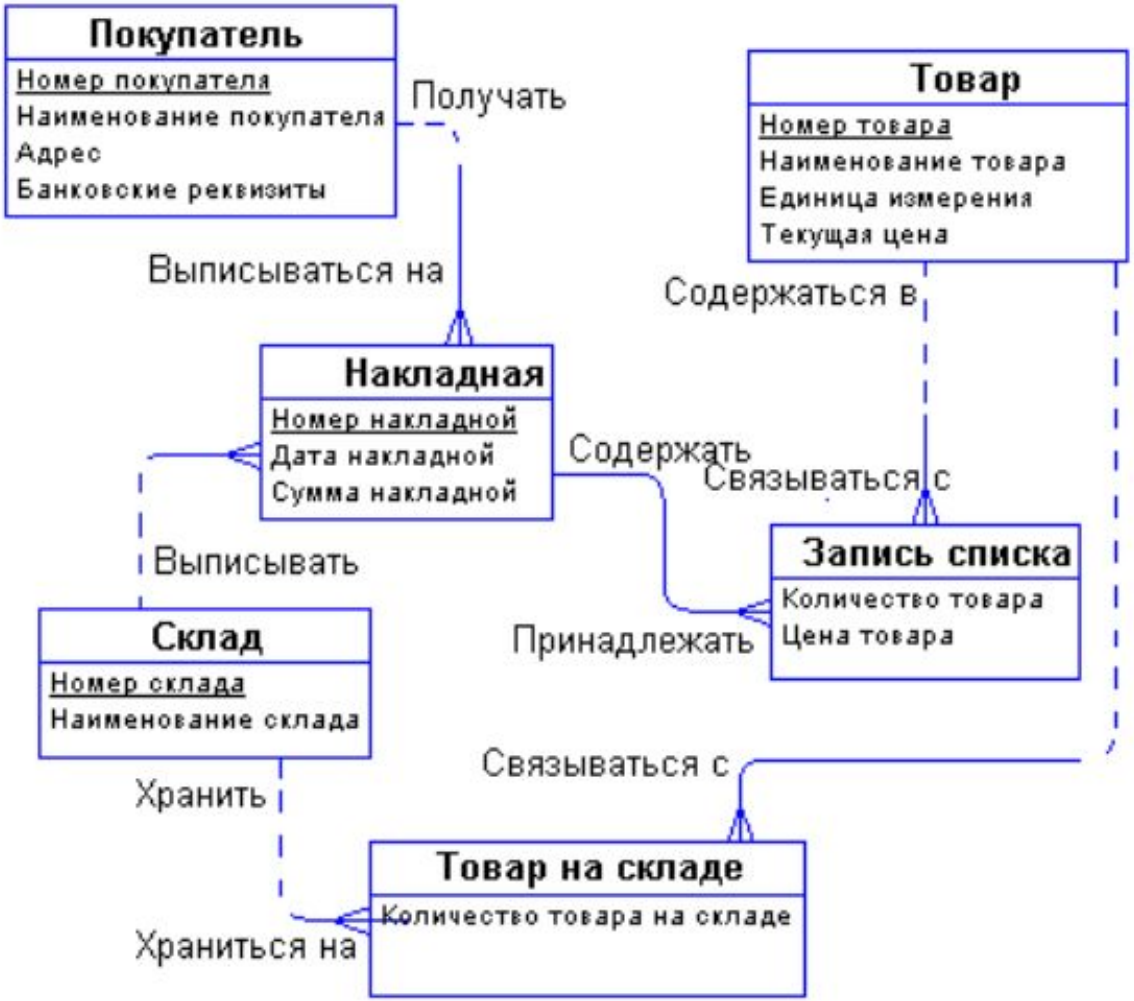
Пример разработки ER-модели

- Атрибуты "Количество товара в накладной" и "Цена товара в накладной" являются атрибутами сущности "Список товаров в накладной".

- Точно также поступим со связью, соединяющей сущности "Склад" и "Товар". Введем дополнительную сущность "Товар на складе". Атрибутом этой сущности будет "Количество товара на складе". Таким образом, товар будет числиться на любом складе и количество его на каждом складе будет свое.

Пример разработки ER-модели

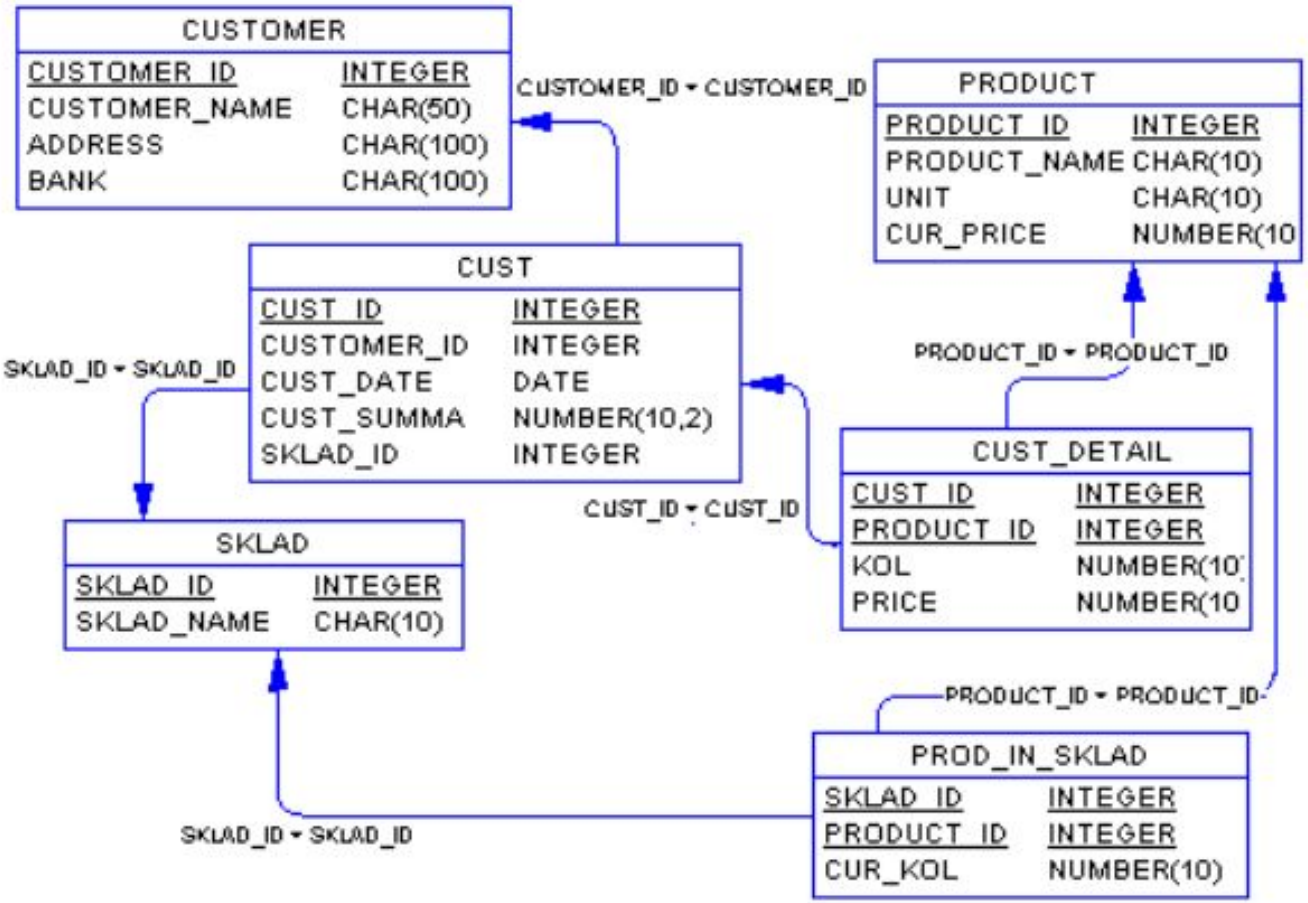
Результаты анализа предметной области отразим на концептуальной модели:



Пример разработки ER-модели

Концептуальная модель не учитывает особенности конкретной СУБД. На ее основе строятся логическая и физическая модели данных. В **физической модели** учитываться такие особенности СУБД, как допустимые типы и наименования полей и таблиц, ограничения целостности и т.п.

Пример физической диаграммы



На данной диаграмме каждая сущность представляет собой таблицу базы данных, каждый атрибут становится колонкой соответствующей таблицы.

Пример разработки ER-модели

Во многих таблицах, например, "CUST_DETAIL" и "PROD_IN_SKLAD", соответствующих сущностям "Запись списка накладной" и "Товар на складе", появились новые атрибуты, которых не было в концептуальной модели - это **ключевые атрибуты родительских таблиц**, мигрировавших в дочерние таблицы для того, чтобы обеспечить связь между таблицами посредством внешних ключей.

Полученные таблицы сразу находятся в 3НФ.

Выводы

- ✔ Реальным средством моделирования данных является не формальный метод нормализации отношений, а так называемое **семантическое моделирование.**
- ✔ В качестве инструмента семантического моделирования используются различные варианты диаграмм **сущность-связь** (ER - Entity-Relationship).
- ✔ **Диаграммы сущность-связь** позволяют использовать наглядные графические обозначения для моделирования сущностей и их взаимосвязей.

Выводы

- Различают **концептуальные и физические** ER-диаграммы. Концептуальные диаграммы не учитывают особенностей конкретных СУБД. **Физические диаграммы строятся по концептуальным и представляют собой прообраз конкретной базы данных**. Сущности, определенные в концептуальной диаграмме становятся таблицами, атрибуты становятся колонками таблиц (при этом учитываются допустимые для данной СУБД типы данных и наименования столбцов), связи реализуются путем миграции ключевых атрибутов родительских сущностей и создания внешних ключей.
- При правильном определении сущностей, полученные таблицы будут сразу находиться в 3НФ. Основное достоинство метода состоит в том, модель строится методом последовательных уточнений первоначальных диаграмм.

Спасибо за внимание!