**Лекция 6:**

**Метод моделирования "сущность-связь"**

**Цель лекции**

Изучив материал настоящей лекции, вы будете знать:

* определение *предметной области* базы данных
* что представляет собой *логическая модель предметной области* базы данных;
* основные конструкции и элементы логической модели базы данных;
* что собой представляют шесть *нормальных форм отношений* ;
* способы приведения *отношений* к нормальным формам;

и научитесь:

* различать основные понятия *предметной области*: *объект (сущность)*, *ядро предметной области*, ситуация, *состояние предметной области* ;
* читать *диаграммы "сущность-связь"*.
* строить нормальные формы в модели "сущность-связь".

**Введение**

Для логического проектирования реляционных ХД применяются следующие методики.

* **Метод моделирования "сущность-связь"** (ER modeling) дает абстрактную модель *предметной области*, используя следующие основные понятия: *сущности* (entities), *взаимосвязи* (relationships) между сущностями и *атрибуты* (attributes) для представления свойств сущностей и взаимосвязей.
* **Метод многомерного моделирования** (*Dimensional* modeling) дает абстрактную модель *предметной области*, используя следующие основные понятия: *показатели* или *метрики* (measures), *факты* (facts) и *измерения* (dimensions).
* **Методы моделирования временных данных** (*Temporal* *data modeling*) дают абстрактную модель фрагмента *предметной области*, представляющего временные ряды данных, и используют следующие основные понятия: *временные метки* (timestamps), *временной ряд* (*time series*), дата, диапазон дат, классы.
* **Метод моделирования "свод данных"** (Data Vault) дает абстрактную модель фрагмента *предметной области*, основываясь на математических принципах нормализации *отношений*, и использует следующие основные понятия: *сущности-концентраторы* (*Hub Entities*), *связывающие сущности* (*Link Entities*), *сущности-сателлиты* (*Satellite Entities*),

В настоящей лекции мы рассмотрим метод моделирования "*сущность-связь*".

**Понятие предметной области и архитектура данных**

**Понятие предметной области**

Основным назначением информационных систем (ИС), в том числе и систем складирования данных, является оперативное обеспечение пользователя информацией о внешнем мире путем реализации *вопросно-ответного отношения*. Вопросно-ответные *отношения*, получая интерпретацию во внешнем мире (мире вне ИС), позволяют выделить для ИС определенный его фрагмент — *предметную область*, который и будет воплощен в системе. Информация о внешнем мире представляется в ИС в форме данных. Это ограничивает возможности смысловой интерпретации информации и конкретизирует семантику ее представления в ИС. Совокупность этих выделенных данных для ИС образует *логическую модель предметной области*, описывающие ее состояние с определенной точностью.

Важно понимать, что *логическая модель предметной области* создается на этапе анализа требований к ИС и не содержит предположений о технологии реализации хранилища или базы данных.

Оперируя терминами "данные" и "вопросы", вопросно-ответное *отношение* можно представить в виде таблицы, столбцами которой являются элементы данных, а строками — вопросы. Каждая ячейка такой таблицы имеет логическое значение 1, если вопрос использует этот элемент данных, или 0 — в противном случае.

Понятие *предметной области* является одним из базовых понятий информатики и не имеет точного определения. Его использование в контексте ИС предполагает существование устойчивой во времени соотнесенности между именами, понятиями и определенными реалиями внешнего мира, не зависящей от самой ИС и ее круга пользователей. Таким образом, введение понятия *предметной области* ограничивает и делает обозримым пространство информационного поиска в ИС, и позволяет выполнять запросы за конечное время.

Совокупность реалий (объектов) внешнего мира — объектов, о которых можно задавать вопросы, — образует объектное *ядро предметной области*. Невозможно получить в ИС ответ на вопрос о том, что ей неизвестно.

Термин "объект" является первичным, неопределяемым понятием. Синонимами термина "объект" являются "реалия", "сущность", "вещь". Отметим, что термин "сущность" понимается далее несколько уже, как компонент определенной *логической модели предметной области*. Выделяемые в *предметной области* объекты превращаются аналитиками (а не проектировщиками) в сущности. При этом ***сущность предметной области*** **понимается как результат абстрагирования реального объекта путем выделения и фиксации его свойств**.

На [рис. 6.1](https://intuit.ru/studies/courses/599/455/lecture/10163?page=1#image.6.1) представлен один из подходов к классификации объектов *предметной области*.



**Рис. 6.1.**Классификация объектов предметной области

Примерами сущностей (с точки зрения *логической модели предметной области* ) или объектов (с точки зрения внешнего мира по отношению к ИС) являются: студент, группа студентов, аудитория для занятий, время занятий и. т. д.

С объектами связано две проблемы — идентификация и адекватное описание. Для идентификации используют **имя**. При этом предполагается, что происходит отказ от его смысла, который присущ естественному языку. Используется только указательная функция имени. **Имя — это прямой способ идентификации объекта**. К косвенным способам идентификации объекта относят его свойства в их понимании как характеристики или признака.

Объекты взаимодействуют между собой через свои свойства, что порождает ситуации. **Ситуации – это взаимосвязи, выражающие взаимоотношения между объектами**. *Ситуации в предметной области* описываются посредством высказываний о *предметной области* с использованием исчисления высказываний и исчисления предикатов, т.е. формальной, математической логики. Например, высказывание "Программист, менеджер есть служащие компании" описывает *отношение* включения. Таким образом, вся информация об объектах и *сущностях предметной области* описывается с помощью утверждений на естественном языке.

Методы математической логики позволяют формализовать эти утверждения и представить их в виде, пригодном для анализа.

**Пример**.

Рассмотрим высказывание: "Студент Иванов А.А, родился в 1992 году". Оно выражает следующие свойства объекта "Иванов А.А.":

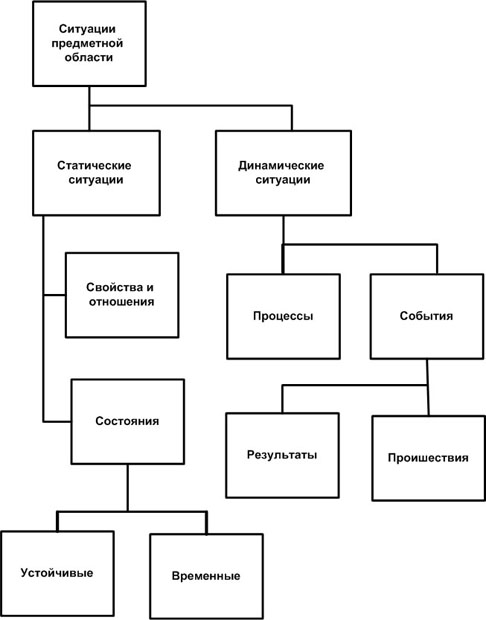
* в явном виде — год рождения;
* в неявном виде – принадлежность к студентам.

Первое свойство устанавливает связь между парами объектов "Иванов А.А." и "год рождения", а второе свойство устанавливает связь между парами объектов "Иванов А.А." и "множество студентов". Формализация этого высказывания представляется как результат присваивания значений переменных, входящих в следующие предикаты:

|  |
| --- |
| РОДИЛСЯ (Иванов А.А., 1982) |
| ЯВЛЯЕТСЯ СТУДЕНТОМ (Иванов А.А.) |

Отметим, что в семантике естественных языков ситуация и взаимосвязь считаются почти синонимами. Ситуация содержит высказывание об объектах *предметной области*, которому можно приписать некоторую оценку истинности и представить в виде предиката после введения переменных. Таким образом, совокупность высказываний о *предметной области* можно трактовать как определение информационного пространства для ИС.

На [рис. 6.2](https://intuit.ru/studies/courses/599/455/lecture/10163?page=1#image.6.2) представлен один из подходов к классификации ситуаций в рамках *предметной области*.

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_02.jpg)

**Рис. 6.2.**Классификация ситуаций предметной области

Различают статические и динамические ситуации. Примерами статических ситуаций являются такие ситуации, как "иметь цвет", "иметь возраст" и т.д. Примерами динамических ситуаций являются такие ситуации, как "создать механизм", "выпечь хлеб" и т.д.

Обратите внимание на то, что ситуация также может представлять собой объект и обладать свойствами. Подобная коллизия порождает неоднозначность при моделировании *предметной области*.

Приведенная выше классификация вводит в *предметную область* два важных аспекта – пространство и время, причем время понимается и как момент события, и как интервал между событиями. *Предметная область* существует в пространстве и во времени, т.е. ей присущи, как и реальному миру, временные и пространственные *отношения* и связи. Следует отличать реальное время внешнего мира и его отражение в ИС и в источниках информации. В ИС взаимосвязи, зависящие от времени, фиксируются только после их регистрации. Таким образом, *предметная область* в каждый конкретный момент времени представляет собой **выделенную совокупность определенных объектов и ситуаций, называемую** ***состоянием предметной области*** **(или снимком)**.

Таким образом, ***предметная область*** **— это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в ИС в качестве алгоритмической модели фрагмента действительности**.

Понятие *предметной области* было введено в начале 80-х годов прошлого века, когда учеными в области ИС была осознана необходимость использовать семантические модели для представления информации в компьютерных системах. Так же, как требования к компьютерной системе формируются средствами естественного языка, так и информация в компьютерных системах представляется средствами особого языка с определенной семантикой. Такой подход впервые был представлен П. Ченом в 1976 году.

Примером классической *предметной области* для создания систем складирования данных и, следовательно, для ХД является задача анализа продаж компании. В качестве объектов этой *предметной области* можно выделить следующие: "менеджер продаж", "товар", "склад", "офис продаж", "покупатель". В качестве ситуаций – "продать товар", "купить товар", "отгрузить товар со склада", "доставить товар".

**Архитектура данных предметной области**

ХД является предметно-ориентированной, интегрированной, постоянной во времени коллекцией исторических данных, используемой различными группами пользователей для поддержки принятия решений или анализа данных. Информация в компьютерных системах, в том числе и в ХД, представляется в виде элементов данных (items). Одно из основных положений концепции ХД состоит в очистке, фильтрации, преобразовании, суммировании и агрегации данных, а затем размещении их в некоторой структуре для обеспечения информационных потребностей пользователей. Определение такой структуры является одной из основных задач *логического моделирования* ХД.

Одной из первых задач проектировщика ХД является определение архитектуры данных. **Архитектура данных — это принципы субъективного представления информации в виде данных в рамках модели предметной области**. При построении архитектуры данных проектировщик ХД определяет элементы данных, их свойства и взаимосвязи между ними. Одним из ключевых моментов построения архитектуры данных является *степень детализации информации* при преобразовании ее в элементы данных. Процесс такого преобразования называют **структуризацией данных**.

Для данных OLTP-систем решение вопросов, связанных с уровнем детализации данных, не является столь важным, как в системах складирования данных. В БД OLTP-систем данные обычно детально структурированы. Для представления данных в ХД проектировщик должен специально решить вопрос об уровне структуризации данных, исходя из требований к системе складирования данных. Решение этого вопроса весьма важно, поскольку при агрегации и суммировании данных некоторые диапазоны данных из подающей OLTP-системы могут быть не представлены в ХД. Например, ХД телекоммуникационной компании может содержать оплату за пользование телефоном, просуммированную по минутам, а подающая система хранит такие данные по секундам.

**Уровень структуризации (детализации или гранулированности) данных** (Data *granularity*) является одной из самых важных характеристик ХД. **Уровень структуризации данных — это степень детализации хранимых данных, оптимальная с точки зрения решения информационно-аналитических задач в рамках предметной области ХД**.

Грубо говоря, уровень структуризации данных определяет количество запросов, на которые можно получить ответы в системе складирования данных. Если в ХД поддерживается высокий уровень структуризации данных, то система поддерживает практически любой запрос в рамках *предметной области* ХД. В примере, приведенном выше, невозможно получить ответ на вопрос: сколько абонент Иванов А.А. заплатил за пять секунд разговора первого января 2009 года.

Поддержка высокого уровня структуризации данных приводит к необходимости хранить и сопровождать большие объемы данных, что может отрицательно сказываться на производительности *системы складирования данных*, в частности, на времени ответа на запрос. С другой стороны, низкий уровень структуризации данных приводит к тому, что система складирования данных может отвечать на строго ограниченный круг запросов. Поэтому одной из задач проектировщика ХД на уровне *логического моделирования* является принятие решения об оптимальном уровне структуризации данных в ХД.

Структуризация данных предполагает разбиение всего набора данных на определенные классы с целью дальнейшей детализации внутри выделенного класса. Для ХД характерны три основных вида данных (класса).

* **Фактические данные** (Real-time data) представляют собой текущее состояние количественных и качественных показателей деятельности организации. Источником таких данных являются обычно OLTP-системы. Таким данным присущ высокий уровень структуризации. Для того чтобы использовать такие данные в ХД, их нужно предварительно обработать с помощью процедур очистки.
* **Производные данные** (Derived data) представляют собой данные, которые получены в результате суммирования, агрегации и усреднения фактических данных. В зависимости от задач анализа такие данные могут быть либо детальными, либо итоговыми.
* **Консолидированные данные** (Reconciled data) — это фактические данные, которые были очищены и представляют собой интегрированный источник данных для решения задач анализа. Основное требование к таким данным — их согласованность (consistency).

**Понятие предметной области и хранилища данных**

Понятие *предметной области* используется практически при проектировании и разработке всех классов информационных систем с базами и хранилищами данных. *Предметная область* определяет ту часть реального мира, которая будет моделироваться и реализоваться в системе. *Предметная область* определяет наиболее общие вытекающие из ее семантики критерии и требования к системе.

ХД по определению есть предметно-ориентированная электронная коллекция, т.е. оно изначально ориентировано на определенные направления деятельности организации, *предметную область*, — например, такие как производство или продажа.

Вопросы, с которыми пользователи обращаются к ХД, носят, как правило, стратегический и более обобщенный характер, чем в OLTP-системах. Ответы на них предполагают агрегацию и суммирование данных по различным направлениям деятельности организации. Это требует от систем с ХД ориентации на конкретные *предметные области* деятельности организации.

*Предметные области* в системах с ХД формируются в соответствии с направлениями деятельности организации. Чтобы определить список *предметных областей* для таких систем, необходимо определить основные виды деятельности организации — например, продажи, производство, клиенты и т.д.

Для выделения *предметных областей* в ХД часто используется так называемая методика "правило SW1", а именно ответы на вопросы: когда (when), где (where), кто (who), что (what), почему (why) и как (how) – по отношению к видам деятельности организации (интересы бизнеса). Например, при ответе на вопрос "кто" интересы бизнеса могут охватывать следующие объекты: "покупатели", "сотрудники", "поставщики", "менеджеры", "партнеры по бизнесу" и т. д.

После построения списка потенциальных *предметных областей* для систем с ХД обычно выполняют их дальнейшую детализацию путем декомпозиции каждой *предметной области*. В результате может быть получен список *предметных областей*, наиболее полно представляющий деятельность организации. При этом важно определить взаимоотношения между выделенными *предметными областями*, что является важным для определения измерений при *многомерном моделировании* ХД.

Отметим, что при решении задач анализа и, следовательно, при разработке BI-систем наиболее перспективным подходом для определения *предметной области* является изучение бизнес-процессов организации, а не функции, как в случае OLTP-систем.

Далее рассмотрим метод моделирования "сущность-связь". Этот метод используется для представления *предметной области* в виде ее логической модели. Применение метода создает модель *предметной области*, не зависимую от реализации. Метод применяется как при моделировании *предметных областей* OLTP-систем, так и при моделировании *предметных областей* BI-систем. Знание этого метода помогает проектировщику ХД быстрее установить логические связи между моделями БД OLTP-систем масштаба организации и моделями ХД BI-систем.

**Моделирование методом "сущность-связь"**

**Основные понятия модели "сущность-связь"**

Результатом моделирования методом "сущность-связь", или ER-моделирования, является ER-модель. ER-модель представляется с помощью *ER-диаграмм*, которые являются графической нотацией для абстрагирования данных в виде сущностей, взаимосвязей и атрибутов. Таким образом, семантика *предметной области* представляется в ER-модели в терминах субъективных средств описания – **сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов** и т.д.

***Сущность предметной области*** **является результатом абстрагирования реального объекта путем выделения и фиксации набора его свойств**. Таким образом, сущность представляет класс объектов, который является результатом абстрагирования реального объекта. Обычно они обозначаются именем существительным естественного языка.

Сущность описывается с помощью данных, именуемых свойствами или **атрибутами** (attributes) сущности. Как правило, атрибуты являются определениями в высказывании о сущности и обозначаются именами существительными естественного языка.

Сущности вступают в связи друг с другом через свои атрибуты. Каждая группа атрибутов, описывающих одно реальное проявление сущности, представляет собой *экземпляр сущности* (instance). Иными словами, ***экземпляр сущности*** **– это реализации сущности, отличающиеся друг от друга и допускающие однозначную идентификацию**. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически, имя сущности дается по имени ее экземпляра.

Одним из основных компьютерных способов распознавания сущностей в ИС является присвоение сущностям **идентификаторов** (Entity identifier). Часто *идентификатор сущности* называют **ключом**. Задача выбора *идентификатора сущности* является семантически субъективной задачей. Поскольку сущность определяется набором своих атрибутов, для каждой сущности целесообразно выделить такое подмножество атрибутов, которое однозначно идентифицирует данную сущность.

Некоторые сущности имеют естественные идентификаторы. Например, естественным идентификатором счета-фактуры является его номер. *Идентификаторы сущности* могут быть составными — состоящими из нескольких атрибутов, и атомарными — состоящими из одного *атрибута сущности*.

***Уникальный идентификатор сущности*** **— это атрибут сущности, позволяющий отличать одну сущность от другой**. Если сущность имеет несколько *уникальных идентификаторов*, так называемых возможных ключей, то проектировщик должен выбрать **первичный ключ сущности**.

Различают **однозначные и многозначные атрибуты. Однозначными являются атрибуты, которые в пределах конкретного** *экземпляра сущности* **имеют только одно значение. В противном случае они считаются многозначными**.

Важным моментом изучения модели *предметной области* проектировщиком является выделение многозначных *атрибутов сущности*. Это связано с тем, что реляционная модель не поддерживает многозначных атрибутов и они должны быть разрешены на последующих стадиях проектирования.

Каждый атрибут имеет *домен* (domain). ***Домен*** **— это выражение, которое определяет значения, разрешенные для данного атрибута**. Иными словами, *домен* — это область значений атрибута. Для каждого *атрибута сущности* должен быть определен *домен*. На уровне *логического моделирования* данных назначение *домена* атрибуту носит общий характер. Например, атрибут текстовый, числовой, бинарный, дата или "не определен". В последнем случае аналитик должен дать описание *домена*. На последующих стадиях тип *домена* конкретизируется, смысл понятия *домена* в физической модели ХД уже, чем его может понимать аналитик. Это связано с тем, что в рамках физической модели *домен* реализуется посредством механизма ограничения *домена*, а СУБД не понимает неопределенных *доменов*.

Сущности не существуют отдельно друг от друга. Между ними имеются реальные *отношения* (Relationship), которые должны быть отражены в модели *предметной области*. При выделении *отношений* акцент делается на фиксацию связей и их характеристик. ***Отношение (связь)*** **представляет собой соединение (взаимоотношение) между двумя или более сущностями**. Каждая связь реализуется через значения *атрибутов сущностей*. Обычно связь обозначается глаголом. Каждая связь также должна иметь свой **уникальный идентификатор связи**.

В реляционной модели *отношения* реализуются только через ограничение целостности по внешнему ключу. Поэтому проектировщик реляционного ХД должен проконтролировать, чтобы связь между сущностями осуществлялась через точно указанные атрибуты, которые будут определять уникальный ключ связи. Выбор ключей сущностей — одно из важнейших проектных решений, которое предстоит сделать проектировщику при переходе к физической модели базы данных.

Связи характеризуются *степенью связи* и *классом принадлежности сущности* к связи. ***Степень (мощность) связи*** **– это отношение числа сущностей, участвующих в образовании связи**. Например, "один к одному", "один ко многим", "многие ко многим". На уровне логической модели допускается неопределенная или неразрешенная связь. ***Класс принадлежности сущности*** **— это характер участия сущности в связи**. Различают **обязательные** и **необязательные** *классы принадлежности сущности* к связи. Обязательным является такой *класс принадлежности*, когда *экземпляры сущности* участвуют в установлении связи в обязательном порядке. В противном случае сущность принадлежит к необязательному *классу принадлежности*. Для необязательного *класса принадлежности сущности* *степень связи* может быть равна нулю, т.е. *экземпляр сущности* можно связать с 0, 1 или несколькими экземплярами другой сущности. Для обязательного *класса принадлежности* *степень связи* не может равняться нулю.

*Отношения*, *связывающие сущность* саму с собой, называются **рефлексивными**. Типичным примером рефлексивных *отношений* является определение структуры подчиненности в отношении "Сотрудники". Рефлексивные *отношения* чаще всего отражают иерархические *отношения* внутри структуры данных. Они порождают ряд проблем проектирования, о которых речь пойдет позже

С точки зрения *отношений* различают **слабые сущности** (*weak*). **Слабые сущности – это сущности, которые не могут присутствовать в базе данных, пока не существует связанного с ней экземпляра другой сущности**. Примером такой сущности является заказ, который не может существовать без клиента. Слабые сущности имеют обязательный *класс принадлежности*, и *степень связи* такой сущности не может равняться нулю. Связь "заказ-клиент" является обязательной.

Выявление слабых сущностей и связанных с ними обязательных *отношений* необходимо для обеспечения целостности и согласованности данных. Так, например, неизвестному клиенту невозможно приписать заказ.

Иногда выделенная сущность несет в себе *отношение* включения или *отношение* "часть-целое". При этом существует некоторый атрибут, значения которого порождают разбиение множества *экземпляров сущности* на непересекающиеся подмножества — **категории сущности**. Категории сущности называют ***подтипами*** и выделяют в подчиненную в рамках *отношения* сущность, которая является категорией исходной сущности. Из исходной сущности выделяются общие для полученных категорий атрибуты, и таким образом выделяется сущность, которая становится *супертипом*. За выделенной сущностью-*супертипом* обычно оставляют наименование исходной сущности, хотя ее семантический смысл меняется.

*Супертип* с порожденными им *подтипами* является примером так называемой **составной сущности**. Составная сущность является логической конструкцией модели для представления набора сущностей и связей между ними как единого целого.

Пример. Сущность "автомобиль" можно разбить на следующие *подтипы*: автомобили с приводом на два колеса, автомобили с приводом на четыре колеса, автомобили с переключаемым приводом.

Для проектировщика важно знать, что все экземпляры сущности-супертипа относятся только к одному из ее *подтипов*. Наличие в модели *подтипов* и *супертипов* усложняют проектирование и создают определенные трудности в реализации. Поэтому важно на ранней стадии проектирования установить, является ли наличие *супертипов* в модели необходимым.

Для этого необходимо предпринять следующие действия:

* установить, много ли одинаковых свойств имеют различные *подтипы*. Следует помнить, что чем меньше *подтипы* похожи друг на друга, тем больше вероятность введения *супертипа* ;
* или найти *экземпляр сущности*, который можно обоснованно включить в более чем один *подтип*. Поскольку это противоречит определению *супертипа*, предлагаемое разбиение недопустимо.

**Графическая нотация модели: диаграммы "сущность-связь"**

Типичной формой документирования *логической модели предметной области* при ER-моделировании являются *диаграммы "сущность-связь"*, или *ER-диаграммы* (Entity Relationship Diagram). *ER-диаграмма* позволяет графически представить все элементы логической модели согласно простым, интуитивно понятным, но строго определенным правилам — **нотациям**.

Для создания ER диаграмм обычно используют одну из двух наиболее распространенных нотаций.

* Integration DEFinition for Information Modeling (*IDEF1X*). Эта нотация была разработана для армии США и стала федеральным стандартом США. Кроме того, она является стандартом в ряде международных организаций (НАТО, Международный валютный фонд и др.).
* Information Engineering (IE). Нотация, разработанная Мартином (Martin), Финкельштейном (Finkelstein) и другими авторами, используется преимущественно в промышленности.

Построение *ER-диаграмм*, как правило, ведется с использованием CASE-средств. В данной лекции во всех примерах, если это не оговорено особо, будет использоваться нотация MS Office Visio 2007.

Сущность на ER-диаграмме представляется прямоугольником с именем в верхней части ([рис. 6.3](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.3)).



**Рис. 6.3.**Представление сущности "Сотрудник" на ER-диаграмме

В прямоугольнике перечисляются *атрибуты сущности*, при этом атрибуты, составляющие *уникальный идентификатор сущности*, подчеркиваются ([рис. 6.4](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.4)).



**Рис. 6.4.**Представление сущности "Сотрудник" с атрибутами и уникальным идентификатором сущности

Каждый *экземпляр сущности* должен быть уникальным и отличаться от других атрибутов. Одним из основных компьютерных способов распознавания сущностей в ИС является присвоение сущностям идентификаторов (entity identifier). Поскольку сущность определяется набором своих атрибутов, для каждой сущности целесообразно выделить такое подмножество атрибутов, которое однозначно идентифицирует данную сущность. Часто *идентификатор сущности* называют первичным ключом (primary key).

**Первичный ключ (primary key) – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности**. Атрибуты первичного ключа на диаграмме не требуют специального обозначения – это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии ([рис. 6.3](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.3)).

Выбор первичного ключа может оказаться непростой задачей, решение которой в состоянии повлиять на эффективность будущей ИС. В одной сущности могут оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются **потенциальными ключами** (candidate key).

Ключи могут быть **сложными**, т.е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения – это список атрибутов выше горизонтальной линии.

Рассмотрим кандидатов на первичный ключ сущности "сотрудник" ([рис. 6.5](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.5)).



**Рис. 6.5.**Определение первичного ключа для сущности "сотрудник"

Здесь можно выделить следующие потенциальные ключи.

1. Табельный номер.
2. Номер паспорта.
3. Фамилия + Имя + Отчество.

Для того чтобы стать первичным, *потенциальный ключ* должен удовлетворять ряду требований.

*Уникальность*. Два экземпляра не должны иметь одинаковых значений возможного ключа. *Потенциальный ключ* ( *Фамилия + Имя + Отчество* ) является плохим кандидатом, поскольку в организации могут работать полные тезки.

*Компактность*. Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Для обеспечения уникальности ключа ( *Фамилия + Имя + Отчество* ) дополним его атрибутами *Дата рождения* и *Цвет глаз*. Если бизнес-правила говорят, что сочетания атрибутов *Фамилия + Имя + Отчество + Дата рождения* достаточно для однозначной идентификации сотрудника, то *Цвет глаз* оказывается лишним, т. е. ключ *Фамилия + Имя + Отчество + Дата рождения + Цвет глаз* не является компактным.

При выборе первичного ключа предпочтение должно отдаваться более простым ключам, т. е. ключам, содержащим меньшее количество атрибутов. В примере ключи № 1 и № 2 предпочтительней ключа № 3.

Атрибуты ключа не должны содержать нулевых значений. Если допускается, что сотрудник может не иметь паспорта или вместо паспорта иметь какое-либо другое удостоверение личности, то ключ № 2 не подойдет на роль первичного ключа. Если для обеспечения уникальности необходимо дополнить *потенциальный ключ* дополнительными атрибутами, то они не должны содержать нулевых значений. При дополнении ключа № 3 атрибутом *Дата рождения* нужно убедиться в том, что даты рождения известны для всех сотрудников.

Значение атрибутов ключа не должно меняться в течение всего времени существования *экземпляра сущности*. Сотрудница организации может выйти замуж и сменить как фамилию, так и паспорт. Поэтому ключи № 2 и 3 не подходят на роль первичного ключа.

Каждая сущность должна иметь, по крайней мере, один *потенциальный ключ*. Многие сущности имеют только один *потенциальный ключ*. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные – *альтернативными ключами*. **Альтернативный ключ (Alternate Key) – это потенциальный ключ, не ставший первичным**.

Некоторые сущности имеют естественные (натуральные) ключи. Например, естественным идентификатором счета-фактуры является его номер. В противном случае проектировщик может создать **суррогатный ключ (Surrogate Key) – атрибут, значение которого создается искусственно и не имеет отношения к предметной области**. При моделировании структур данных для ХД суррогатные ключи во многих ситуациях являются более предпочтительными.

*Домены* назначаются аналитиками и фиксируются в специальном документе — *словаре данных* (*Data Dictionary*). При создании логической модели *домены* могут быть специфицированы в сущностях на ER-диаграмме.

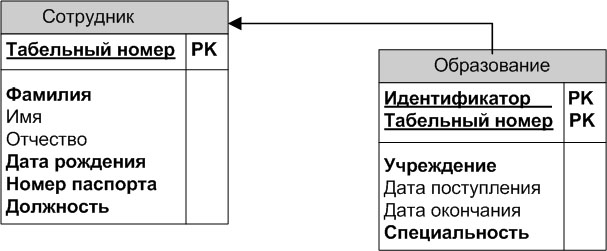
Каждый атрибут имеет *домен*. *Домен* можно определить как абстрактный атрибут, на основе которого можно создавать обычные атрибуты, при этом создаваемые атрибуты будут иметь все свойства домена-прародителя. Каждый атрибут может быть определен только на одном *домене*, но на каждом *домене* может быть определено множество атрибутов. В понятие *домена* входит не только тип данных, но и область значений данных. Например, можно определить *домен* "Возраст" как положительное целое число и определить атрибут *Возраст сотрудника* как принадлежащий этому *домену*.

На уровне *логического моделирования* данных назначение *домена* атрибуту носит общий характер. Например, атрибут текстовый, числовой, бинарный, дата или "не определен". В последнем случае аналитик должен дать описание *домена*. На последующих стадиях тип *домена* конкретизируется, смысл понятия *домена* в физической модели ХД уже, чем его может понимать аналитик. Это связано с тем, что в рамках физической модели *домен* реализуется посредством механизма ограничения *домена*, СУБД не понимает неопределенных *доменов*.

Проектировщик должен тщательным образом изучить *домены* каждого атрибута с точки зрения их реализуемости в СУБД, с участием аналитиков внести в них изменения, если условие реализуемости не выполняется. При этом проектировщик руководствуется следующим:

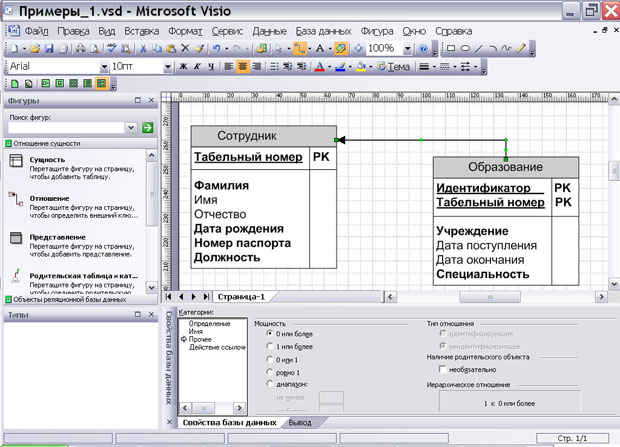
* для реализации реляционного ХД требуется использовать реляционную или объектно-реляционную СУБД, например, MS SQL Server 2008;
* в большинстве реляционных СУБД в качестве языка манипулирования и описания данных используется SQL, поддерживающий определенные стандарты, например, ANSI SQL-92.

*Отношение (связь)* сущностей на ER-диаграмме изображается линией, соединяющей эти сущности. *Отношение* читается вдоль линии либо слева направо, либо справа налево. На [рис. 6.6](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.6) представлено следующее отношение: каждая специальность по образованию должна быть зарегистрирована за определенным физическим лицом (персоной), физическое лицо может иметь одну или более специальностей по образованию.



**Рис. 6.6.**Представление отношения между двумя сущностями на ER-диаграмме

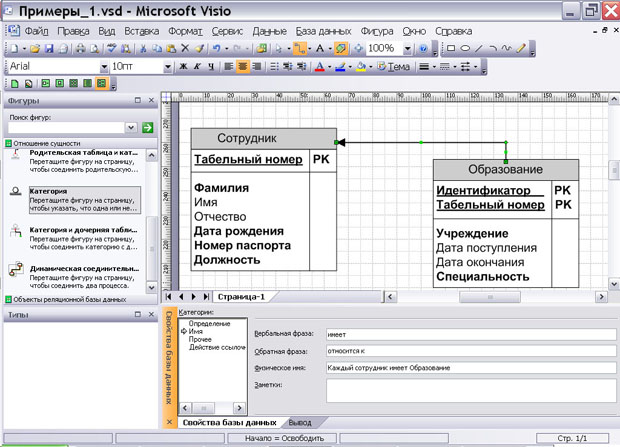
В MS Office Visio *имя связи*, *степень связи* (мощность) и *класс принадлежности сущности* к связи определяется на вкладке "Свойства базы данных", как показано на [рис. 6.7](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.7). Стрелка на линии связи указывает на *родительскую таблицу*.

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_07.jpg)

**Рис. 6.7.**Определение мощности связи отношения между сущностями "Сотрудник" и "Образование"

При выделении связей акцент делается на выявление их характеристик. Связь представляет собой взаимоотношение между двумя или более сущностями. Каждая связь реализуется через значения *атрибутов сущностей*, например, *экземпляр сущности* "Сотрудник" ([рис. 6.6](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.6)) связан с *экземпляром сущности* "Образование" по одинаковым значениям атрибутов *Табельный номер*. Другими словами, при создании связи в одной из сущностей, называемой дочерней сущностью, создается новый атрибут, называемый **внешним ключом** (Foreign Key, FK) (на [рис. 6.6](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.6) это атрибут *Табельный номер* ). Иногда атрибуты внешнего ключа обозначаются символом (FK) после своего имени.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой **Имя связи (Verb Phrase) – фраза, характеризующая отношение между родительской и дочерней сущностями**. *Имя связи* выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. На [рис. 6.8](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.8) показано присвоение связи имени.

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_08.jpg)

**Рис. 6.8.**Именование связи между сущностями "Сотрудник" и "Образование"

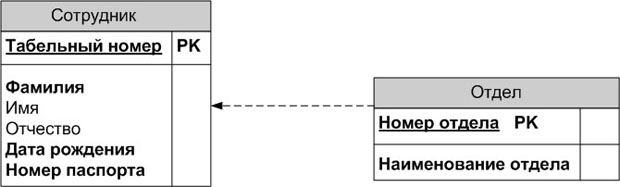
Существуют различные типы связей: идентифицирующая связь (identifying relationship) "один ко многим", связь "многие ко многим" и неидентифицирующая связь (non-identifying relationship) "один ко многим". С типами связей связывают и различные типы сущностей.

Различают два типа сущностей: **зависимые** (Dependent entity) и **независимые** (Independent entity). Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский *конец связи*) и зависимой (дочерний *конец связи*) сущностями.

Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на [рис. 6.8](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.8) информация о специальности не может быть внесена и не имеет смысла без информации о сотруднике, который имеет специальность по диплому об образовании. При установлении идентифицирующей связи (на рисунке непрерывная линия) атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности (непрерывная линия). Эта *операция дополнения* атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности такой атрибут считается внешним ключом.

Если модель создается при помощи CASE-средств, то при генерации схемы БД атрибуты первичного ключа получат признак NOT NULL, что означает невозможность внесения записи в таблицу "Сотрудники" без информации о табельном номере сотрудника.

При установлении неидентифицирующей связи ([рис. 6.9](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.9), пунктирная линия) дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей ([рис. 6.9](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.9)).

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_09.jpg)

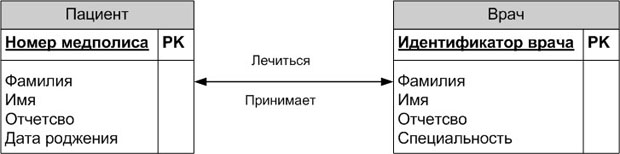
**Рис. 6.9.**Неидентифицирующая связь

*Экземпляр сущности* "Сотрудник" может существовать безотносительно к какому-либо *экземпляру сущности* "Отдел", т. е. сотрудник может работать в организации и не числиться в каком-либо отделе.

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи (см. [рис. 6.8](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.8)), неидентифицирующая – пунктирной (см. [рис. 6.9](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.9)).

**Связь "многие ко многим"** (many-to-many relationship) может быть создана только на уровне логической модели. На [рис. 6.10](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.10) показан пример определения связи "многие ко многим". Врач может принимать много пациентов, пациент может лечиться у нескольких врачей. Такая связь обозначается сплошной линией с двумя стрелочками на концах.

Связь "многие ко многим" должна именоваться двумя фразами – в обе стороны (в примере "принимает/лечится"). Это облегчает чтение диаграммы. Связь на [рис. 6.10](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.10) следует читать так: Врач <принимает> Пациента, Пациент <лечится> у Врача.

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_10.jpg)

**Рис. 6.10.**Связь "многие ко многим"

Как было указано выше, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей.

**Характеристическая** – зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.

**Ассоциативная** – сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

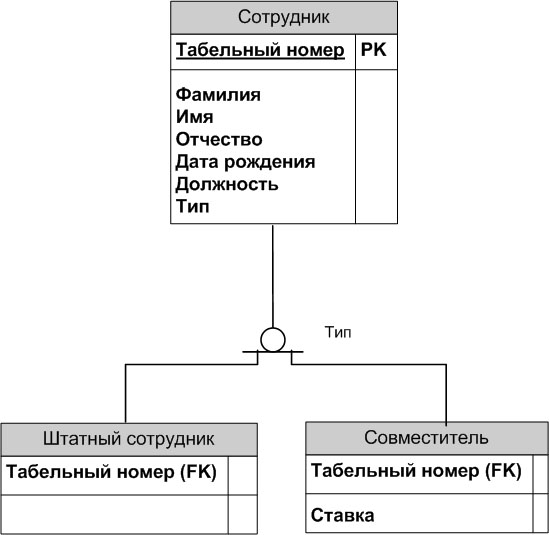
**Именующая** – частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

**Категориальная** – дочерняя сущность в иерархии наследования.

**Иерархия наследования** (*subtype* relationship), или иерархия категорий, представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (штатные служащие), и совместители. Из их общих свойств можно сформировать обобщенную сущность (родовой предок) "Сотрудник" (см. [рис. 6.11](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.11)), чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках) "Штатный сотрудник" и "Совместитель".

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты либо когда сущности имеют общие по смыслу связи (например, если бы "Штатный сотрудник" и "Совместитель" имели сходную по смыслу связь "работает в" с сущностью "Организация"), либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор (discriminator) – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой (атрибут Тип на [рис. 6.11](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.11)).

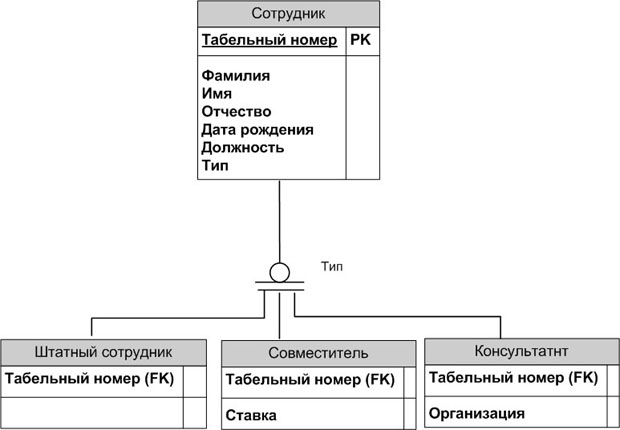


**Рис. 6.11.**Иерархия наследования. Неполная категория

Иерархии категорий делятся на 2 типа – **полные** и **неполные**. **В полной иерархии категорий** (Complete *subtype* relationship) одному экземпляру родового предка (сущность "Сотрудник", [рис. 6.12](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.12)) обязательно соответствует экземпляр в каком-либо потомке, т. е. в этом примере служащий обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет **неполной** (*Incomplete* *subtype* relationship). На [рис. 6.11](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.11) показана неполная категория – сотрудник может быть не только постоянным или совместителем, но и консультантом, однако сущность "Консультант" еще не внесена в иерархию наследования.

На [рис. 6.12](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.12) показан пример полной категории.

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_12.jpg)

**Рис. 6.12.**Иерархия наследования. Полная категория

Полная категория помечается символом, неполная –.

Возможна комбинация полной и неполной категорий. Помимо постоянных сотрудников и совместителей могут быть и консультанты, которые могут быть не отражены в иерархии (неполная категория), но каждый постоянный сотрудник либо мужчина, либо женщина (полная категория).

Рассмотрим возможные стадии построения иерархии наследования.

1. *Определение сущностей с общими (по определению) атрибутами*.

Предположим, в процессе проектирования созданы сущности "Штатный сотрудник" и "Совместитель" ([рис. 6.13](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.13)). Можно заметить, что часть атрибутов у этих сущностей ( *Фамилия, Имя, Отчество, Дата рождения, Должность* ) имеет одинаковый смысл.



**Рис. 6.13.**Сущности с общими по смыслу атрибутами

1. *Перенос общих атрибутов в сущность – родовой предок*. В случае обнаружения совпадающих по смыслу атрибутов следует создать новую сущность "Сотрудник" – родовой предок, и перенести в нее общие атрибуты ( *Фамилия, Имя, Отчество, Дата рождения, Должность* ).
2. *Создание неполной структуры категорий*. Создается категориальная связь от новой сущности – родового предка к старым сущностям-потомкам. Новая сущность дополняется атрибутом – дискриминатором категории ( *Тип* ) (см. [рис. 6.11](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.11)).
3. *Создание полной структуры категорий*. Проводится дополнительный поиск сущностей, имеющих общие по смыслу атрибуты с родовым предком. В примере это сущность "Консультант" ([рис. 6.14](https://intuit.ru/studies/courses/599/455/lecture/10163?page=2#image.6.14)).



**Рис. 6.14.**Дополнительная сущность с общими по смыслу атрибутами

Общие атрибуты переносятся в родового предка, и категория преобразуется в полную. Сущность "Консультант" не имеет атрибута *Должность*, поэтому в родовом предке значение этого атрибута в случае консультанта будет NULL. В зависимости от бизнес-правил атрибут *Должность* может быть перенесен обратно из родового предка в сущности-потомки "Штатный сотрудник" и "Совместитель".

1. *Комбинации полной и неполной структур категорий*. При необходимости создание иерархии категорий можно продолжить. Для каждого потомка может найтись сущность с общими атрибутами, тогда сущность-потомок становится родовым предком для новых потомков и т. д.

**Нормализация модели "сущность-связь"**

Имеются 3 подуровня логического уровня модели данных, отличающихся по глубине представления информации о данных:

* диаграмма "сущность-связь" (Entity Relationship Diagram, *ERD*);
* модель данных, основанная на ключах (Key *Based model*, KB);
* полная *атрибутивная* модель (Fully Attributed model, FA).

***Диаграмма "сущность-связь" (ER-диаграмма)*** отображает сущности и взаимосвязи, отражающие основные бизнес-правила *предметной области*, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. *Диаграмма "сущность-связь"* может отображать связи "многие ко многим" и не включать описание ключей. Как правило, *ER-диаграммы* используются для презентаций и обсуждения структуры данных с экспертами *предметной области*.

Модель данных, основанная на ключах, – более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют *предметной области*.

Полная *атрибутивная* модель – наиболее детальное представление структуры данных: представляет данные в *третьей нормальной форме* и включает все сущности, атрибуты и связи.

Рассмотрим теперь *процесс нормализации* данных, который сопровождает создание *полной атрибутивной модели*.

**Нормализация** – процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте.

*Процесс нормализации* сводится к последовательному приведению структуры данных к **нормальным формам** – формализованным требованиям к организации данных. Известно 6 нормальных форм:

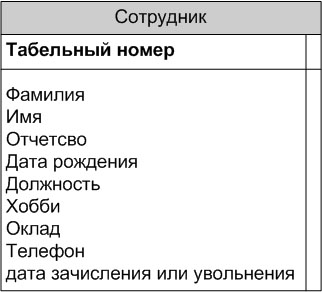
* *первая нормальная форма (1NF)* ;
* *вторая нормальная форма (2NF)* ;
* *третья нормальная форма (3NF)* ;
* *нормальная форма Бойса-Кодда* (усиленная *3NF*);
* *четвертая нормальная форма (4NF)* ;
* пятая нормальная форма (*5NF*).

На практике обычно ограничиваются приведением данных к *третьей нормальной форме*. В данном подразделе будут достаточно кратко рассмотрены первые три нормальные формы и, в качестве иллюстрации, *четвертая нормальная форма*. Для углубленного изучения нормализации следует рекомендовать книгу [Дейт].

Нормальные формы основаны на понятии функциональной зависимости (в дальнейшем будет использоваться термин "зависимость"). Приведем формальное определение для функциональной зависимости.

**Функциональная зависимость (FD)**. Атрибут B сущности E функционально зависит от атрибута A сущности E тогда и только тогда, когда каждое значение A в E связало с ним точно одно значение B в E, т. е. A однозначно определяет B.

**Полная функциональная зависимость**. Атрибут B сущности E полностью функционально зависит от ряда атрибутов A сущности E тогда и только тогда, когда B функционально зависит от A и не зависит ни от какого подмножества атрибутов A.



**Рис. 6.15.**Ненормализованная сущность "Сотрудник"

На [рис. 6.15](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.15) в сущности "Сотрудник" значения атрибутов *Фамилия*, *Имя* и *Отчество* однозначно определяются значением атрибута *Табельный номер*, т. е. атрибуты *Фамилия, Имя* и *Отчество* зависят от атрибута *Табельный номер*.

Функциональные зависимости определяются бизнес-правилами *предметной области*. Так, если оклад сотрудника определяется только должностью, то атрибут *Оклад* зависит от атрибута *Должность* ; если оклад зависит еще, например, от стажа, то такой зависимости нет. В нижеследующих примерах будем считать для определенности, что такая зависимость есть.

Рассмотрим нормальные формы.

*Первая нормальная форма (1NF)* Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения.

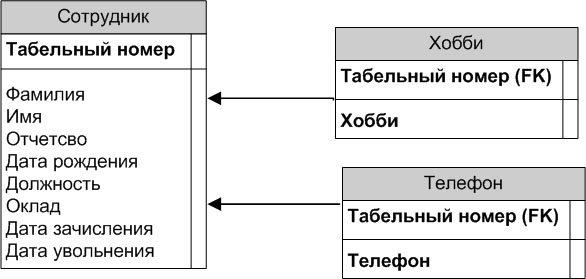
Среди атрибутов не должно встречаться повторяющихся групп, т. е. несколько значений для каждого экземпляра. На [рис. 6.15](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.15) атрибуты *Телефон* и *Хобби* могут представлять повторяющиеся группы и тем самым нарушать требования *первой нормальной формы*. Сотрудник может иметь несколько рабочих телефонов или иметь несколько увлечений (рыбалка, охота, плавание). Добавление в сущность нескольких атрибутов для представления значений повторяющейся группы не является решением проблемы, поскольку в большинстве случаев нельзя заранее точно определить, сколько таких значений может быть.

Нарушением требований нормализации является хранение в одном атрибуте разных по смыслу значений. На [рис. 6.15](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.15) атрибут *Дата зачисления или увольнения* хранит информацию как о зачислении, так и об увольнении сотрудника. Если хранится только одно значение, то невозможно понять, какая именно дата внесена. Если внести атрибут-*признак типа* даты, тип можно будет определить, но останется возможность хранения только одной даты для каждого сотрудника.

Для приведения сущности к *первой нормальной форме* следует:

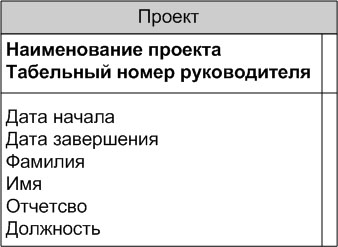
* разделить сложные атрибуты на атомарные;
* создать новую сущность;
* перенести в нее все "повторяющиеся" атрибуты;
* выбрать возможный ключ для нового PK (или создать новый PK);
* установить идентифицирующую связь от прежней сущности к новой, PK прежней сущности станет внешним ключом (FK) для новой сущности.

На [рис. 6.16](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.16) показана сущность "Сотрудник", приведенная к *первой нормальной форме*.



**Рис. 6.16.**Сущность "Сотрудник", приведенная к первой нормальной форме

*Вторая нормальная форма (2NF)* Сущность находится во *второй нормальной форме*, если она находится в *первой нормальной форме* и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа). *Вторая нормальная форма* имеет смысл только для сущностей, имеющих сложный первичный ключ.

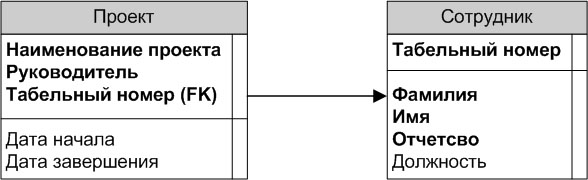


**Рис. 6.17.**Сущность "Проект"

Предположим, что сущность "Проект" содержит информацию о проекте, которым руководит сотрудник, причем информация содержится как непосредственно о проекте, так и о руководителе проекта ([рис. 6.17](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.17)). Атрибуты *Фамилия, Имя, Отчество* и *Должность* зависят только от атрибута *табельный номер руководителя*, но вовсе не от атрибута *Наименование проекта*. Другими словами, имеется зависимость только от части ключа.

Для приведения сущности ко *второй нормальной форме* следует:

* выделить атрибуты, которые зависят только от части первичного ключа, создать новую сущность;
* поместить атрибуты, зависящие от части ключа, в их собственную (новую) сущность;
* установить идентифицирующую связь от прежней сущности к новой ([рис. 6.18](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.18)).



**Рис. 6.18.**Сущность "Проект", приведенная ко второй нормальной форме

*Вторая нормальная форма* позволяет избежать следующих аномалий при выполнении операций:

* обновления (UPDATE). Происходит дублирование данных о сотруднике, если он руководит несколькими проектами. Если данные о сотруднике изменяются, необходимо менять несколько записей (по числу ведомых проектов);
* вставки (INSERT). Невозможно ввести данные о сотруднике, если он в данный момент не руководит проектами;
* удаления (DELETE). Если сотрудник временно прекращает руководство проектами, данные о нем теряются.

На [рис. 6.18](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.18) показана сущность "Проект", приведенная ко *второй нормальной форме*.

*Третья нормальная форма (3NF)*. Сущность находится в *третьей нормальной форме*, если она находится во *второй нормальной форме* и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

На [рис. 6.16](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.16) сущность "Сотрудник" находится во *второй нормальной форме* (имеется только один атрибут первичного ключа, поэтому не может быть зависимости неключевых атрибутов от части ключа), но неключевой атрибут *Оклад* зависит от другого неключевого атрибута – *Должности*.

Для приведения сущности к *третьей нормальной форме* следует:

* создать новую сущность и перенести в нее атрибуты с одной и той же зависимостью от неключевого атрибута;
* использовать атрибут (атрибуты), определяющий (определяющие) эту зависимость, в качестве первичного ключа новой сущности;
* установить неидентифицирующую связь от новой сущности к старой ([рис. 6.19](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.19)).

[](https://intuit.ru/EDI/20_07_20_2/1595197216-9970/tutorial/632/objects/6/files/05_19.jpg)

**Рис. 6.19.**Сущность "Сотрудник", приведенная к третьей нормальной форме

В *третьей нормальной форме* каждый *атрибут сущности* зависит от ключа, от всего ключа целиком и ни от чего другого, кроме как от ключа.

*Третья нормальная форма* также позволяет избежать ряда аномалий.

* Обновление (UPDATE). Происходит дублирование данных об окладе, если должность занимают несколько сотрудников. Если оклад, соответствующий должности, меняется, необходимо менять несколько записей (по числу сотрудников на одной должности).
* Вставка (INSERT). Невозможно ввести данные об окладе, соответствующем должности, если в данный момент нет сотрудника, занимающего эту должность.
* Удаление (DELETE). В случае удаления из таблицы сотрудника, занимающего уникальную должность, данные об окладе теряются.

*Четвертая нормальная форма (4NF)* требует отсутствия многозначных зависимостей между атрибутами.

В примере на [рис. 6.20](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.20) (слева) преподаватель читает лекции по нескольким предметам и курирует несколько групп студентов. Одна группа студентов может изучать несколько предметов, одному предмету могут обучаться несколько групп студентов. Имеется многозначная зависимость между атрибутами *Предмет* и *Группа*. При этом возможна аномалия: если у преподавателя появляется новая группа, приходится добавлять несколько записей, по числу читаемых предметов.

Для приведения сущности к *четвертой нормальной форме* следует создать новую сущность и перенести атрибуты с многозначной зависимостью в разные сущности ([рис. 6.20](https://intuit.ru/studies/courses/599/455/lecture/10163?page=3#image.6.20), справа). Связь между новыми сущностями при этом устанавливать нельзя, поскольку в результате миграции атрибутов внешних ключей атрибуты с многозначной зависимостью вновь окажутся в одной сущности. Ссылочную целостность в этом случае следует поддерживать при помощи триггеров.



**Рис. 6.20.**Иллюстрация четвертой нормальной формы

**Резюме**

Метод моделирования "*сущность-связь*" был предложен С. Ченом в 1976 году. Ряд исследователей разработали несколько графических нотаций для представления элементов модели. Проектировщик ХД может выбрать графическую нотацию по своему вкусу.

Применение метода моделирования "*сущность-связь*" помогает проектировщикам создать *логическую модель предметной области*, не зависимую от программно-аппаратной реализации. Этот метод используется как при моделировании *предметных областей* OLTP-систем, так и при моделировании *предметных областей* BI-систем. *Знание* этого метода помогает проектировщику ХД быстрее установить логические связи между моделями *БД* OLTP-систем масштаба организации и моделями ХД BI-систем.

Независимо от выбранной нотации, действия проектировщика ХД при ER-моделировании сводятся к следующему алгоритму.

Для каждой *сущности предметной области* *базы данных* необходимо:

* получить список *атрибутов сущности* ;
* определить функциональные зависимости;
* определить возможные ключи, в частности, рассмотрев *уникальный идентификатор сущности* ;
* выполнить нормализацию сущности;
* назначить первичные ключи новых, полученных в результате нормализации сущностей;
* сформировать бизнес-правила поддержки целостности сущности.

Для каждой связи между сущностями необходимо:

* определить мощность связи;
* определить обязательность вхождения сущности в связь;
* разрешить связи "многие ко многим";
* назначить первичные ключи ассоциативных сущностей, исходя из уникального идентификатора связи и процедуры миграции ключей при нормализации;
* определить неключевые атрибуты ассоциативных сущностей, если они необходимы;
* сформировать бизнес-правила поддержки целостности связей;
* документировать логическую модель реляционной базы данных;
* проверить логическую модель реляционной базы данных;