Диаграммы сущность – связь.

Бессарабов Н.В.
bes@fpm.kubsu.ru
2018 г.

Виды моделей данных

Уже упоминалось, что обычно выделяют модели:

- Концептуальные (инфологические) – выделяют сущности/концепты и связи между ними.
- Логические (даталогические) – модели для реализации в некотором классе СУБД.
- Физические – модели для конкретной СУБД.

В базах данных принято инфологические модели данных называть семантическими. Как будет упомянуто далее, это не совсем корректно.

Наиболее известные инфологические модели:

- “Сущность - связь” (ERM и расширенная модель EER).
- “Объект-роль” (ORM).

Мы будем рассматривать только ER и отчасти EER модели.

Замечание 1: У нас вынужденно узкий подход к разработке информационных систем, который следовало бы начинать с модели бизнес-процессов, например, в стандарте BPM.

Замечание 2: Семантика моделей данных, включающих онтологии и темпоральные данные, описывается ещё в интеграционном стандарте ISO 15926, использующем языки OWL и SPARQL. Мы его не изучаем.

Что моделируем

- Сущности вещные/предметные, может быть изменяющиеся со временем
- Сущности-процессы

Модели данных, которые называют семантическими

Инфологические модели принято называть семантическими. Среди них наиболее известна модель “сущность - связь”.

Семантическая модель данных обычно определяется как схема, показывающая соотношения хранимых символов (наборов записей, сущностей) с реальным миром.

Два больших “но”:

1)Ещё в 1988 г. Э.Кодд указывал, что ярлык “семантическое” не должен интерпретироваться в каком-либо абсолютном смысле. Дело в том, что семантики существуют в любой модели данных, но объём и содержание их могут сильно отличаться.

2)А почему прагматика обычно не рассматривается в семантических моделях? Она что, определена однозначно?

Пример: Как интерпретировать обычную таблицу?

Ответы (основанные на трёх разных прагматиках):

- Как набор записей.
- Как многомерный куб.
- Как импликацию (продукцию).

Семантические модели данных зачем они?

На начальной стадии создания приложения (анализ бизнеса) необходимо иметь модель предметной области, обеспечивающую наглядное и, по возможности, неформальное описание всех особенностей бизнеса известных постановщику.

При этом отбрасывание деталей, которые “не ложатся” на модели данных, применяемые на последующих стадиях реализации проекта, может привести к существенному искажению постановки задачи.

На этапе анализа полноту сведений следует предпочесть возможности их формального описания.

В рамках семантической модели создается концептуальная схема базы данных, которая вручную или автоматизированно (но обычно не автоматически) преобразуется в логическую и физическую схемы базы допустимую в рамках моделей данных, реализуемых на следующих стадиях жизненного цикла проекта – проектировании, разработке и сопровождении.

Замечание: Моделей данных без семантики не бывает. Пока же будем считать, что “семантическими” называют инфологические модели, в которых ”больше” семантики интерпретируемой человеком, чем семантики, доступной СУБД.

Семантическая модель “Сущность-Связь” (Entity-Relationship)

Наиболее известна семантическая модель **“сущность – связь”** (“**e**ntity - **r**elationship” -- ER) предложенная Питером Пин Шен Ченом (Peter Chen) в 1976 г.

Три основных понятия ER-модели: **сущность, связь, атрибут.** У сущности есть имя и атрибуты, у атрибута имеется имя и значение. Связи также имеют имя и атрибуты.

Замечание: к семантическим моделям данных следует отнести диаграммы классов модели UML, схемы Баркера, схемы Закмана и др.

Определены четыре **уровня представления информации и данных**, в которых рассматриваются все модели данных.

Четыре уровня представления моделей данных (по Чену)

1. Информация об объектах и связях предметной области (ПО).
2. Данные, описывающие объекты и связи предметной области (структурированная информация о ПО).
3. Структуры данных, не зависящие от способа доступа (то есть не связанные с поиском, индексацией и т. д.).
4. Структуры данных зависящие от способа доступа.

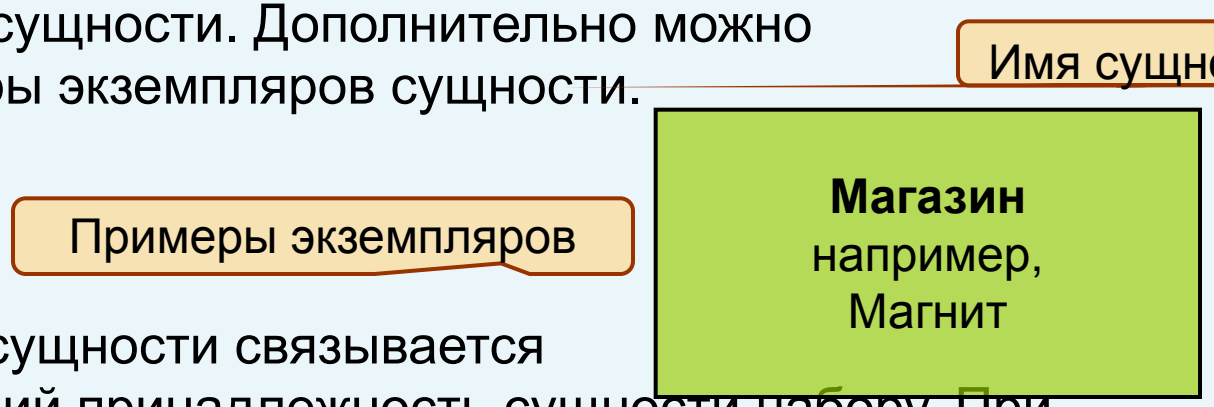
Забегая вперёд заметим, что реляционная модель относится к уровням 2 и 3. Сетевая и иерархическая модели, в том виде как они существовали 20 лет назад, работают в основном на уровне 4 и отчасти 3. UML это уровни 1, 2 и отчасти 3, но UML далеко выходит за рамки описания данных. IDEF1x работает на уровнях 2, 3 и 4.

Замечание: Нетрудно заметить, что приведенная классификация устарела, так как в ней не использованы семантики и прагматики данных.

Сущность и набор сущностей

Сущность это воображаемый объект или процесс, информация об экземплярах которого должна сохраняться в своем наборе записей.

Сущность определяет **тип**, а не экземпляр. На ER- диаграммах сущность представляется прямоугольником, в котором обязательно указывается имя сущности. Дополнительно можно указывать примеры экземпляров сущности.



С каждым типом сущности связывается предикат, задающий принадлежность сущности набору. При определении типа сущности необходимо гарантировать, что экземпляры сущности различимы один от другого. Это требование аналогично требованию отсутствия записей-дубликатов или кортежей в реляционных таблицах.

Замечание 1: Сущности могут быть вещными, или процессными.

Замечание 2: В общем случае сущности бизнеса описываются предикатами более сложными чем в логике первого порядка. Например, в них могут существовать разносортные атрибуты, в том числе, атрибуты-ресурсы.

Связи

Связь – это тѝповое понятие, устанавливающее правила связывания сущностей. Каждый экземпляр типа связи, устанавливается между экземплярами типа сущности. Может существовать рекурсивная связь между типом сущности и им же самим (как бы его дубликатом).

Пока рассматриваем только бинарные связи, устанавливаемые между двумя типами сущностей. О связях с большей арностью поговорим позднее.

Концы бинарной связи в ER-модели характеризуется:

- **степенью** конца связи (сколько экземпляров данного типа сущности должно присутствовать в каждом экземпляре данного типа связи);
- **обязательностью** связи (т. е. любой ли экземпляр связываемой сущности должен участвовать в некотором экземпляре данного типа связи).

В продвинутых системах могут использоваться:

- имена **ролей** (имена концов связи), определяющие функции связи по отношению к связываемым сущностям;

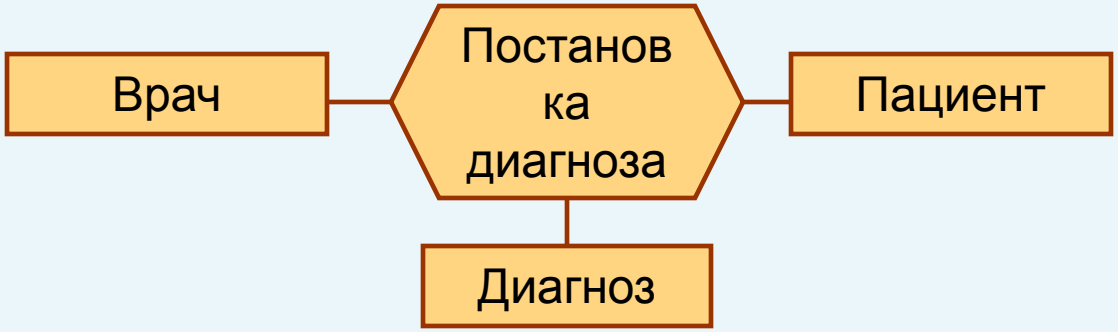
Обозначения и примеры связей

Обозначения:

Бинарная связь:

Несколько связей между двумя сущностями:

Тернарная связь:



Атрибуты, значения, наборы значений и типы

Атрибут это свойство сущности или связи, получаемое путем наблюдения или измерения. Информацию об экземпляре сущности выражают набором нескольких пар “атрибут – значение”, например:

Атрибут Значение

<Имя: ‘Петя’>

Атрибут принимает одно или несколько значений из некоторого набора.
Пример: В анкете предлагается подчеркнуть один или несколько предусмотренных ответов в качестве значения атрибута.
Заполненная строка выглядит так:
“Как часто Вы посещаете лекции по базам данных (нужное подчеркнуть): часто, редко, довольно часто, довольно редко, по настроению, в дождливую погоду, в конце семестра”.

Значения атрибутов обычно принадлежат одному типу, но возможны бестиповые атрибуты.

Атомарность атрибутов

Свойство атомарности атрибута корректно определяется только в рамках выбранной семантики. Атрибут атомарный, если его компоненты не имеют смысла или не должны быть выделены в этой семантике. Атрибут, атомарный в одной семантике, может быть неатомарным в другой.

Если, например, мы обещаем никогда не интересоваться отдельно фамилией именем и отчеством, то атрибут ФИО, то есть “Фамилия, имя, отчество” атомарный. Если эти компоненты нам нужны, то ФИО неатомарный атрибут.

Вопрос “будут ли какие-нибудь компоненты атрибута использоваться по отдельности?” как раз позволяет определиться с атомарностью. Если будут, то атрибут следует разделить на осмысленные компоненты и, может быть продолжить этот процесс.

Замечание: В классических моделях данных и простых СУБД атрибуты должны быть атомарными. Однако, в современных СУБД

над основной моделью имеется второй слой, позволяющий работать с регулярными выражениями и/или XML, то есть с компонентами того, что на нижнем уровне считается

Связи также имеют атрибуты

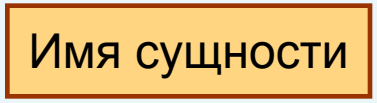
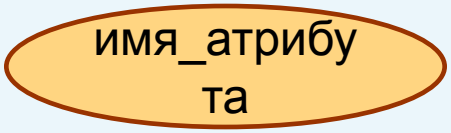
Выделим две разновидности атрибутов связей:

1. Атрибуты **привязки**, через которые осуществляется привязка к связываемым сущностям.
2. Атрибуты, определяющие свойства сущностей, проявляющиеся только при наличии связи. Такие атрибуты называют **эмерджентными**.

Пример: Сущности “Работник” и “Проект” со связью “Проект - Работник”, содержащей атрибуты связи “Номер_работника”, “Номер_проекта” и эмерджентный атрибут свойства связи “Ресурс_времени”. Смысл последнего атрибута “Плановые затраты времени работника с указанным номером_работника на работу в рамках проекта с указанным номером_проекта”.

Примеры связей зависящих от времени (на доске)

Сводка обозначений

Обозначение	Значение
	Тип, определяющий набор независимых сущностей
	Тип, определяющий набор зависимых сущностей
	Атрибут
	Ключевой атрибут
	Тип, определяющий набор бинарных связей

Условность разделения на сущности, связи и атрибуты

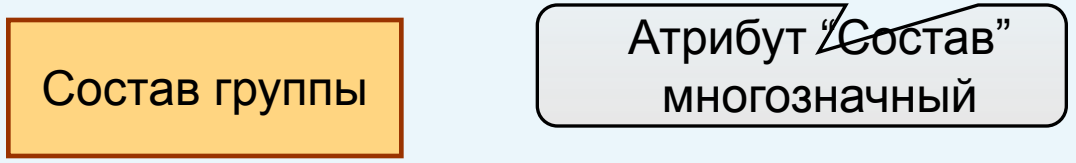
Разделение на сущности, связи и атрибуты условно.

Пример: То, что студент должен относиться к какой-нибудь учебной группе можно выразить:

1. Как связь

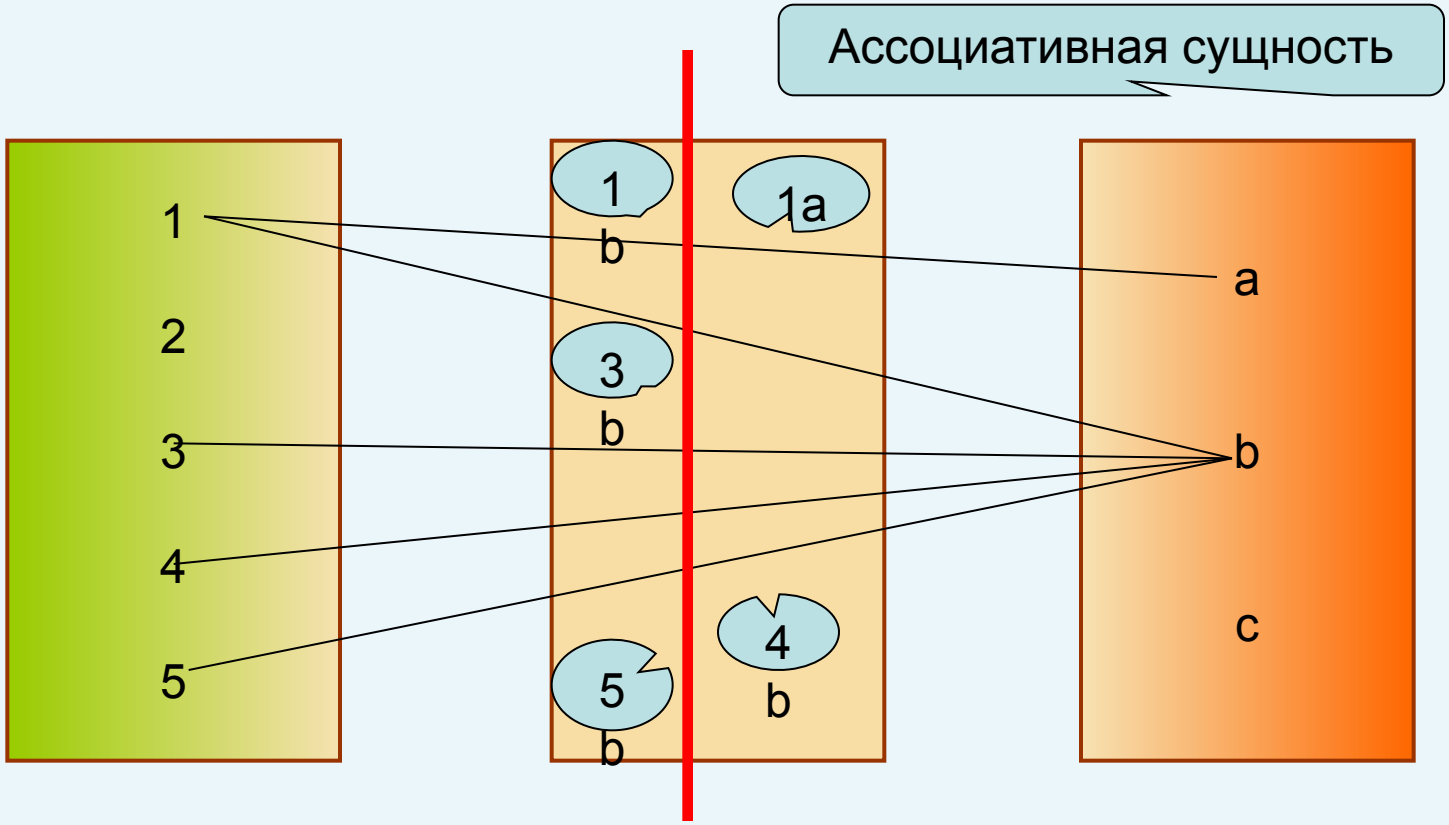
2. Как пара атрибутов

3. Как сущность



Замечание: Не следует делать вывод о том, что выбор сущностей произволен и нет предпочтительного варианта.

Разрешение связей вида “многие-ко-многим”



Значения элементов ассоциативной сущности:
1a, 1b, 3b, 4b, 5b

Связи многие-ко-многим не используются в реализациях баз данных. Вводим фиктивную дополнительную сущность и две связи один-ко-многим.

Неопределенные значения (Null)

Null это универсальное (бестиповое, не зависящее от типа данных) значение, показывающее, что необходимое значение не введено в рассматриваемой записи. Позволяет различать пустые значения (например, пустую и потому неотображаемую строку) и отсутствующие значения (нет никакого значения, даже пустого). Помните, что пустое значение (нуль или пустая строка), часто задаваемое по умолчанию, это не null.

При обработке данных с неопределенными значениями необходимо пользоваться одной из трехзначных логик.

Неопределенные значения существуют в любых моделях данных. Их нет в языках программирования общего назначения.
Не путайте null с пустыми ссылками в этих языках.

Null **это не** **Пустая ссылка**

Правило: любые алгебраические операции (сложение, умножение, конкатенация строк и т.д.) с операндом null должны давать также неопределенное значение null.

Какие сорта атрибутов могут использоваться?

Некоторые сорта атрибутов:

1. Атрибуты, однозначно определяющие экземпляр сущности. Это ключи (первичные, уникальные, альтернативные). Если с этой целью необходимо использовать несколько атрибутов, то образуется блок атрибутов.
2. Неуникальные ключи.
3. Атрибуты состояния.
4. Атрибуты ресурсов, представляющих какую-то часть сущности-ресурса доступную экземплярам другой сущности (Я и 100000 р. в кармане).
5. Темпоральные атрибуты задающие моменты или интервалы времени.
6. Атрибуты допускающие отсутствие значения (Null).
7. Атрибуты связанные по смыслу. В таких системах один из атрибутов доопределяет другой. Например, “Вес” и “Единица измерения веса”. Образуется блок атрибутов.
8. Необязательные атрибуты. В полуструктурированных данных часть атрибутов обязательна, а другие атрибуты, не обязательные, могут существовать только у некоторых экземпляров сущности.

Замечание 1: Атрибуты могут объединяться в блоки. Например, составной (конкатенированный) ключ

О ключах в ERWin и других инструментах(1/2)

Изображение каждой сущности разделяется горизонтальной линией на верхнюю часть (**ключевую область**), в которой расположены **ключевые атрибуты (поля)** (**первичные** (PK) и, может быть, **внешние ключи** (FK)) и нижнюю часть (**область данных),** где расположены **неключевые атрибуты** и, может быть атрибуты **внешних** и **альтернативных** (AK) ключей.

Свойства первичного ключа:

- Уникальным образом идентифицирует экземпляр.
- Не использует NULL значений (подумайте почему так?).
- Не изменяется со временем.

Экземпляр идентифицируется при помощи первичного ключа.

При изменении ключа идентифицируемый им экземпляр считается другим (не может считаться тождественным старому экземпляру).

Замечание: Если значения поля или нескольких полей позволяют выбрать более одного экземпляра, принято говорить о существовании **неуникального ключа**. На него может быть создан неуникальный индекс который может ускорить выборку таких групп экземпляров.

О ключах в ERWin и др. инструментах (2/2)

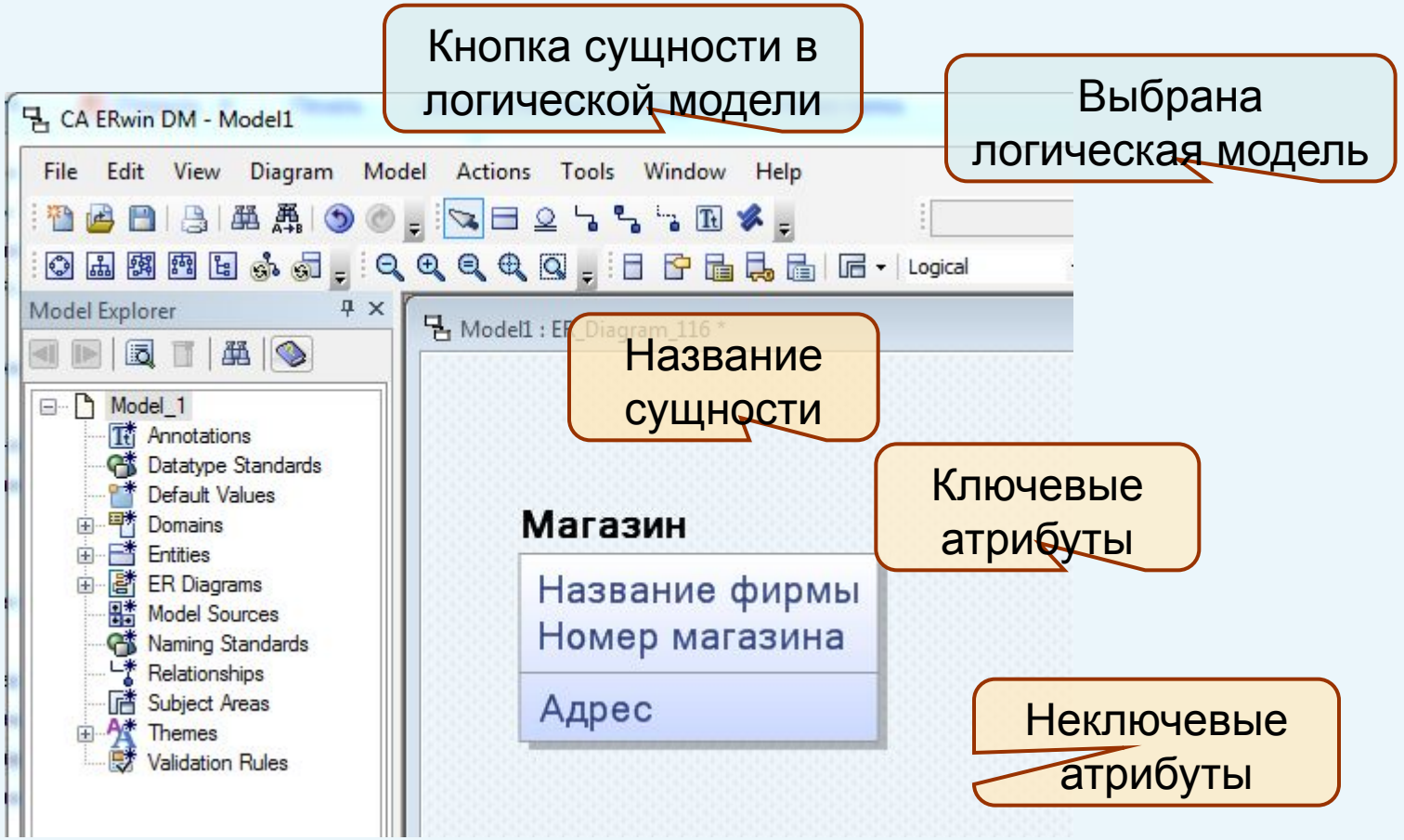
Свойства уникального ключа

Уникальные ключи (Unique Key) отличаются от первичных тем, что в них могут использоваться неопределенные значения null.

Внешние ключи

Если сущности связаны, то связь может передать ключ (набор ключевых атрибутов) дочерней сущности. Эти переданные атрибуты оказываются внешними ключами. Передаваемые атрибуты принято называть **мигрирующими**.

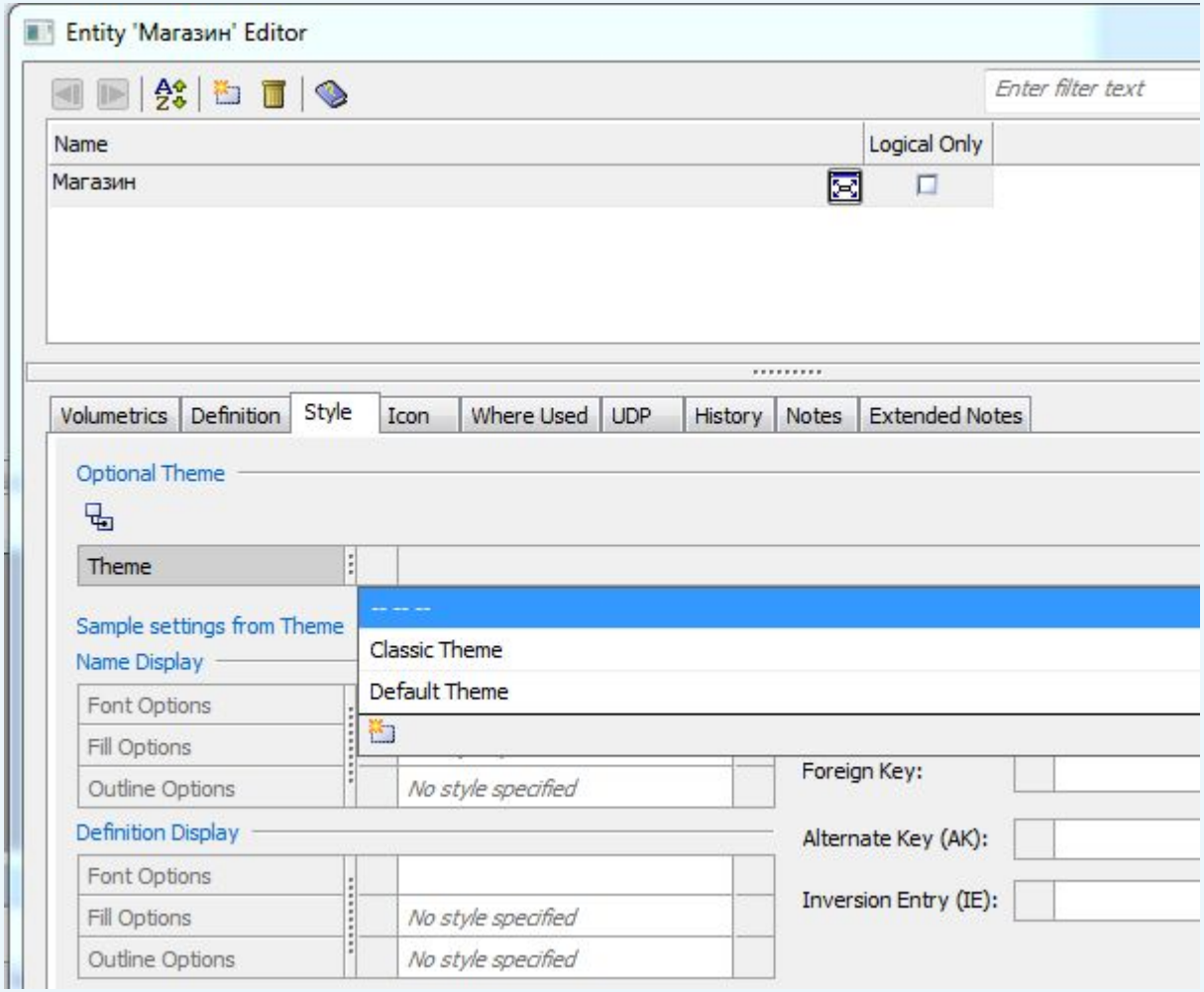
ER-диаграммы в ERWin (1/2)



Ссылка для скачивания ERwin Data Modeler Community Edition r9.5:
http://www.itshop.ru/Computer-Associates-CA/CA-ERwin-Data-Modeler-Community-Edition-R8-SKAChAT-BESPLATNO/l4t1i162474

ER-диаграммы в ERWin (2/2)

Запись определений (Definition), примечаний (Notes, Extended Notes) свойств, определенных пользователем (User Definied Properties --UDP) и проч.



Связи в ERWin

Связи между сущностями обозначаются линиями, может быть снабжёнными дополнительными символами на концах.

Связи именуются глаголами или словосочетаниями с глаголами, которые показывают, как соотносятся сущности между собой.

В простых схемах имена связей могут не назначаться и не проставляться

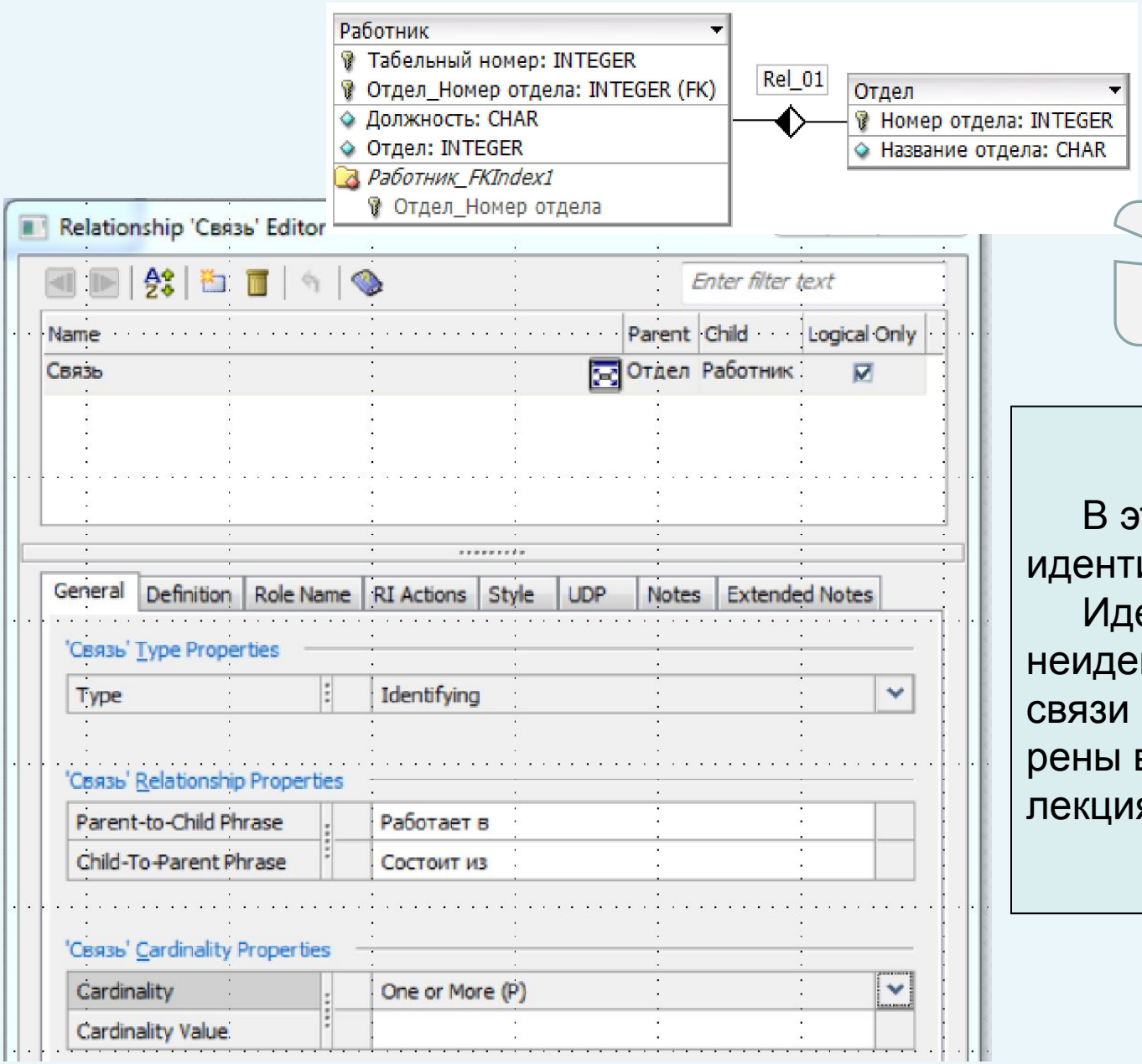
Пример: (связь неидентифицирующая)

Если бы инструменты позволяли, то можно было бы описывать связи более детально,

например, так:

Штриховая линия на конце связи "Иметь" задает модальность, то есть возможность – "Может иметь".

Создание связей в ERWin и DBDesigner

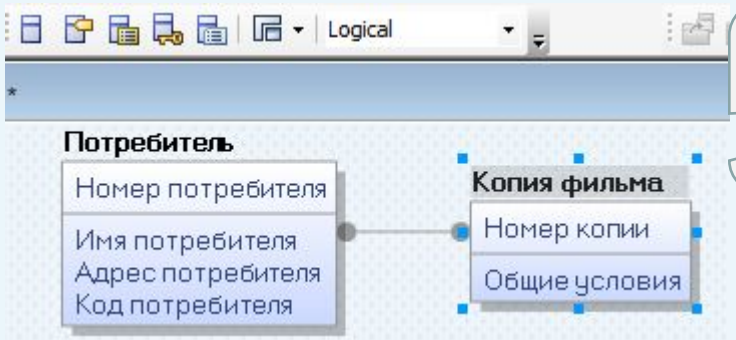


Это логическая схема

В этом примере связь идентифицирующая.

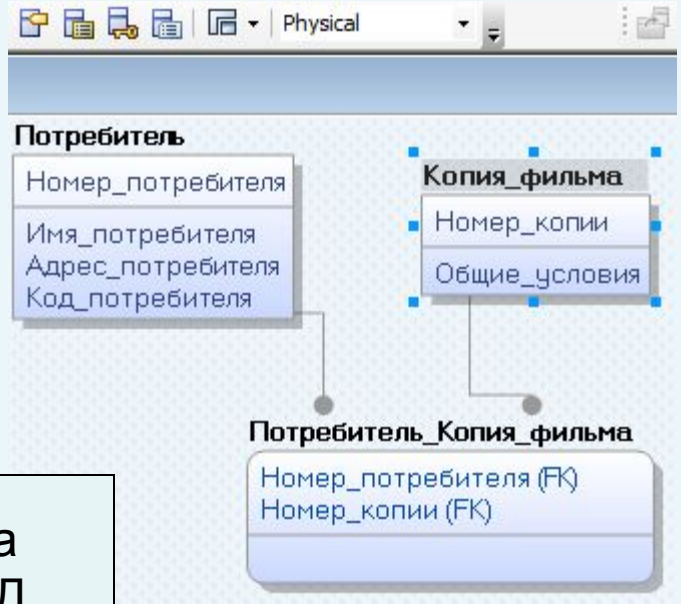
Идентифицирующая и неидентифицирующая связи подробно рассмотрены в следующих лекциях

Пример связи вида “многие-ко-многим” и её разрешение в ERWin



Создаём диаграмму со связью N:M на логическом уровне

И переводим её на физический уровень



Появляется ассоциативная таблица и две связи один-ко-многим

Вспомните, что связи типа “многие-ко-многим” в СУБД не реализуются

Сильные и слабые сущности (1/2)

Если при выборе экземпляра сущности С1 необходимо как-то указать на его связь с экземпляром другой сущности С2, то сущность С1 будет считаться **слабой**. При этом, С2 ***не обязательно сильная сущность.***

В первичный ключ такой сущности С1 обязательно включается внешний ключ, заимствуемый из С2.

Пример: Если считается, что указание на игрока обязательно должно сопровождаться указанием на имя его команды, то сущность Игрок слабая. В ERWin'е зависимые сущности изображаются прямоугольниками со скругленными краями.

До установления связи

И после

Сильные и слабые сущности (2/2)

А если посчитать, что все игроки никак не связаны с командой? А если игроки сами по себе, но бывают в команде?

Обозначения концов связей:
Модальность “может” - - - - - -
Модальность “должен” ----------

Построение описания связи: <экземпляр сущности1> <модальность_связи> <название_связи> <тип_связи> <экземпляр сущности2>. Пример:

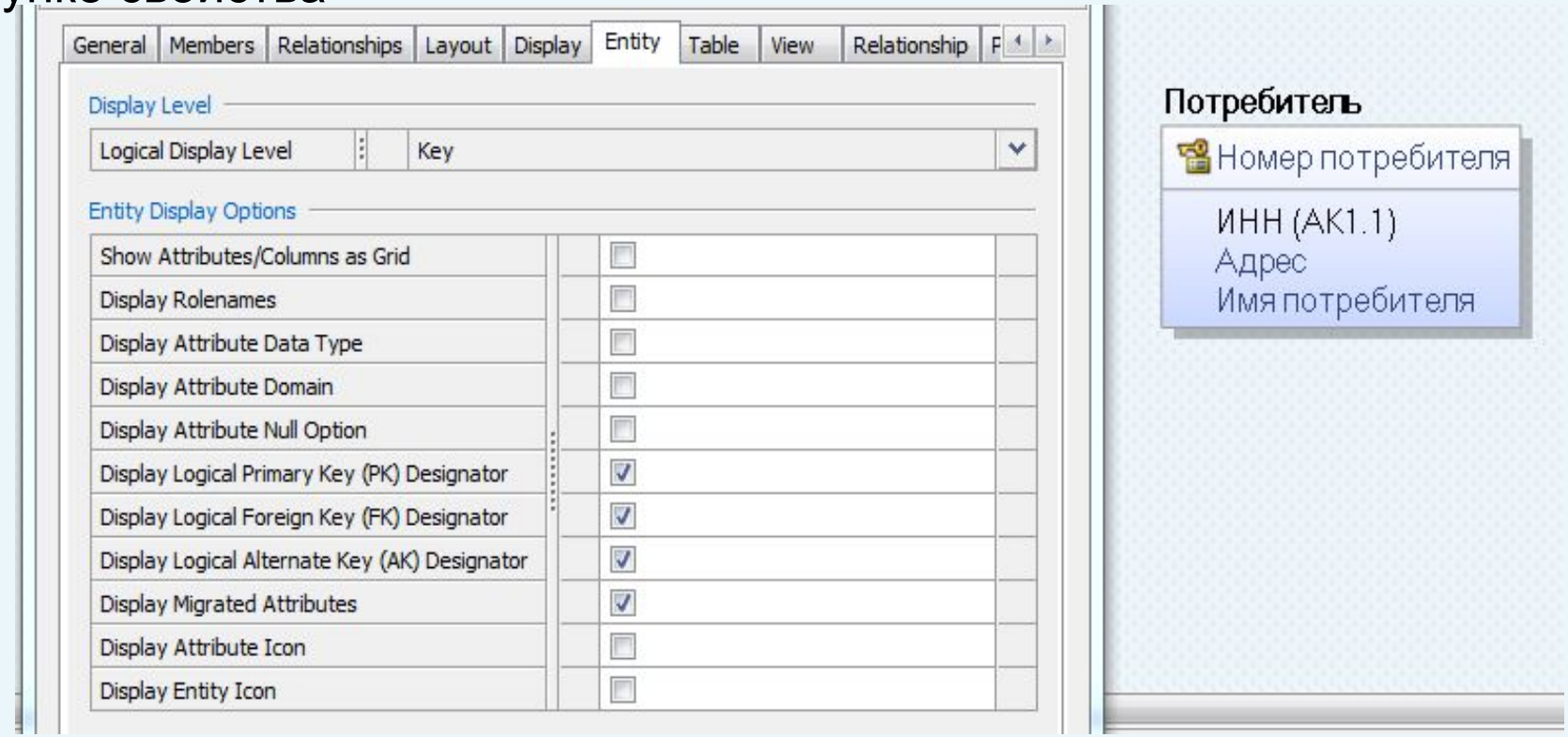
Слева направо: "каждый сотрудник может иметь несколько детей”
Справа налево: "Каждый ребенок обязан принадлежать ровно одному сотруднику”
Уточните семантику!! Какое ограничение накладывается на сотрудников?
Исправьте сами!

Обратите внимание, на то, что с разделением сущностей на сильные и слабые нельзя разобраться не выходя в предметную область

Альтернативные ключи

Потенциальные ключи, не использующиеся как первичные, могут быть определены как **альтернативные ключи** и записаны в секции данных модели с символом (AKn.m), где n.m – номер альтернативного ключа в формате “номер_сущности”. “номер_ключа”.

Для того, чтобы высвечивались обозначения этих ключей и пиктограммы PK, необходимо, нажав правой кнопкой мыши по панели, выбрать Properties в панели свойств редактора установить указанные на рисунке свойства



Зачем нужна и когда используется модель сущность-связь?

ER-диаграммы предназначены для разработки концептуальных моделей, но инструментарий обычно позволяет создавать логические и физические модели.

Чтобы уточнить ответ на вопрос “когда используется?” мы в очередной раз выйдем за общепринятые рамки предмета баз данных. Далее весьма упрощенно рассмотрим жизненный цикл информационной системы, частью которой обычно бывает база данных, выделив всего две понятные начинающим модели этого цикла. Вы увидите, что модель сущность - связь создается на начальных стадиях – анализе и проектировании. По созданной физической модели автоматически генерируются скрипты создающие базу.

Всё начинается с описания бизнес-процессов, а затем к ним привязывают обеспечивающие структуры данных, описываемые ER-диаграммами. Но в этом курсе мы должны рассматривать только базы данных. Остальную информацию вы можете найти в материалах курса “CASE-средства”.

Другие семантические модели

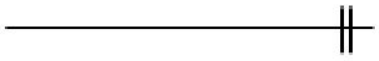
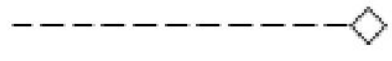
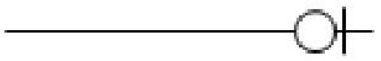
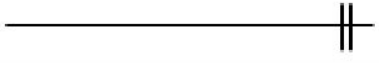
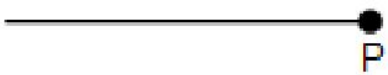
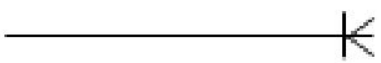
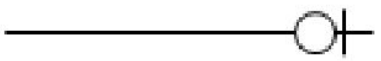
- UML
- Диаграммы Баркера
- Семантические сети
- Концептуальные графы
- Объектная семантическая модель (Крёнке)
- BPM
- Модели ISO 15926
- …….

Для описания возможного набора моделей используют диаграммы Захмана.

Использование семантических моделей позволяет существенно расширить фиксируемую в модели семантику данных, но не исчерпывает все возможные семантики.

Замечание: подробности см. в курсе “Семантические модели данных”

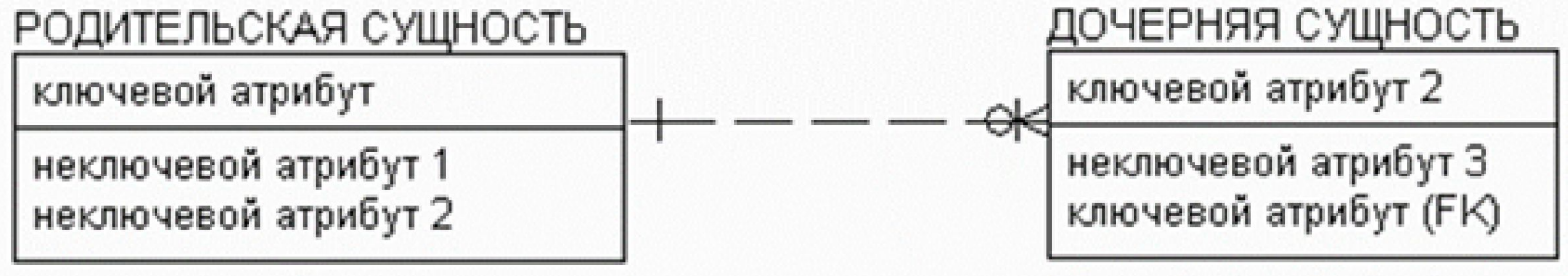
Нотации IDEF1X и IE для логической модели

Обязательность связи	Мощность связи справа	Внешний вид	
		IDEF1X	IE
Со стороны родительской таблицы			
Обязательная	1		
Необязательная	0 или 1		
Со стороны дочерней таблицы			
	1		
	0 .. ∞		
	1 .. ∞	 P	
	0 или 1	 Z	

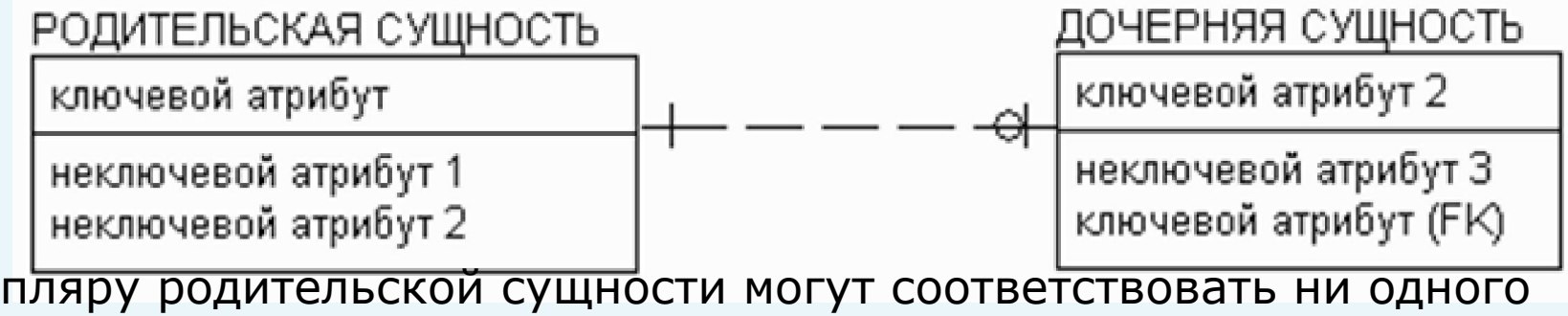
IE - Information Engineering (информационное проектирование). Слабые и сильные сущности изображают одинаково – прямоугольниками. Обозначения:

- o – ноль;
- | – один;
- || – один и только один (строго один). Обычно со стороны родительск. таблицы;
-  – много (больше 0). Этот знак иногда называют «вороньей лапкой» (crow's foot).

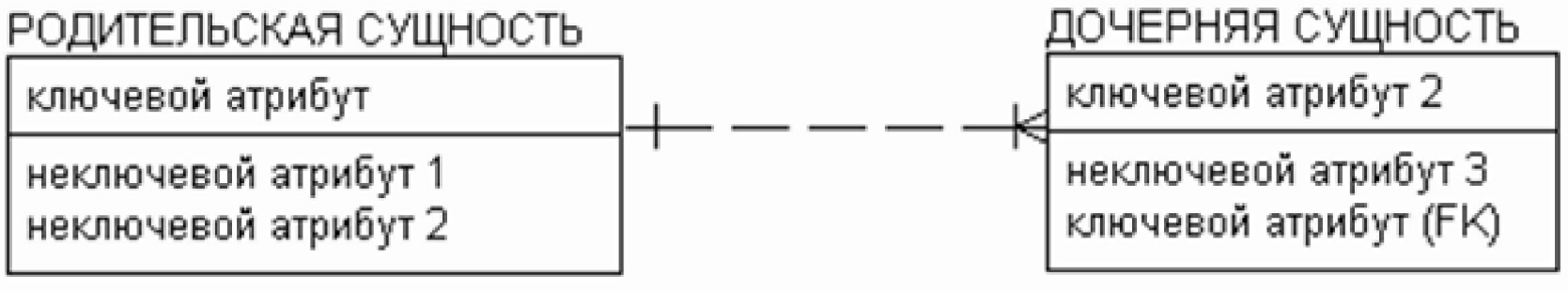
Примеры в нотации IE



Экземпляру родительской сущности могут соответствовать 0, 1 или много экземпляров дочерней сущности



Экземпляру родительской сущности могут соответствовать ни одного или один экземпляр дочерней сущности.



Экземпляру родительской сущности могут соответствовать 1 или много экземпляров дочерней сущности.

О жизненном цикле баз данных

Выделим этапы разработки, называемые «Анализ», «Проектирование» и «Реализация». Обратим внимание на то, что только в некоторых технологиях разработки эти этапы выполняются последовательно и один раз за весь процесс разработки.

Будем помнить, что модель базы данных отображает часть информации из модели бизнеса, а пользователь “видит” данные в базе через “окно” интерфейса, пользователя, который в настоящее время пишется преимущественно в объектной парадигме программирования.

Анализ

- Определяются **цели** создания информационной системы. Выбирается стратегия разработки. Исследуются риски. Определяются особенности управления проектом.
- Подробно исследуют **бизнес-процессы**, создавая их описание, например, в BPwin (стандарты IDEF0, IDEF3). Выявляют информацию, необходимую для выполнения этих процессов. Выделяют сущности, их атрибуты и связи между ними. Создают информационную модель. На следующем этапе проектирования будет разработана модель данных.
- Особое внимание следует уделить **полноте информации, анализу возможных противоречий, поиску неиспользуемых и дублирующихся данных.** Помните, что заказчику легче формулировать спецификации отдельных компонентов системы, а не всей системы.
- Желательно описание бизнеса и информационной модели оформлять в виде документа – спецификации.

Два взаимосвязанных аспекта спецификации:

- функциональный - описание процессов;
- информационный - описание данных, необходимых для управления этими процессами.

Проектирование

На этапе проектирования используя результаты анализа уточняют семантику данных и разрабатывают:

1. схему базы данных (описания таблиц, представлений, столбцов, ограничений целостности, индексов, последовательностей, кластеров, процедур, функций курсоров и т.д.) С этой целью можно создавать в ERwin логические диаграммы “сущность-связь” (стандарт IDEF1x);
2. набор спецификаций модулей приложения.

Схема базы и спецификации модулей приложения должны быть согласованы с результатами этапа анализа и между собой.

Важнейшая задача этапа проектирования информационной системы – обеспечение производительности.

Рекомендуется результаты проектирования оформлять в виде единого документа, который называется **технической спецификацией (ТС).**

Реализация (разработка)

Разработка это написание кодов приложения, в том числе:

1. скриптов создающих базу и, может быть, частично заполняющих ее (серверная часть приложения, размещаемая на сервере базы данных);
2. текстов хранимых процедур, функций, триггеров, курсоров (серверная часть приложения);
3. текстов интерфейсов пользователя (клиентская часть приложения);
4. коммуникационной части системы, в т.ч. серверы приложений.

Важнейшие компоненты реализации:

- Продуманное всестороннее тестирование;
- Периодическое возвращение к этапам анализа и проектирования.

Всегда помните и правильно интерпретируйте совет классика:

“Кодированию программы следует сопротивляться до последней возможности.”

Цена ошибок, допущенных на начальных стадиях слишком велика. Чем аккуратнее вы выполнили анализ и проектирование, тем меньше вам придётся исправлять код.

Две модели жизненного цикла информационной системы

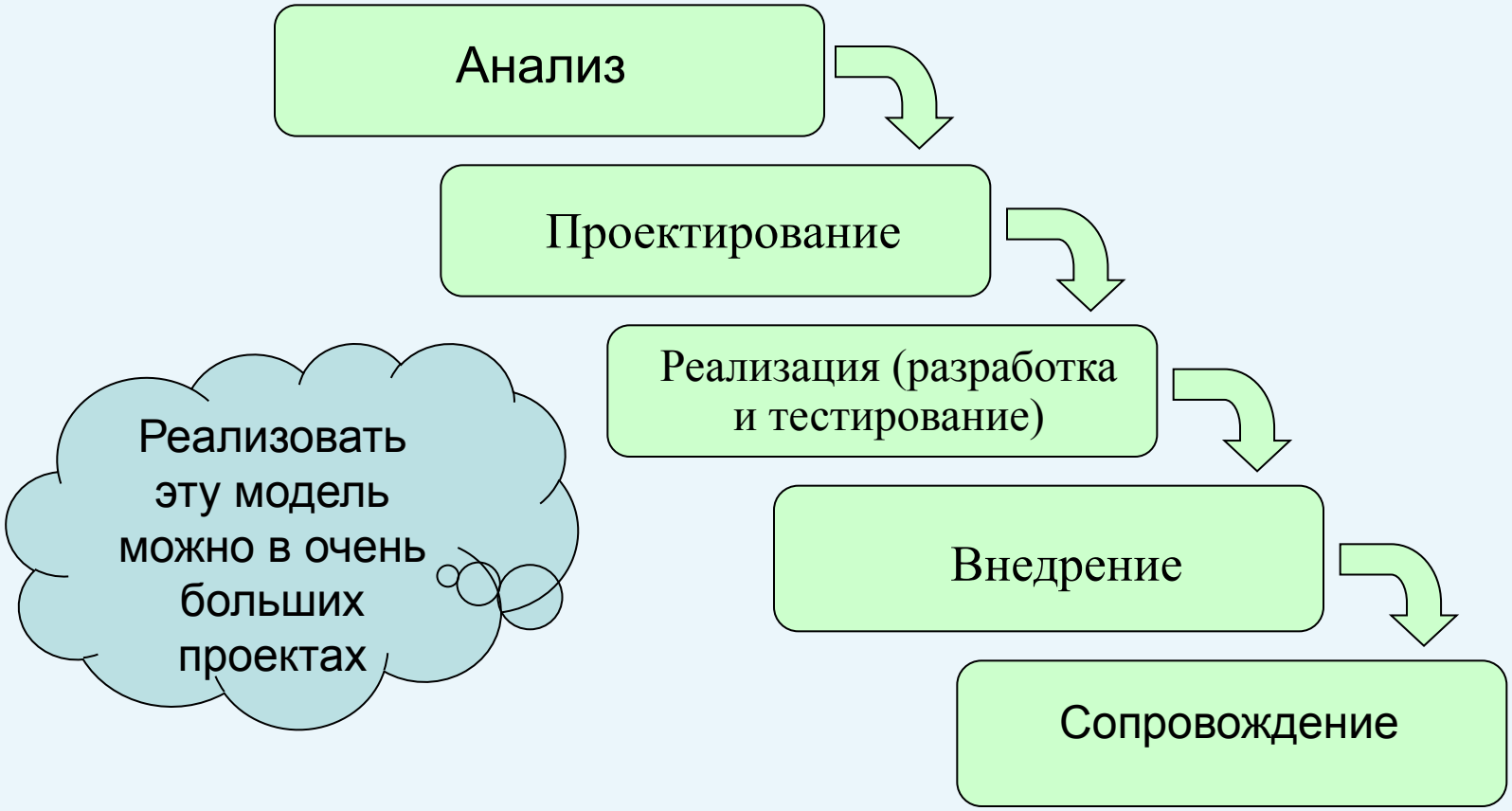
Последовательная (waterfall)

- Каждый этап выполняется для всей системы. Система разрабатывается и внедряется вся сразу, а не отдельными модулями.
- Следующий этап начинается после завершения предыдущего

Инкрементная

- Система разбивается на модули, разрабатываемые раздельно и не одновременно
- Функциональность системы наращивается постепенно
- Итерации повторяются многократно и могут перекрываться по времени

Последовательная модель ЖЦ



Замечание: Этап внедрения требует в основном организационных усилий. Сопровождение это постоянные доработки информационной системы; составляет порядка 2/3 общей стоимости владения.

Недостатки последовательной модели

- Внедрение системы и поиск основной массы ошибок откладываются до окончания разработки.
- Пользователи не работают с системой до момента внедрения и не имеют времени для оценки и изменения постановки задачи.
- Руководство заказчика оценивает работу только по бумажным документам. Это может порождать недоверие.
- Почти невозможно написать хорошую спецификацию не имея возможности ее последовательного уточнения. Для этого желательно работать с интерфейсами пользователя, тестировать отдельные алгоритмы, просматривать отчеты.

Область применения: большие проекты, выполняемые сотнями и тысячами исполнителей, например, в оборонной промышленности

Инкрементная модель ЖЦ

В один момент времени могут прорабатываться несколько этапов, обычно для разных подсистем. Возможен возврат на всю глубину последовательности этапов.

Замечание: Не следует считать, что рассмотренные модели ЖЦ рекомендуются для работы с небольшими и средними проектами. Рекомендуется ознакомиться с экстремальным программированием и др. современными технологиями. (См. курс ”Технологии программирования”).

Заключение: семантические модели

Рассмотрены два достаточно сложных вопроса:

- модели данных называемые семантическими (бегло);
- диаграммы сущность – связь (подробнее, но не исчерпывающе);

Пришлось вспомнить о семантике данных.

Семантические модели необходимы потому, что ни одна из поддерживаемых СУБД моделей данных не обеспечивает полного представления семантики данных предметной области. Так, в реляционной модели невозможно описание декларативных ограничений целостности информационной системы кроме метаданных, первичных, уникальных, внешних ключей и ключей-кандидатов. Модель сущность-связь описывает семантику лучше, но не формально и не исчерпывающе.

Стал понятнее термин “сущность” ранее использованный без уточнений.

Важно уяснить, что ER-диаграммы это частный случай семантических моделей данных. Всех проблем он не решает. В расширенной EER модели (и в ERWin) используется примитивный вариант наследования (категория).

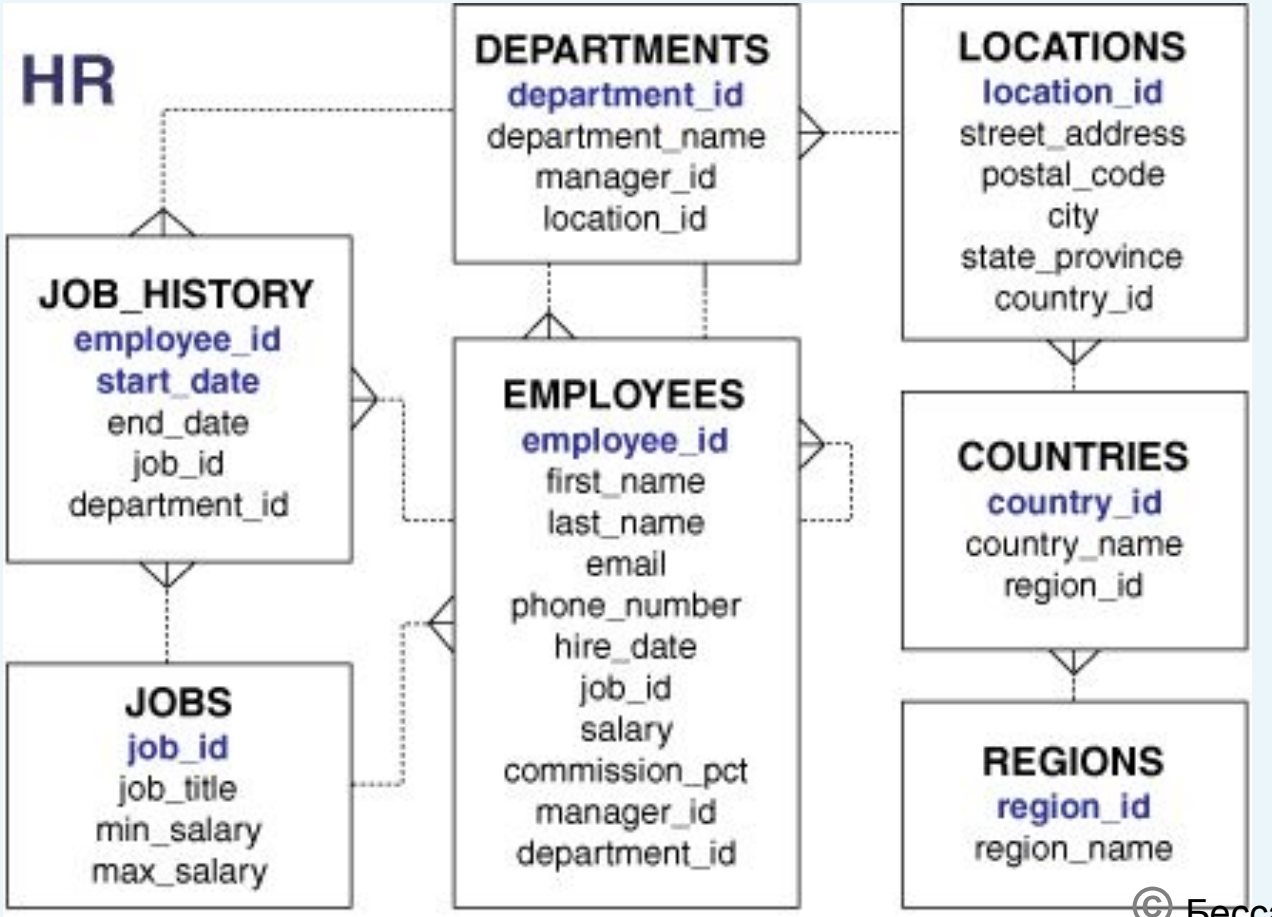
Итак, в нашем распоряжении модель данных сущность – связь.

Задание для самостоятельной работы

Разберите схему HR (Human Resources). Опишите ее семантику. Укажите недостатки.

Сосредоточьтесь на таблицах EMPLOYEES и JOB_HISTORY. Обратите внимание на то, что, комиссионные положено платить только “продажникам”. Какие проблемы с комиссионными могут возникнуть?

Попробуйте для выделенных двух таблиц (EMPLOYEES и JOB_HISTORY) написать обобщение схемы, пригодное для широкого круга задач.

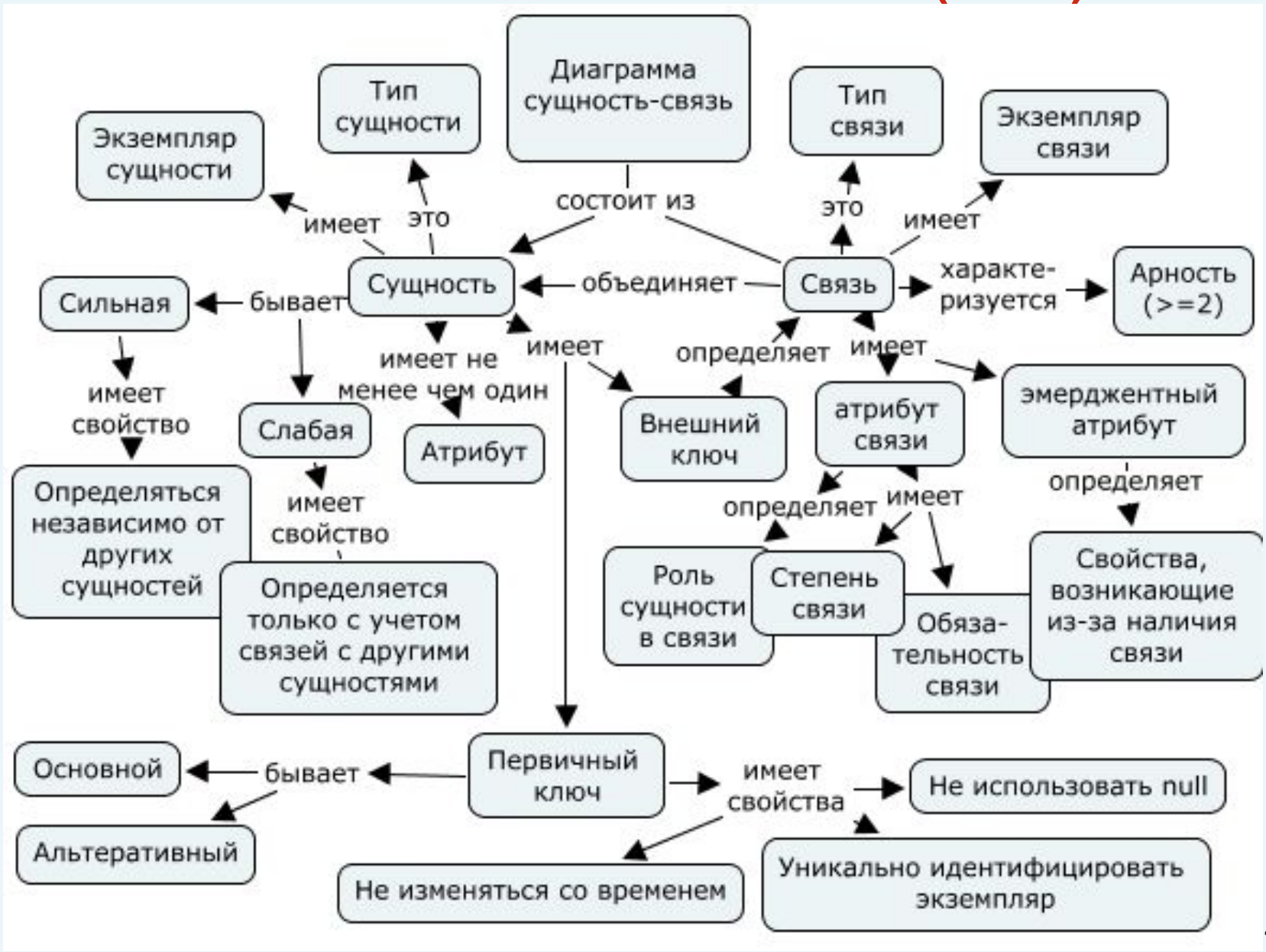


Литература

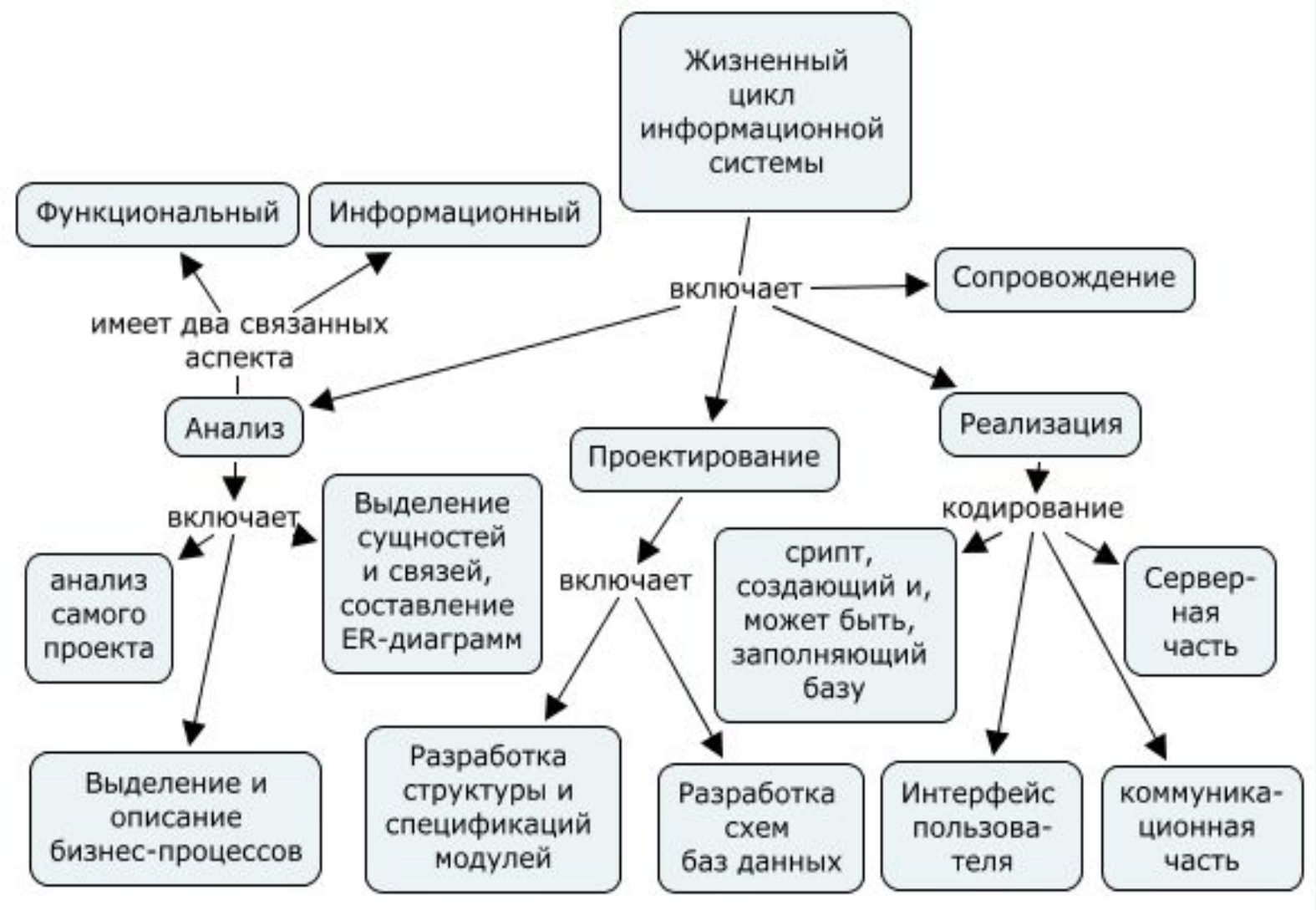
1. Петер Пин-Шен Чен. Модель “сущность-связь” – шаг к единому представлению о данных. СУБД (перевод из ACM Transactions on Database System, v. 1, #1, 1976)

Совет начинающим: Любите издания Association of Computer Machinery (ACM) и IEEE – источник знания по базам данных (и не только по ним). Адреса:
www.acm.org/dl (платный, в настоящее время в университете не доступен)
Ieeexplore.ieee.org/xpl/periodicals.jsp (платный)

Основные понятия (1/2)



Основные понятия (2/2)



Словарь студента (1/4)

- **Анализ** – этап жизненного цикла информационной системы, на котором анализируют предметную область и составляют её описания (модели). В анализе могут быть выделены два связанных аспекта – функциональный информационный (структурный).
- **Атрибут** - это свойство сущности или связи значение атрибута может иметь простой тип или быть списком значений.
- **Атрибут привязки** – указывает какие сущности участвуют в связи.
- **Атрибут мигрирующий** – передаётся от родительской сущности к дочерней при установлении связи.
- **Атрибут эмерджентный** – описывает свойство, появляющееся при создании связи.
- **Ключ альтернативный** – потенциальный ключ, не выбранный в качестве первичного.
- **Ключ внешний** – используется для задания связи с другой сущностью через сравнение с её первичным ключом. Ключ внешний -- это совокупность атрибутов отношения, значения которых являются одновременно значениями первичного или возможного ключа другого отношения.

Словарь студента (2/4)

- **Ключ первичный** – набор атрибутов, который уникальным образом идентифицирует экземпляр, не использует NULL значений и не изменяется со временем. Принято называть первичным ключом минимальный первичный ключ.
- **Ключ уникальный** – отличается от первичного возможностью использования значения null.
- **Ключ неуникальный** – задает несколько сущностей.
- **Модель “сущность – связь”** – семантическая модель данных, в которой выделяют сущности и связи между ними, обладающие атрибутами; Связи могут иметь арность больше двух, определять зависимость по существованию (быть идентифицирующими и неидентифицирующими, обязательными и необязательными), иметь тип n:m.
- **Модель семантическая** – “так называют модели данных, обладающие более развитыми средствами отображения семантики предметной области по сравнению с … сетевой, иерархической и реляционной моделями данных” (М.Р. Когаловский).
- **Обязательность связи** – в обязательной связи любой экземпляр связываемой сущности должен участвовать хотя бы в одном экземпляре связи.
- **Проектирование** – этап жизненного цикла информационных систем, на котором разрабатывают схему базы данных и спецификации программных модулей.

Словарь студента (3/4)

- **Реализация** -- этап жизненного цикла информационных систем, на котором кодируются модули реализующие функции информационной системы, пишутся скрипты, создающие и, может быть, заполняющие базу, выполняется тестирование.
- **Роль** -- имя конца связи определяющее функцию связи по отношению к связываемой сущности.
- **Связь** – определяет отношение между двумя или более сущностями. Возможно определение связи сущности с собой (пример: связь “непосредственный начальник -- подчиненный” заданная на двух экземплярах отношения “сотрудники”). Связи именуются глаголами.
- **Степень конца связи** – показывает, сколько экземпляров данного типа сущности должно связываться одним экземпляром данного типа связи.
- **Сущность** – некоторый объект, документ, событие, процесс, определяемый в предметной области. Сущности именуются существительными.

Словарь студента (4/4)

- **Сущность ассоциативная** – искусственно созданная сущность, позволяющая свести связь типа m:n к двум связям типа 1:n. Желательно определить её имя содержательно.
- **Сущность сильная** – определяется в изоляции от других сущностей, только через смысл своих атрибутов и своего имени.
- **Сущность слабая** – может быть определена только совместно с какой-нибудь сильной сущностью и потому необходимо внести в неё по крайней мере один атрибут из связанной сильной сущности.