Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №**8

по дисциплине «Базы данных»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

Кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Санкт-Петербург, 2025

Оглавление

[Яндекс Практикум 9](#_Toc193839841)

[Инфологическое проектирование баз данных. ER-диаграммы 9](#_Toc193839842)

[Er-диаграммы в нотации Чена 12](#_Toc193839843)

[ER-диаграммы в нотациях Баркера и Мартина. CASE-средства 13](#_Toc193839844)

[Проектирование баз данных с использованием методологии IDEF1X 15](#_Toc193839845)

**ЛАБОРАТОРНАЯ РАБОТА № 1 СЕДЬМОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ» РАБОТА С ДИАГРАММАМИ В НОТАЦИИ ЧЕНА**

Требуется, основываясь на описании предметной области, спроектировать базу данных, все отношения в которой нормализованы до НФБК.

Предметная область – учет накопителей на жестких дисках (HDD), используемых в организации. О дисках мы знаем следующее:

* у жесткого диска есть производитель;
* у производителя есть web-сайт (считаем, что только один), где можно почерпнуть много важной для нас информации;
* для жесткого диска всегда определена его модель;
* модель определяет объем диска (в гигабайтах), скорость вращения шпинделя, тип используемого интерфейса;
* по названию модели можно определить производителя;
* у конкретного экземпляра жесткого диска есть серийный номер, мы знаем его модель, дату приобретения, дату выхода из строя (если диск вышел из строя; возможность ремонта и восстановления не рассматриваем); может возникнуть необходимость внести текстовые комментарии по поводу его работы.

При проектировании надо учитывать, что:

* для любой модели обязательно должен быть указан производитель и объем;
* для любого диска должна быть указана модель;
* фирмы-производители и модели дисков именуются уникальным образом; серийные номера дисков также уникальны.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

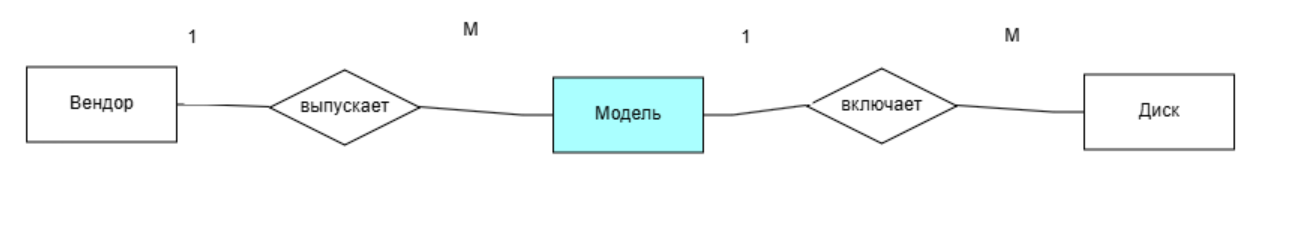
A screenshot of a computer program

AI-generated content may be incorrect.

**Задание 1**

Теперь для той же предметной области надо выделить сущности и связи между ними.  На бумаге или в любом редакторе построите диаграмму в нотации Чена, указав на ней только сущности, связи и мощность связи (атрибуты  наносить на диаграмму не надо).

**Решение**

****

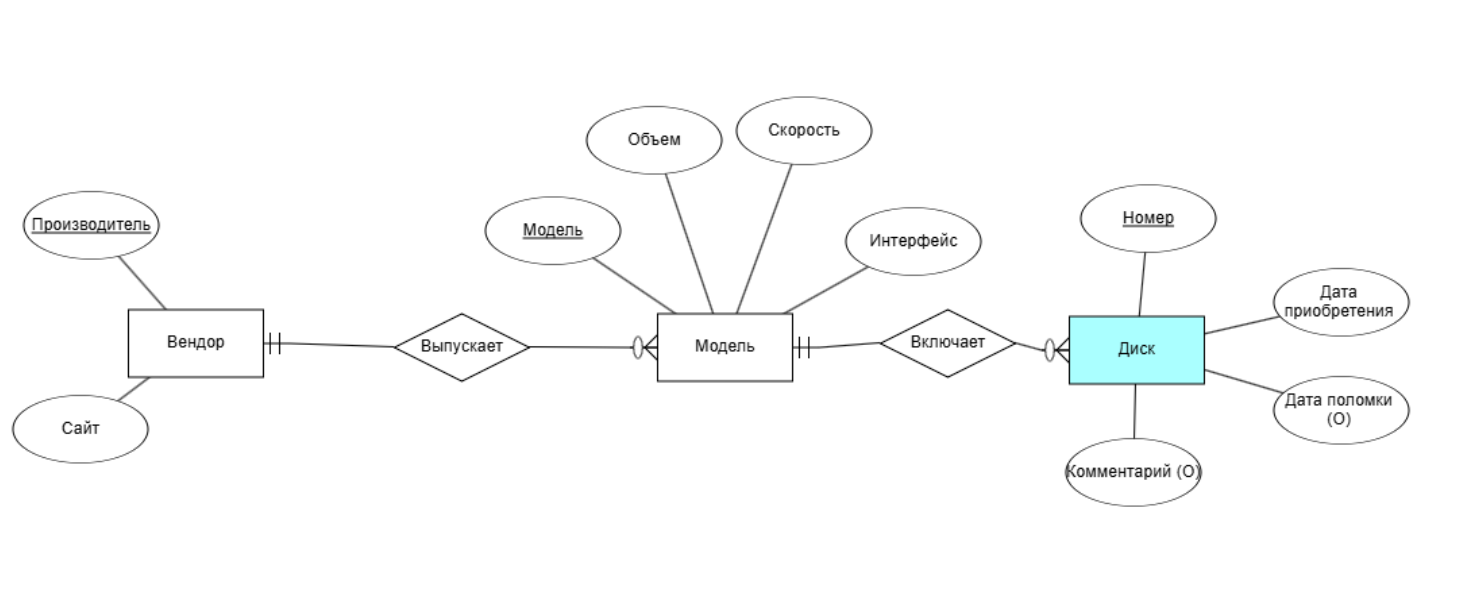
**Разбор связей:**

* Вендор выпускает Модель (1:M)
  + Один вендор может выпускать множество моделей.
  + Каждая модель создается только одним вендором.
* Модель включает Диск (1:M)
  + Одна модель может включать в себя несколько дисков.
  + Каждый диск принадлежит только одной модели.

**Задание 2**

Теперь для той же задачи постройте ER-диаграмму в среде ERD PLUS, указав на диаграмме все атрибуты.

**Решение**



**Объяснение элементов схемы:**

**Сущности:**

1. **Вендор (Производитель)**
   * Атрибуты:
     + Производитель — название производителя (уникальное).
     + Сайт — веб-сайт производителя.
2. **Модель (Model)**
   * Атрибуты:
     + Модель — название модели (уникальное).
     + Объем — ёмкость диска в гигабайтах.
     + Скорость — скорость вращения шпинделя.
     + Интерфейс — тип интерфейса (например, SATA, NVMe).
3. **Диск (HDD)**
   * Атрибуты:
     + Номер — серийный номер жёсткого диска (уникальный).
     + Дата приобретения — дата покупки диска.
     + Дата поломки (O) — дата выхода из строя (опциональный атрибут, так как диск может быть рабочим).
     + Комментарий (O) — текстовые заметки по поводу состояния диска (опционально).

**Связи между сущностями:**

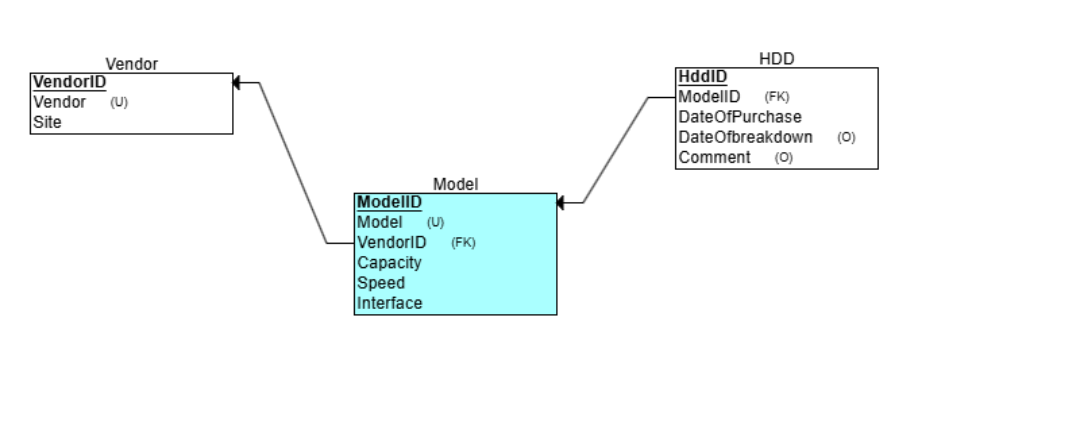
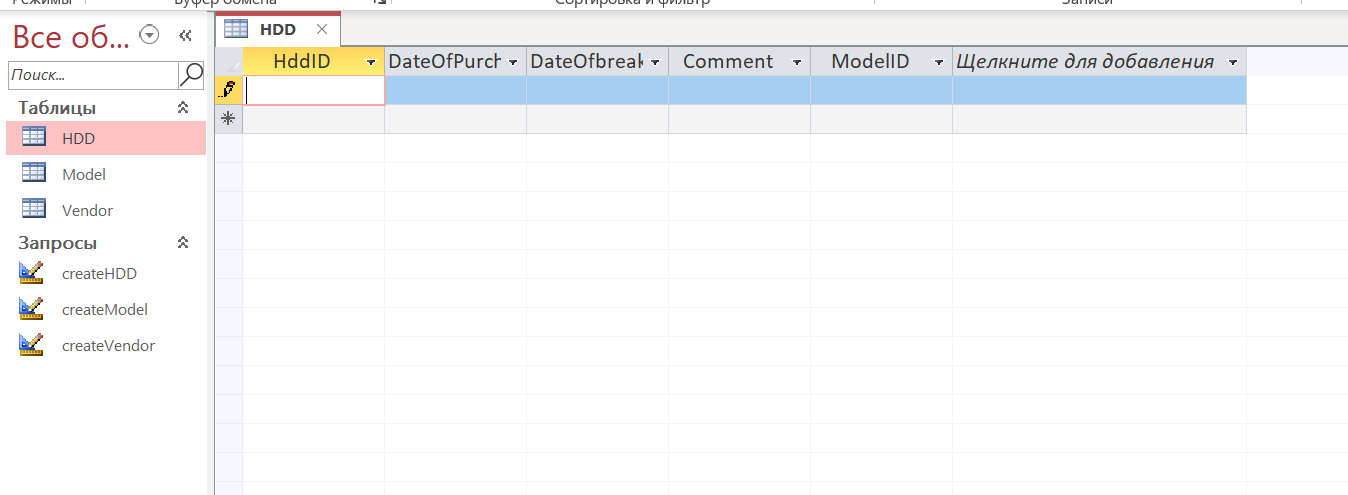
**Производитель (Вендор) → Модель (Model)**

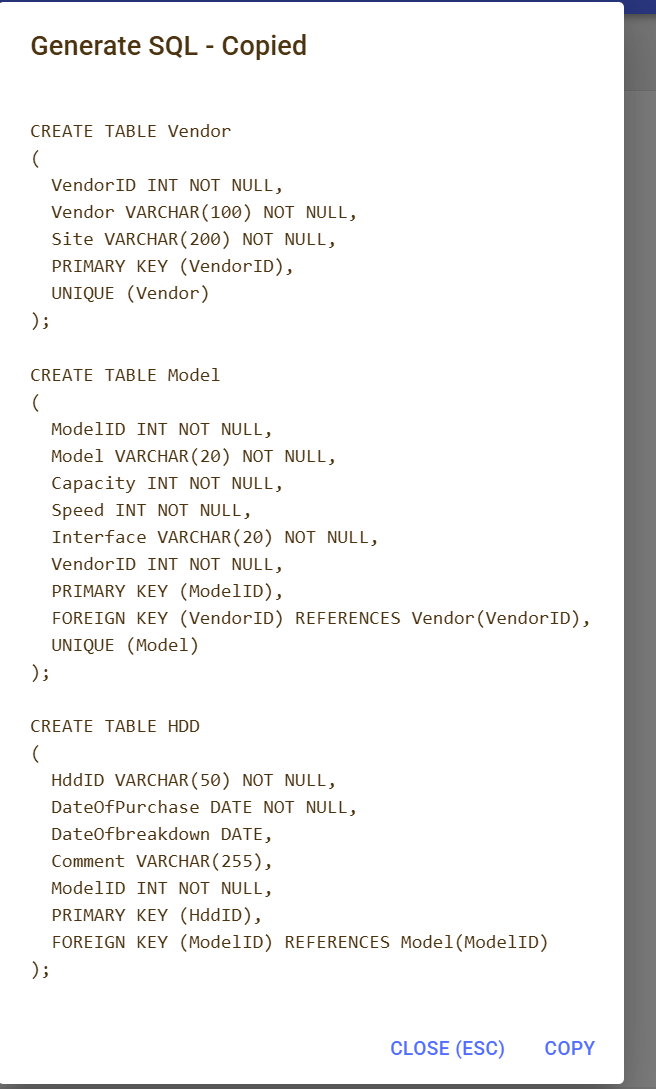
* **Производитель (Вендор)**
  + Может быть **без моделей** (**"Optional"**).
  + Может выпускать **много моделей** (**"Many"**).
* **Модель (Model)**
  + **Обязательно должна иметь производителя** (**"Mandatory"**).
  + Может принадлежать **только одному производителю** (**"One"**).

**Модель (Model) → Диск (HDD)**

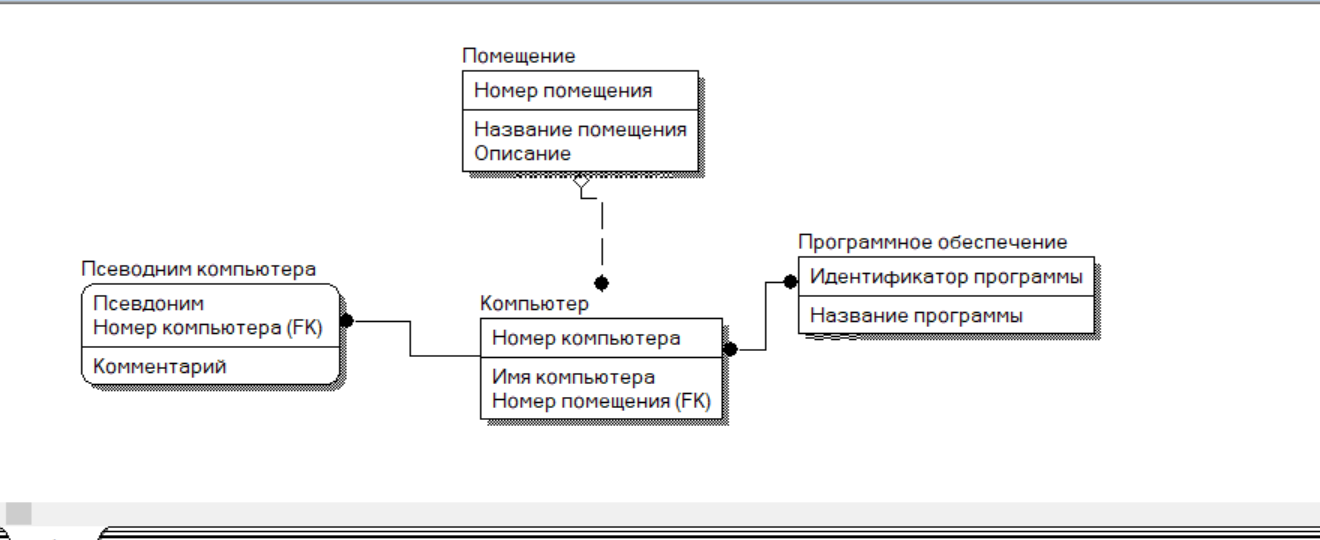
* **Модель (Model)**
  + **Optional** → это значит, что **модель может существовать без дисков**.
  + **Many** → это значит, что **одна и та же модель может включать множество дисков**.
* **Диск (HDD)**
  + **Mandatory** → это значит, что **у каждого диска обязательно должна быть модель**.
  + **One** → это значит, что **каждый конкретный диск относится только к одной модели**.

Дополнительное задание (выполняется по желанию).



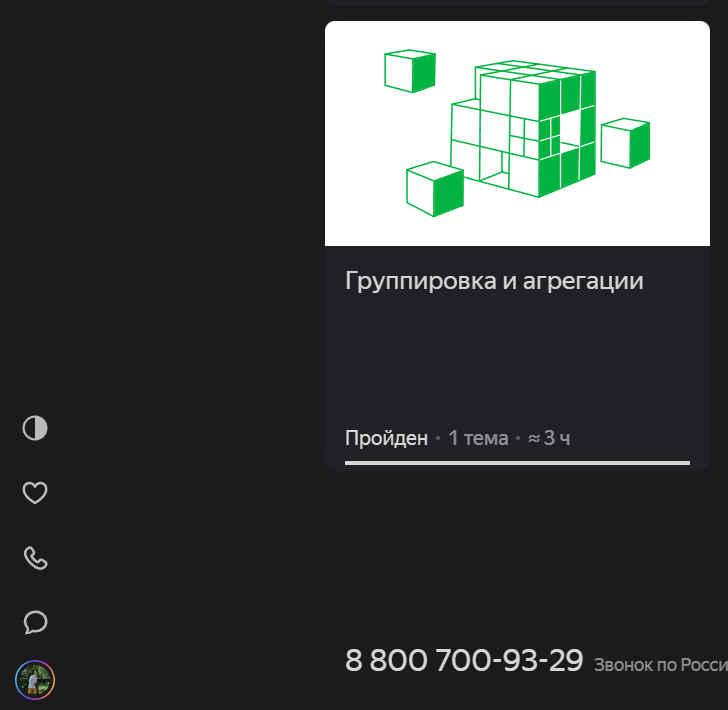


**ЛАБОРАТОРНАЯ РАБОТА № 2 СЕДЬМОЙ НЕДЕЛИ КУРСА «УПРАВЛЕНИЕ ДАННЫМИ»**



Яндекс Практикум

A screenshot of a computer

AI-generated content may be incorrect.

Инфологическое проектирование баз данных. ER-диаграммы

Описание предметной области в инфологической модели обычно производится в терминах «объект» - «отношение» Описание модели «сущность - связь» было впервые опубликовано П. Ченом в 1976 г.

Сущность (объект) - множество экземпляров реальных или абстрактных однотипных объектов, присутствующих в предметной области. Именовать сущность принято существительным в единственном числе, например «Рейс» или «Заказчик». Сущности бывают правильными (другое название - «сильными») и слабыми. Правильная сущность соответствует тем объектам, чье существование не зависит от других объектов. Слабая сущность находится в зависимости от других сущностей (не может существовать при отсутствии некоторой другой сущности). Например, если заказ всегда должен быть сделан каким-то клиентом, то «Заказ» будет слабой сущностью, зависящей от сущности «Клиент». В этом примере, если дополнительных условий не задано, «Клиент» - правильная сущность. Сущность имеет набор свойств, называемых атрибутами. Каж- дая группа атрибутов, описывающая одно реальное проявление сущности, представляет собой экземпляр. Для того чтобы одну сущность можно было отличить от другой, выбирается один или несколько атрибутов с уникальным значением (комбинацией), который называется идентификатором сущности. Подобные атрибуты называют ключевыми.

Атрибуты могут быть:

1) простыми и составными;

2) ключевыми и неключевыми;

3) обязательными и необязательными;

4) однозначными и многозначными.

Относительно разделения атрибутов на простые и составные можно сделать следующее замечание. Составной атрибут, такой как адрес, может включать в себя набор значений: индекс, страна, город, улица, дом. При построении инфологической модели такие атрибуты допустимы. При переходе к даталогической модели все зависит от выбранной модели данных и возможностей СУБД. Например, в реляционной модели составных атрибутов нет, их надо будет представлять как множество простых.

Значение обязательных атрибутов всегда должно быть задано. Необязательные атрибуты могут отсутствовать. Атрибуты также могут быть однозначными и многозначными: в компании может быть только один главный бухгалтер и несколько номеров телефонов. При построении инфологической модели предметной области допустимо указывать и многозначные атрибуты, но при переходе на более низкий уровень проектирования, когда надо учитывать особенности выбранной модели данных и СУБД, от подобных атрибутов, возможно, придется избавляться. В частности, реляционная модель не поддерживает многозначные атрибуты. Чтобы решить эту проблему, обычно вводят дополнительное реляционное отношение. В нашем примере с множеством телефонов это может быть «Телефон фирмы».

Атрибуты должны быть заданы на доменах. На ER-диаграммах, которые будут рассматриваться ниже, обычно домены не указываются. Эта информация помещается в отдельный документ, называемый словарем данных. В то же время некоторые варианты нотации Чена допускают указания доменов на диаграммах под изображением атрибута.

Связь (отношение) при инфологическом проектировании описывает имеющуюся в предметной области логическую связь между двумя или более сущностями. Объекты, которые соединяются, называются участниками связи. Число участников определяет степень связи. Так же, как и в случае с сущностями, выделяют тип и экземпляр связи. Связь принято именовать с помощью глагола, например связь между сущностями «Клиент» и «Заказ» можно назвать «Размещает».

Пусть R - тип связи, Е - тип сущности. Если каждый экземпляр сущности типа Е находится, по крайней мере, в одном экземпляре связи типа R, то участие Е в R называется полным или обязательным, в противном случае - частичным или необязательным. Например, пусть рассматриваемая предметная область — это образование, в ней выделены сущности «Преподаватель» и «Курс», а между ними определена связь «Читает». Если каждый курс читается хотя бы одним преподавателем, но есть преподаватели, которые не читают ни одного курса, то участие сущности «Курс» в связи «Читает» является полным, а сущности «Преподаватель» - частичным. Выделяют следующие типы связей:

1) «один-к-одному»;

2) «один-ко-многим»;

3) «многие-ко-многим».

Приведем формальные определения для данных типов связей. Пусть имеются две сущности (А и В), участвующие в связи R. Если каждый экземпляр А, сущности А связан не более чем с одним экземпляром сущности В, а каждый экземпляр В, сущности В связан не более чем с одним экземпляром сущности А, то связь R имеет тип «один-к-одному». На практике данный тип связи встречается достаточно редко.

Связь R имеет тип «один-ко-многим», если экземпляр А, сущности А может быть связан с нулем, одним или несколькими экземпляра-ми сущности В, а экземпляр В, сущности В может быть связан не более чем с одним экземпляром сущности А. Обратным по отношению к типу «один-ко-многим» является тип связи «многие-к-одному». Связи типа «один-ко-многим» наиболее часто встречаются при разработке БД.

Связь между сущностями А и В будет иметь тип «многие-ко-многим», если экземпляр А, сущности А может быть связан с нулем, одним или несколькими экземплярами сущности В и экземпляр В, сущности В также может быть связан с нулем, одним или несколькими экземплярами сущности А. Например, если один курс могут читать несколько преподавателей, а один преподаватель может читать несколько курсов, то между сущностями «Преподаватель» и «Курс» есть связь типа «многие-ко-многим». В большинстве случаев в процессе проектирования связи типа «многие-ко-многим» удаляются путем введения составных (ассоциированных) сущностей и определения двух связей типа «один-ко-многим».

Составная (ассоциированная) сущность представляет данные, которые ассоциируются со связью между двумя или более сущностями. Введение данного типа сущностей связано с тем, что связь, по своей сути, не должна обладать собственными свойствами. Но часто случается, что это не так. Например, связь между пассажиром и авиарейсом, которым он летит, описывается набором собственных параметров: номер места, класс, цена, дата приобретения билета и т. д. Чтобы представить все эти данные, можно ввести составную сущность «Билет».

Существует несколько нотаций или правил изображения диаграмм «сущность - связь» (ER-диаграмм).

Er-диаграммы в нотации Чена

Сущности изображаются прямоугольником, внутри которого помещается имя сущности. Прямоугольник, соответствующий слабой сущности, обводится двойной рамкой. Атрибуты изображаются в виде овала, соединенного с соответствующим прямоугольником, но обычно на диаграммах атрибуты вообще не отображают. Ключевые атрибуты выделяются подчеркивание мили служебным символом в начале имени. Связь обозначается ромбом. Ромб окружен двойной линией, если связь задана между слабой сущностью и сущностью, от которой она зависит. Участники связи присоединены к ромбу линией. Для обозначения типа связи используются символы «1» и «М» (иногда вместо «М» применяют символ бесконечности или ставят одинарные и двойные стрелки). Двойная линия обозначает полное участие сущности в связи. Ассоциированные сущности изображаются ромбом, заключенным в прямоугольник,

A diagram of a diagram

AI-generated content may be incorrect.

На рис. 6.1 изображены две сущности «Сотрудник» и «Подчиненный» и задана связь «Подчиняется» типа «один-ко-многим» (одному сотруднику могут подчиняться несколько человек, каждый подчиненный имеет «над собой» только одного прямого начальника). Для сущности «Сотрудник» также обозначены ее атрибуты. Здесь атрибут «номер» является ключевым, атрибут «адрес» - составным. Диаграмма «сущность - связь» в некотором смысле является абстрактным макетом БД, поэтому был выработан ряд правил, облегчающих переход от диаграмм к реляционным отношениям:

1) каждый правильный (сильный) тип сущности соответствует базовому реляционному отношению;

2) каждая бинарная связь типа «многие-ко-многим» также соответствует отдельному отношению, которое должно включать в себя два внешних ключа, ссылающихся на потенциальные ключи отношений, соответствующих сущностям - участникам связи;

3) связь типа «один-ко-многим» между сильными сущностями может быть представлена с помощью внешнего ключа и не требует отдельного отношения;

4) связь слабого объекта с сильным, от которого он зависит, является связью типа «многие-к-одному» и может быть представлена внешним ключом. В некоторых случаях, когда и сильная, и подчиняющаяся ей слабая сущности представлены одним реляционным отношением, внешний ключ может ссылаться на первичный ключ своего же отношения. Так можно поступить в примере «Сотрудник» - «Подчиненный». Однако в подобных случаях самый главный руководитель станет «самоначальником» - один и тот же человек будет являться и начальником, и подчиненным;

5) атрибуты сущностей приводятся к атрибутам отношений;

6) в случае более чем бинарной связи (n-арной), обычно вводят n + 1 отношение: по одному на каждую сущность и одну на связь.

ER-диаграммы в нотациях Баркера и Мартина. CASE-средства

В начале 1980-х гг. были предложены новые подходы к инфологическому проектированию БД, в большей степени ориентированные на БД реляционного типа. Среди работавших в этом направлении исследователей можно назвать Р. Баркера (Richard Barker) и авторов нотации Information Engineering (сокр. IE) Дж. Мартина (JamesMartin) и К. Финкельштейна (Clive Finkelstein).

В предложенных нотациях сущности изображаются сходным образом в виде прямоугольника, содержащего в заголовке имя сущности, и далее идет перечень атрибутов. Ключевые атрибуты выделяются на диаграмме шрифтом, специальными символами или отделяются чертой от остальных. Все связи являются бинарными (т. е. только с двумя участниками) и изображаются линией, соединяющей сущности. На рис. 6.2 представлены правила изображения связей в нотациях Баркера и Мар-тина [10].Нужно отметить, что из-за особенностей изображения связей нотации Баркера и Мартина в литературе иногда называют «crow's footnotation» (дословно - «нотация вороньей лапки»)

A table of text with different symbols

AI-generated content may be incorrect.

На рис. 6.3 приведен фрагмент диаграммы в нотации Мартина, изображающей две сущности («Клиент» и «Заказ») и связь между ними. Первичные ключи на рисунке выделяются символом «#». Предполагается, что:

* клиент может разместить один, несколько или ни одного заказа;
* заказ может быть размещен одним и только одним клиентом.

В настоящее время также широкое распространение получила нотация, определенная стандартом IDEF1X (полное название на англ. - Integration Definition for Information Modeling), речь о которой пойдет в следующем параграфе.

Задача проектирования БД для современной информационной системы корпоративного уровня может быть достаточно трудоемкой и требовать совместной работы большой группы специалистов-аналитиков, разработчиков БД, разработчиков прикладного ПО, специалистов в предметной области, для которой разрабатывается БД. Для автоматизации этого процесса широко используютсяCASE-средства - программные средства, поддерживающие однуили несколько технологий проектирования БД (также есть средствапроектирования ПО и т. д.). В качестве примера можно назвать про-граммные продукты erwin Data Modeler (ныне принадлежит QuestSoftware), ER/Studio Data Architect (IDERA, Inc.), PowerDesigner (pa3-работчик - компания SAP SE). Отчасти подобная функциональ-ность реализована и в популярном офисном программном продуктеMicrosoft Visio.

Проектирование баз данных с использованием методологии IDEF1X

Стандарт, описывающий методологию проектирования IDEF1X, был разработан в США в 1993 г. Он уточняет предшествующую версию IDEF1 и является частью семейства стандартов IDEF, к которому также относятся стандарты на методологию функционального моделирования систем IDEFO, моделирования динамически изменяющихся систем IDEF2 и ряд других.

Методология IDEF1X разрабатывалась для создания информационных моделей, представляющих структуру данных, описывающих предприятие или другую предметную область. Подобные модели также называют концептуальными.

Поддерживающие IDEF1X программные средства, такие как erwin Data Modeler, позволяют построить независимую от СУБД логическую модель БД и по ней создать физическую модель для реализации в среде конкретной СУБД. При этом появляется возможность уже на ранних этапах проектирования системы (при построении логической модели) согласовать описание предметной области: выделить объекты, их взаимосвязи, атрибуты, описывающие отдельные характеристики. При необходимости для одной согласованной логической модели можно создать физические модели для разных СУБД. Например, erwin Data Modeler 12.5 поддерживает создание физических моделей для СУБД IBM DB2, Microsoft SQL Server, MySQL, Oracle Database, SAP Adaptive Server Enterprise и ODBC-совместимых (от англ. Open Database Connectivity) СУБД.

На основе физической модели может быть сгенерирован скрипт, выполнение которого в среде выбранной СУБД приведет к созданию необходимых структур БД. Таким образом, при создании физической модели необходимо описать все подробности относительно БД, например конкретный тип данных, поддерживаемый выбранной СУБД, тогда как при создании логической модели можно ограничиться указанием обобщенного типа (числовой, строковый и т. п.).

Разделение на логическую и физическую модель БД имеет ряд преимуществ и с точки зрения документирования разрабатываемого решения. В силу ряда причин, таких как накладываемые СУБД ограничения на длину имен таблиц и столбцов, ограниченная поддержка национальных алфавитов в именах объектов БД, многие разработчики предпочитают использовать для таблиц и столбцов короткие имена, отображаемые латиницей. Например, столбец с идентификатором рабочей станции может называться «WstId». Такие CASE-средства, как erwin Data Modeler, позволяют при создании логической модели БД использовать понятные читаемые названия, а в физической модели БД - заменить их на короткие технические обозначения. С точки зрения документирования решений полезной является и возможность добавления комментариев и заметок в процессе работы над моделью.

При создании модели БД описывают сущности, их атрибуты и связи между сущностями. Терминология IDEF1X очень близка к терминологии, используемой в других методиках моделирования БД. Сущность (англ. Entity) является представлением множества реальных или абстрактных объектов (явлений, событий и т. п.), относимых к одному типу: они обладают одинаковым набором характеристик и могут участвовать в однотипных связях с другими сущностями. Конкретный объект из подобного множества будет называться экземпляром. Экземпляры должны отличаться один от другого. Сущности именуются существительным в единственном числе, например «Компьютер». Также стандарт IDEF1X допускает в имени указание уникального в рамках модели БД номера сущности, например «Компьютер/1». Обратите внимание, что в приведенном примере «1» — это номер сущности, а не ее экземпляра. Атрибут (англ. attribute) соответствует свойству или характеристике, имеющейся у всех или некоторых экземпляров сущности. Множество возможных значений атрибута ограничивается доменом, на котором он определяется.

Связь (англ. relationship) описывает логическое отношение между двумя сущностями или между экземплярами одной сущности. При моделировании БД связь принято именовать глаголом или глагольной фразой.

Рассмотрим пример, иллюстрирующий использование различных типов сущностей и связей, определяемых нотацией IDEF1X.Пусть требуется разработать БД для хранения информации о компьютерной системе предприятия. С помощью программы erwin DataModeler начнем создавать логическую модель. Компьютеры имеют уникальные инвентарные номера и имена, причем для одного компьютера может быть использовано несколько дополнительных имен (псевдонимов). Например, компьютер serv1.corp.ru может иметь псевдонимы www.mycorp.net и ftp.mycorp.net. Компьютеры размещены в помещениях организации, на них установлено ПО. Фрагмент логической модели БД представлен на рис. 6.4.

A diagram of a company

AI-generated content may be incorrect.

Сущности в IDEF1X изображаются с помощью прямоугольника, над которым указывается его название, а в верхней части над чертой перечисляются атрибуты, входящие в первичный ключ. Сущности могут быть независимыми и зависимыми. Отличие заключается в том, что для существования экземпляра зависимой сущности необходима его связь с экземпляром сущности, от которой он зависит. Зависимые сущности изображаются прямоугольником со скругленными краями. В примере, приведенном на рис. 6.4, имеются независимая сущность «Компьютер» и зависящая от нее сущность «Псевдоним компьютера». Это означает, что псевдоним не может существовать без указания компьютера, для которого он задан.

Между независимой сущностью и подчиненной ей зависимой устанавливается идентифицирующая связь типа «один-ко-многим». Связь изображается непрерывной линией с черным кружком на стороне подчиненной сущности. При создании модели БД в erwin Data Modeler сущности сначала создаются как независимые, а после установки идентифицирующей связи дочерняя сущность автоматически преобразуется в зависимую. При этом в ее первичный ключ автоматически добавляются атрибуты первичного ключа родительской сущности, которые также сформируют внешний ключ: на рис. 6.4 это атрибут «Номер компьютера», который входит в состав первичного ключа сущности «Псевдоним компьютера». Внешний ключ отмечается на диаграмме символами «(FK)».

Операция автоматического добавления (при создании связи) ключевых атрибутов родительской сущности в дочернюю сущность в качестве внешнего ключа называется миграцией ключей (англ. кеуmigration).

Рассмотрим вторую связь типа «один-ко-многим» представленную на диаграмме (рис. 6.4). Это связь между двумя независимыми сущностями «Помещение» и «Компьютер». В терминологииIDEF1X такая связь будет являться не идентифицирующей (англ.Nonidentifying Relationship). Она изображается на диаграмме пунктирной линией с черным кружком со стороны дочерней сущности. В данном случае мы считаем, что для компьютера может быть определено не более одного помещения и могут существовать компьютеры, для которых помещение не задано. С точки зрения проектирования реляционной БД это означает, что в таблице «Компьютер» для внешнего ключа «Номер помещения» допустимо значение NULL. На диаграмме это изображается пустым ромбом со стороны родительской сущности (рис. 6.4). Такая связь называется опциональной, или необязательной (англ. optional nonidentifying relationship) [11]. Если ромба нет, это означает обязательное участие подчиненной сущности в данной связи, что реализуется ограничением NOT NULL для соответствующего внешнего ключа (подробнее об ограничениях, определяемых при создании таблиц реляционной БД, см. тему7). В IDEF1X такая связь называется обязательной не идентифицирующей связью (англ. mandatory nonidentifying relationship).

При добавлении в модель не идентифицирующей связи также осуществляется миграция ключей: в дочернюю сущность автоматически добавляется внешний ключ, формируемый на основе первичного ключа родительской сущности. Однако в отличие от идентифицирующей связи, в данном случае внешний ключ не становится частью первичного ключа дочерней сущности.

Третий тип связи, определяемый методологией IDEF1X, связь «многие-ко-многим». При проектировании реляционной БД подобная связь допустима только в логической модели. В приведенном на рис. 6.4 примере это связь между сущностями «Компьютер» и «Программное обеспечение». Связь «многие-ко-многим» изображается сплошной линией с двумя черными кружками на концах. В соответствии с IDEF1X такую связь рекомендуется именовать двумя глагольными фразами, разделенными косой чертой, что облегчает чтение диаграммы. Пусть связь «многие-ко-многим» установлена между сущностями «Сотрудник» и «Проект». Ее можно назвать «участвует/задействован». При чтении диаграммы получим «Сотрудник участвует в Проекте» и «в Проекте задействован Сотрудник». В примере, представленном на рис. 6.4, связь «многие-ко-многим» можно назвать «присутствует/установлено».

Реляционная модель в явном виде не поддерживает связи типа «многие-ко-многим». При проектировании БД в среде erwin, при переходе от логической модели БД к физической, связь «многие-ко-многим» автоматически преобразуется: создается зависимая сущность, для которой определяются две связи «один-ко-многим». На рис. 6.5 изображен фрагмент физической модели БД, построенной для логической модели, представленной на рис. 6.4. Еще раз хочется обратить внимание, что erwin Data Modeler позволяет давать различные имена соответствующим объектам физической и логической модели. Поэтому сущностям «Компьютер» и «Программное обеспечение» в физической модели на рис. 6.5 соответствуют таблицы «Computer» и «Software»; переименованы были и столбцы таблиц. Автоматически созданная таблица «Computer\_Software», имя которой было сгенерировано на основе имен родительских таблиц, получила составной первичный ключ, включающий первичные ключи исходных таблиц. Таким образом, миграция ключей в данном случае произошла на уровне физической модели.

A diagram of a computer software

AI-generated content may be incorrect.

Прежде чем продолжить описание типов связей, определенных нотацией IDEF1X, рассмотрим уровни представления модели БД, предлагаемые erwin Data Modeler. Надо отметить, что данное разделение уровней не определяется в IDEF1X, но в целом характерно для CASE средств проектирования БД, поддерживающих эту методологию. В соответствии с идеологией нисходящего проектирования разработчики erwin Data Modeler ввели несколько промежуточных уровней моделирования [14]. Для логической модели определяются следующие формы представления. Диаграмма «сущность связь» включает основные сущности, выявленные в предметной области, и связи между ними. Атрибуты не указываются. Диаграммы не перегружены деталями и могут использоваться для обсуждения структуры проектируемой базы в целом. Пример подобной диаграммы представлен на рис. 6.6.

A diagram of a diagram

AI-generated content may be incorrect.

Модель базы данных, основанная на ключах (англ. keybased), является более подробной, включает все сущности, их ключи, а также связи. В интерфейсе erwin Data Modeler при работе с моделью можно переключиться на представление, использующее только первичные ключи (англ. primary key display level, рис. 6.7, а) или первичные и внешние ключи (англ. keys display level, рис. 6.7, б). Нижний уровень для логической модели полная атрибутивная модель (англ. fullyattributed, FA), включающая все сущности, их атрибуты и связи. Структура проектируемой БД должна быть приведена в соответствие третьей нормальной форме. В качестве примера может выступать модель, изображенная на рис. 6.4, при условии, что все необходимые объекты предметной области, их связи и характеристики представлены на диаграмме.

Для физической модели определены два уровня:

1. трансформационная модель (англ. Transformation Model,TМ) описывает преобразование логической реляционной модели в структуры, поддерживаемые СУБД, выбранной для реализации.
2. Эти структуры оптимизированы, исходя из возможностей СУБД, объемов данных, ожидаемых запросов к данным, и могут быть деморализованы (не соответствовать требованиям третьей нормальной формы);2) модель СУБД (англ. DBMS Model) создается на основе трансформационной модели и содержит отображение структур системного каталога сервера БД, описывающих разрабатываемую БД.

A diagram of a company

AI-generated content may be incorrect.

Далее по ходу изложения в иллюстрациях будут приводиться фрагменты моделей разного уровня, в зависимости от того, что наиболее важно показать на конкретном рисунке.

Вернемся теперь к рассмотрению особенностей связей между сущностями, учитываемых при проектировании в соответствии с методологией IDEF1X. Рассматриваемая нотация при необходимости позволяет указать мощность связи количество экземпляров подчиненной сущности, которые могут быть связаны с экземпляром родительской сущности. Мощность может задаваться для связи типа «один-ко-многим», как идентифицирующей, так и не идентифицирующей. Могут быть определены следующие варианты:

* по умолчанию одному экземпляру родительской сущности может соответствовать 0, 1 или несколько экземпляров дочерней сущности; этот вариант на диаграмме специально не отмечается;
* Р указывает, что экземпляру родительской сущности соответствует 1 или более экземпляров дочерней сущности, символ ставится рядом с черной точкой в изображении связи;
* Z (от англ. zero ноль) указывает, что экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности
* число рядом с черной точкой указывает количество экземпляров подчиненной сущности; также можно задать диапазон или в скобках рядом с точкой указать комментарий, поясняющий возможные значения. Последний вариант указания мощности средой erwin Data Modeler не поддерживаются.

Примеры указания мощности связи приведены в табл. 6.1.

Задать мощность связи в ykwin можно в окне свойств (пункт «Properties» контекстного меню, открывающегося при щелчке правой клавишей мыши на изображении связи). Нужно отметить, что при настройках по умолчанию ERwin не отображает на диаграмме название и мощность связи. Если нужно включить отображение мощности связи на диаграмме, надо щелчком правой клавиши мыши на свободном месте поля диаграммы вызвать контекстное меню, в нем выбрать пункт «Properties» и на вкладке «Relationship» отметить опцию «Display Relationship Cardinality». Если нужно отобразить на диаграмме название связи, на той же вкладке надо отметить опцию «Display Logical Relationship Name»,

A white sheet with black text

AI-generated content may be incorrect.

Определяя в erwin связь типа «один-ко-многим», которая в большинстве случаев поддерживается на уровне СУБД с помощью внешнего ключа, можно уточнить, что должно происходить при изменениях в данных, таких как удаление экземпляра родительской сущности, для которой определены связанные дочерние. Подобные изменения могут привести к нарушению ссылочной целостности, и такие случаи должны специальным образом обрабатываться. Правило, определяющее поведение СУБД в каждом конкретном случае, называется правилом ссылочной целостности (англ. referentialintegrity, RI). Эти правила можно определить при проектировании в среде ERwin, а на уровне СУБД они будут реализованы или с помощью опций при создании внешних ключей, или с помощью триггеров. Триггер — это последовательность действий, описываемая на языке SQL и автоматически выполняемая при наступлении заданного события (подробнее о триггерах см. тему 9). Триггер ссылочной целостности (RI триггер) [13] особый вид триггера, используемый для поддержания ссылочной целостности между двумя таблицами, которые связаны между собой. Если строка в одной таблице вставляется, изменяется или удаляется, то RI триггер определяет, что нужно делать с теми строками в других таблицах, у которых значение внешнего ключа совпадает со значением первичного ключа вставленной (измененной, удаленной) строки.

Правила могут задаваться для шести случаев: удаление, добавление, изменение экземпляра дочерней сущности (англ. Child delete rule, child insert rule, child update rule соответственно) или родительской сущности (англ. parent delete rule, parent insert rule, parent update rule). Могут быть определены следующие действия:

* None никаких действий по поддержанию ссылочной целостности не требуется; •
* No action никаких действий не предпринимается; ·
* Cascade действие «каскадируется», например, при удалении экземпляра родительской сущности все связанные экземпляры дочерней сущности будут удалены; ·
* Restrict действие запрещается, если оно может привести к нарушению ссылочной целостности; например, попытка удаления экземпляра родительской сущности будет отклонена, если у нее есть связанные экземпляры дочерней сущности; ·
* Set Null после удаления объекта ссылки внешнего ключа внешнему ключу будет установлено значение Null (не определено);
* Set Default в аналогичной предыдущему случаю ситуации внешний ключ получит значение по умолчанию, если оно для него определено.

Например, зададим для связи между сущностями «Помещение» и «Компьютер» (рис. 6.7) в Parent Delete Rule действие Set Null. Тогда в случае удаления описания помещения во всех описаниях компьютеров, располагавшихся в этом помещении, соответствующий внешний ключ получит значение Null (т. е. «не определено»). Рассмотрим теперь некоторые частные случаи определения связей между сущностями. Начнем с парных связей. Связь между двумя сущностями называется бинарной. Если участников связи n (n > 2), связь будет парной. Строго говоря, нотация IDEF1X определяет правила изображения только бинарных связей, а для представления парных вводится дополнительная сущность. На рис. 6.8 представлен пример описания подобной связи между сущностями «Компания», «Продукт» и «Заказчик». Сущность «Контракт» описывает заказ товара у поставщика указанным клиентом в заданную дату.

Представленная на рис. 6.8 сущность «Контракт» является ассоциативной (англ. associative) [12] она связана с несколькими родительскими сущностями и содержит дополнительную информацию о связи этих сущностей. Если дополнительных атрибутов нет, а есть только внешние ключи, такая сущность будет называться именующей [12] (англ. designative, может встречаться перевод «указательная» и другие близкие варианты).

A diagram of a computer

AI-generated content may be incorrect.