КОНСПЕКТ ЗАНЯТИЯ № 1 ШЕСТОЙ НЕДЕЛИ КУРСА «БАЗЫ ДАННЫХ»

1. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ.

На этой неделе мы рассмотрим методы и средства моделирования данных. Вначале дается краткое описание тройственной схематической модели ANSI/SPARC (мы подробно разбирали ее на первом занятии). С помощью этой модели разъясняются роль и место моделирования данных в процессе проектирования баз данных. Далее описывается модель «сущность—связь» в том виде, как она изначально была предложена. Как вы далее увидите, есть существенные соображения в пользу того, чтобы изучать эту модель именно в такой форме. После мы кратко рассмотрим нотации Чена, Баркера и Мартина. В завершении рассматривается более новая версия модели «сущность—связь» — модель IDEFX1, получившая наибольшее распространение в современной промышленности.

1.1. Тройственная схематическая модель ANSI/SPARC

Тройственная схематическая модель базы данных ANSI/SPARC, предложенная Комитетом по планированию и разработке требований к стандартам (Standards Planning and Requirements Committee, SPARC) Американского национального института стандартов (American National Standards Institute, ANSI), была впервые опубликована в 1975 году. Несмотря на то, что эта модель уже довольно стара, с ее помощью исключительно удобно описывать роль и цели моделирования данных.

Модель включает в себя три схемы — внешнюю, концептуальную внутреннюю (рис. 2.1). *Внешняя схема* (external schema), называемая также пользовательским представлением (user view), описывает то, как пользователи представляют себе базу данных. Для всех баз данных, кроме простейших, внешняя схема отображает лишь часть реальной базы данных.

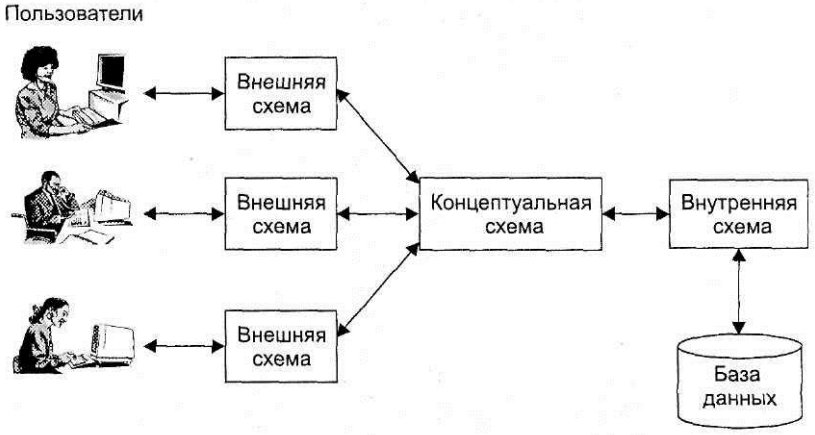


Рис. 2.1. Тройственная схематическая модель

Концептуальная схема (conceptual schema) — это полное логическое представление базы данных, включающее описание всех данных и связей между ними. Слова «логическое представление» имеют тот смысл, что концептуальная схема не зависит от конкретного способа хранения данных. Хранить концептуальную схему можно в базе данных, в файловом архиве, в виде набора связанных между собой электронных таблиц и другими способами.

Внутренняя схема (internal schema) — это представление, описывающее физическую реализацию концептуальной схемы с использованием конкретного продукта и/или технологии. Описание набора таблиц, ключей, внешних ключей, индексов и других физических структур представляет собой внутреннюю схему. Одна и та же концептуальная схема может быть представлена различными внутренними схемами: например, одна из них может быть для Oracle, другая — для DB2. Эти две внутренние схемы могут быть очень похожими или совершенно различными.

1.2. Модели данных и модели базы данных

Каждая система БД реализует ту или иную модель данных. Модель данных определяет правила порождения допустимых для системы видов структур данных, возможные операции над такими структурами, классы представимых средствами системы ограничений целостности данных. Таким образом, модель данных задает границы множества всех конкретных БД, которые могут быть созданы средствами этой системы.

Вспомним, какие модели данных выделяют:

- иерархическая (англ. hierarchical), конец 1960-х и 1970-е;
- сетевая (англ. network), 1970-е;
- реляционная (англ. relational), 1970-е и начало 1980-х;
- сущность-связь (англ. entity-relationship), 1970-е;
- расширенная реляционная (англ. extended relational), 1980-е;
- семантическая (англ. semantic), конец 1970-х и 1980-е;
- объектно-ориентированная (англ. object-oriented), конец 1980-х – начало 1990;
- объектно-реляционная (англ. object-relational), конец 1980-х – начало 1990-х;
- полуструктурированная (англ. semi-structured), конец 1990-х до настоящего времени.

Первыми появились модели данных, основанные на теории графов – *иерархическая* и *сетевая* (данные модели будут предложены вам в виде дополнительного материала для ознакомления, поскольку на сегодняшний день эти модели являются неакутальными). Следующей появилась разработанная Эдгаром Коддом (Edgar Codd) *реляционная модель* данных (мы посвятили разбору данной модели три занятия), основанная на математической теории множеств. На сегодняшний день она является самой распространенной.

Модель *«сущность-связь»* была предложена Питером Ченом (Peter Chen) в 1976 году, в качестве унифицированного способа описания предметной области. Как самостоятельная модель данных (в соответствии с приведенным выше определением), она развития не получила, но стала основой для создания ***инфологических моделей*** БД.

Естественно, что проект базы данных надо начинать с анализа предметной области и выявления требований к ней отдельных пользователей (сотрудников организации, для которых создается база данных).

Объединяя частные представления (т.е. объединяя внешние представления ANSI/SPARC) о содержимом базы данных, полученные в результате опроса пользователей, администратор баз данных (АБД) сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных, называют ***инфологической моделью*** баз данных (рис. 1.3). На этом этапе создается концептуальная схема ANSI/SPARC.

Рис. 1.3. Уровни моделей данных

Другими словами: ***инфологическая модель*** отражает информацию о предметной области без ориентации на конкретную СУБД (или даже на тип предполагаемой к использованию СУБД).

Остальные модели, показанные на рис. 1.3, являются компьютеро-ориентированными. С их помощью СУБД дает возможность программам и пользователям осуществлять доступ к хранимым данным лишь по их именам, не заботясь о физическом расположении этих данных. Нужные данные отыскиваются СУБД на внешних запоминающих устройствах по ***физической модели*** баз данных.

Физическая организация данных оказывает основное влияние на эксплуатационные характеристики БД. Разработчики СУБД пытаются создать наиболее производительные физические модели данных, предлагая пользователям тот или иной инструментарий для поднастройки модели под конкретную БД.

Другими словами: ***физическая модель*** БД строится с учетом возможностей по организации и хранению данных, предоставляемых СУБД и используемой программно-аппаратной платформой. Она, в частности, определяет используемые запоминающие устройства и способы организации данных в среде хранения.

Так как указанный доступ осуществляется с помощью конкретной СУБД, то модели должны быть описаны на языке описания данных этой СУБД. Такое описание, создаваемое АБД по инфологической модели данных, называют ***даталогической моделью*** баз данных.

Другими словами: ***даталогическая модель*** БД – модель логического уровня, представляющая собой отображение логических связей между элементами данных независимо от их содержания и среды хранения. Эта модель строится в терминах информационных единиц, допустимых в той СУБД, в среде которой будет создаваться БД. Этап создания данной модели называется даталогическим или логическим проектированием.

Трехуровневая архитектура (инфологический, даталогический и физический уровни) позволяет обеспечить *независимость хранимых данных* от использующих их программ. АБД может при необходимости переписать хранимые данные на другие носители информации и (или) реорганизовать их физическую структуру, изменив лишь физическую модель данных. АБД может подключить к системе любое число новых пользователей (новых приложений), дополнив, если надо, даталогическую модель. Указанные изменения физической и даталогической моделей не будут замечены существующими пользователями системы (окажутся "прозрачными" для них), так же как не будут замечены и новые пользователи. Следовательно, независимость данных обеспечивает возможность развития системы баз данных без разрушения существующих приложений.

При проектировании БД, первой строится инфологическая модель, после чего – даталогическая, и только после нее – физическая.

1.3. Построение концептуальной схемы (инфологическое проектирование)

Цель инфологического проектирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка).

Как уже было сказано, проектирование начинается с построения инфологической модели, т.е. построения концептуальной схемы. Сбор информации начинается с опроса пользователей: они предоставляют сведения о данных с точки зрения своей работы, то есть своего собственного представления (внешней схемы) данных. Нашей же целью будет, взяв за отправную точку все эти внешние схемы, построить единую концептуальную схему, которая бы поддерживала каждое из заданных нам на входе пользовательских представлений.

Не менее важно при создании концептуальной схемы четко отличать ее от внутренней схемы. Не следует смешивать физическое представление данных с их логическим представлением. В противном случае мы придем к тому, что характеристики СУБД будут ограничивать и искажать наше видение концептуальной модели. Из-за этого мы можем упустить из виду какой-то важный аспект в мире пользователей или, что еще хуже, позволить СУБД ограничивать свободу действий пользователей.

Поэтому, создавая концептуальные модели, старайтесь абстрагироваться от физических структур и элементов внутренней схемы. Вместо этого направьте свои усилия на то, чтобы предоставить пользователям нужные им внешние схемы, с помощью которых они могли бы успешно выполнять свою работу.

Конкретный пример [Сага о Брюсе и Зельде. Кренке Д. Теория и практике построения баз даннх].

Чтобы более наглядно продемонстрировать суть тройственной схематической модели, рассмотрим пример с Брюсом и Зельдой. Брюс — биолог, занимающийся изучением рек; он исследует загрязнение воды и берет пробы на наличие полихлорида бифенила (PCB) и других химических веществ. Несколько раз в году он объезжает реки и подсчитывает количество рыбы в них, так как эта статистка служит важным показателем благополучия речной среды. Результаты своих исследований он заносит в специальную форму (рис. 2.2), содержащую данные о реке и таблицу для записи количества рыбы.

Зельда, зоолог, занимается изучением рыбы. Основываясь на полученных ею данных, агентства отдельных штатов и федерального уровня определяют квоты на ловлю рыбы, налагают запрет на вылов определенных пород и оценивают состояние

здоровья различных видов рыбы. Для записи своих результатов Зельда также использует специальную форму (рис. 2.3). Она выбирает интересующий ее вид рыбы и заносит в таблицу данные по каждой реке, где встречает этот вид.

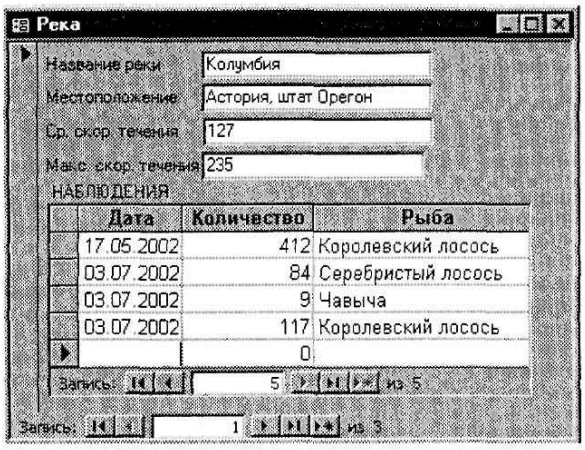


Рис. 2.2. Внешняя схема Брюса

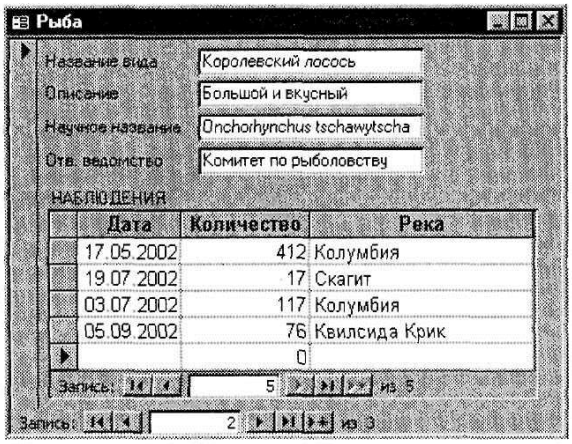


Рис. 2.3. Внешняя схема Зельды

Брюс и Зелъда работают в одном университете, и однажды университет решает создать единую базу данных, в которой содержалась бы вся информация о рыбе и реках. Первым шагом в этом деле, как известно, является интервьюирование пользователей, и вот некто назначает встречу с Брюсом и Зельдой.

Встреча происходит приблизительно так. Начинает разговор Зельда: «Брюс, я посмотрела, как ты записываешь свои результаты (см. рис. 2.2). У тебя же все вывернуто наизнанку! Сперва нужно выбрать интересующую нас породу рыбы, а потом занести в таблицу данные о количестве экземпляров данной породы в различных реках».

Брюс возражает: «Боюсь, Зельда, что формальдегид от твоих рыб ударил тебе в голову. Так никто не делает. Сначала надо выбрать определенную реку, а потом для данной конкретной реки записать статистику по всем породам рыбы, которые мы в ней обнаружим. Если бы я записывал свои данные так же, как это делаешь ты (см. рис. 2.3), это бы заняло у меня часы!»

Однако Зельда не сдается: «Напротив, Брюс, это твои мыслительные способности, похоже, пострадали от слишком большой дозы PCB. Если мы будем записывать результаты так, как предлагаешь *ты...».* Далее дебаты продолжаются в том же духе, и каждая из сторон доказывает, что именно ее способ записи является правильным. Хотите верьте, хотите нет, но на подобные споры были потрачены тысячи, если не миллионы часов. Проблема, разумеется, состоит в том, что Брюс и Зельда смотрят на одни и те же данные с совершенно разных позиций.

При разговоре Брюса и Зельды присутствует концептуализатор Конрад. Взглянув на формы, представленные на рис. 2.2 и 2.3, он говорит себе: «Эти формы — не что иное, как взгляд на одни и те же данные с двух различных точек зрения, то есть две внешние схемы. Нельзя ли объединить их в одной концептуальной схеме?»

Конрад приходит к выводу, что это сделать можно, и конструирует концептуальную схему, показанную на рис. 2.4. Как вы помните, концептуальная схема — это полное логическое представление данных. Она будет содержать множество других элементов, помимо тех, что показаны здесь, но пока нам хватит и этого.

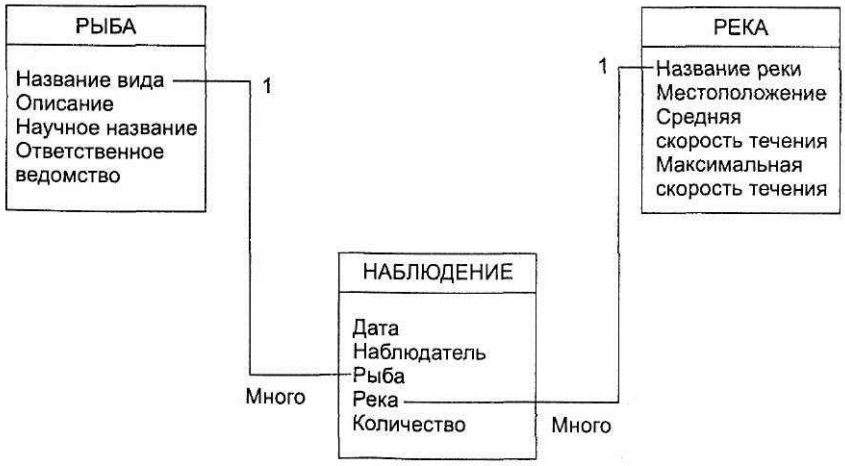


Рис. 2.4. Концептуальная схема базы данных по наблюдениям

Концептуальная схема Конрада (см. рис. 2.4) включает три различных группы данных: РЫБА, РЕКА и НАБЛЮДЕНИЕ. Линии представляют связи между ними. Обратите внимание, что с одной рекой потенциально мол-сет быть связано много наблюдений. То же самое справедливо и для рыбы.

Теперь, если следовать подходу Брюса, мы сперва выбираем нужную нам реку в группе РЕКА, затем в группе НАБЛЮДЕНИЕ находим записи обо всех наблюдениях, сделанных на данной реке, и из этих наблюдений получаем данные о количестве рыбы различных видов. Если потребуются какие-то дополнительные сведения о рыбе, помимо названия вида (столбцы Рыба и Название вида), мы возьмем их из группы РЫБА. С точки же зрения Зельды, мы сначала выбираем интересующий нас вид рыбы в группе РЫБА, затем в группе НАБЛЮДЕНИЕ находим записи обо всех наблюдениях, где фигурирует данный вид, и из этих записей извлекаем название реки. Дополнительные сведения о реках, помимо их названий (столбцы Река и Название реки), можно взять из группы РЕКА.

Концептуальная схема, изображенная на рис. 2.4, хороша для данного примера, однако для построения реальной базы данных необходимо гораздо больше подробностей. Далее в этой главе мы рассмотрим методы и средства, которые используются для представления таких концептуальных схем при проектировании реальных баз данных.

Но прежде мы доведем до конца нашу историю. Конрад передаст свою концептуальную схему Оливии — специалисту по Oracle. Оливия, исходя из схемы Конрада, создаст внутреннюю схему, или проект базы данных, для СУБД Oracle. Она спроектирует таблицы, определит ключи и внешние ключи, создаст индексы, выделит пространство под таблицы на физических носителях и примет ряд других

решений относительно структуры внутренней схемы. Затем она физически реализует созданную схему в виде базы данных и сопутствующих ей структур.

1.4. Модель "сущность-связь"

Инфологическая модель данных является начальным прототипом будущей базы данных. Она строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно *реляционной* модели. Основным средством разработки логической модели в настоящий момент являются различные варианты ***ER-диаграмм (диаграммы сущность-связь).*** Одну и ту де ER-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных.

Моделирование структуры базы данных при помощи алгоритма нормализации, описанного в предыдущих разделах, имеет серьезные недостатки:

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

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

Первый вариант модели сущность-связь был предложен в 1976 г. Питером Пин-Шэн Ченом. Позже появилась расширенная модель «сущность—связь» (extended E-R model). Сегодня, как правило, используют расширенный вариант.

Многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). По сути, все варианты диаграмм сущность-связь исходят из одной идеи - *рисунок всегда нагляднее текстового описания*. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

1.4.1. Выбор версии модели

Проще всего было бы вскинуть руки и сказать: «Чем связываться со всеми этими версиями, давайте лучше сосредоточимся на расширенной модели „сущность-связь", а остальные версии рассмотрим потом, если понадобится». Проблема заключается в том, что когда модель IDEF1X стала национальным стандартом, компании, занимавшиеся разработкой средств для моделирования данных, были вынуждены обеспечить соответствие своих продуктов этому стандарту, чтобы иметь возможность продавать их правительственным учреждениям. Игнорировать такой большой рынок было бы непростительно, поэтому большинство популярных в настоящее время программных продуктов для моделирования данных, например, ERWin и Visio, используют IDEF1X.

В то же время, растущая популярность объектно-ориентированной системной разработки и объектно-ориентированного программирования дала толчок распространению UML. Однако изначально UML предназначался в первую очередь для моделирования процессов и программ, и, по мнению многих, он пока не готов к использованию в качестве полноценного средства моделирования данных. Должно пройти еще какое-то время, прежде чем UML достигнет той стадии зрелости, на которой находятся модель IDEF1X и информационная инженерия. Кроме того, если вы не знакомы с концепциями объектно-ориентированного программирования, UML может оставить у вас странное впечатление.

По правде говоря, какое бы решение мы ни приняли в этой ситуации, оно все равно будет неудовлетворительным. Тем не менее, наш подход будет таков: знание классической расширенной модели «сущность—связь» необходимо, потому что содержащиеся в ней идеи и символы лежат в основе всех без исключения версий, а также потому, что данная модель столь широко распространена в промышленности. К несчастью, без знания модели IDEF1X также не обойтись, поскольку нам предстоит работать со средствами моделирования данных, а они по большей части следуют именно этой версии.

Исходя из вышесказанного, структура этой главы будет такова. Сначала будет описана расширенная модель «сущность—связь» в ее классическом варианте. Далее мы представим ER-диаграммы в нотации Чена, ER-диаграммы в нотациях Баркера и Мартина. В заключительном разделе рассмотрим стандарт IDEF1X, который не только использует другие символы, но и по-другому классифицирует связи.

1.4.2. Элементы модели «сущность-связь» (описание подходит для всех вариаций: расширенной модели, нотации Чена, Баркера и Мартина, IDEF1X)

Ключевыми элементами модели «сущность—связь» являются сущности, атрибуты, идентификаторы и связи. Рассмотрим каждый из них по очереди.

Определение 1. ***Сущность*** - это класс однотипных объектов, информация о которых должна быть учтена в модели.

Каждая сущность должна иметь наименование, выраженное существительным в единственном числе (примерами сущностей могут быть такие классы объектов как "Поставщик", "Сотрудник", "Накладная").

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, то «Клиент» – правильная сущность.

Определение 2. ***Экземпляр сущности*** - это конкретный представитель данной сущности.

Экземпляры сущностей должны быть различимы, т.е. сущности должны иметь некоторые свойства, *уникальные* для каждого экземпляра этой сущности.

Важно уяснить разницу между *классом сущностей* и *экземпляром сущности*. Обычно класс сущностей содержит множество экземпляров сущности. Например, класс КЛИЕНТ содержит множество экземпляров — по одному на каждого клиента, о котором имеется запись в базе данных. Пример класса сущностей и двух экземпляров сущности показан на рис. 2.5.

Рис. 2.5. КЛИЕНТ: пример сущности

Определение 3. ***Атрибут сущности*** - это именованная характеристика, являющаяся некоторым свойством сущности.

Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Примерами атрибутов сущности "Сотрудник" могут быть такие атрибуты как "Табельный номер", "Фамилия", "Имя", "Отчество", "Должность", "Зарплата" и т.п.

Исходное определение модели «сущность—связь» включает в себя композитные атрибуты (composite attributes) и многозначные атрибуты (multi-valued attributes). В качестве примера композитного атрибута можно привести атрибут Адрес, состоящий из группы атрибутов {Улица, Город, Штат, Индекс}. Примером многозначного атрибута может служить атрибут ДоверенноеЛицо сущности КЛИЕНТ, который может содержать имена нескольких доверенных лиц данного клиента. Атрибут может быть одновременно и композитным, и многозначным — например, композитный атрибут Телефон, состоящий из группы атрибутов {КодРегиона, МестныйНомер}, будучи многозначным, позволяет иметь в базе данных несколько телефонных номеров одного и того же лица. В большинстве версий модели «сущность—связь» однозначные композитные атрибуты игнорируются, и требуется, чтобы многозначные атрибуты преобразовывались в сущности. (*В частности, реляционная модель не поддерживает многозначные и составные атрибуты. Чтобы решить эту проблему обычно вводят дополнительное реляционное отношение).*

Определение 4. ***Ключ сущности (идентификаторы)*** **-** это неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из ключа нарушается его уникальность (Например, экземпляры сущностей класса СОТРУДНИК могут идентифицироваться по атрибутам НомерСоциальнойСтраховки или ТабельныйНомерСотрудника).

Сущность может иметь несколько различных ключей.

Определение 5. ***Связь*** - это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою.

Связи позволяют по одной сущности находить другие сущности, связанные с нею.

Например, связи между сущностями могут выражаться следующими фразами - "СОТРУДНИК может иметь несколько ДЕТЕЙ", "каждый СОТРУДНИК обязан числиться ровно в одном ОТДЕЛЕ".

Связь может соединять несколько сущностей. Число сущностей, участвующих в связи, называется степенью связи (relationship degree). Изображенная на рис. 2.6, а связь ПРОДАВЕЦ—ЗАКАЗ имеет степень 2, поскольку в ней участвуют две

сущности — ПРОДАВЕЦ и ЗАКАЗ. Связь РОДИТЕЛЬ на рис. 2.6, б имеет степень 3, так как в ней участвуют три сущности — МАТЬ, ОТЕЦ и РЕБЕНОК. Связи степени 2 весьма распространены, их часто называют еще бинарными связями (binary relationships).

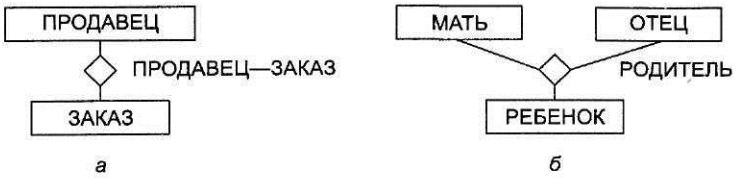


Рис. 2.6. Различные степени связей: *а* — связь степени 2; *б* — связь степени 3

Пусть R – тип связи, E – тип сущности. Если каждый экземпляр сущности типа E находится в одном экземпляре связи типа R, то участие E в R называется *полным* или *обязательным*, в противном случае – *частичным* или *необязательным*. Например, пусть рассматриваемая предметная область – 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.

Каждая связь может иметь один из следующих **типов связи**:

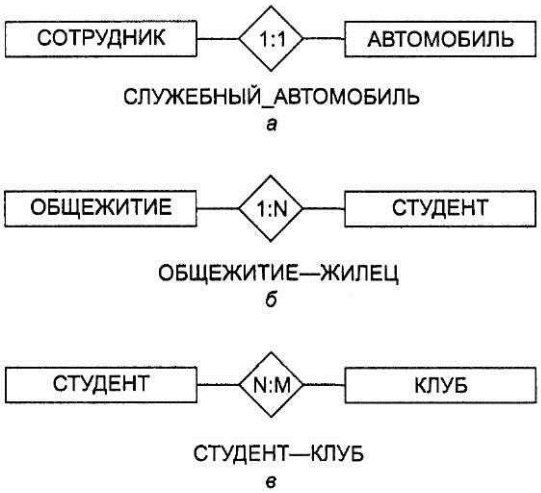


Рис. 2.7. Три типа бинарных связей: *а* — 1:1; *б* — 1:N; *в* — N:M

Связь типа один-к-одному (1:1) означает, что один экземпляр первой сущности связан с одним экземпляром второй сущности. Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две. *(На рис. 2.7а связь СЛУЖЕБНЫЙ_АВТОМОБИЛЬ связывает конкретного сотрудника с конкретным автомобилем. Согласно этой диаграмме, ни за одним сотрудником не закреплено более одного автомобиля, и ни один автомобиль не закреплен более чем за одним сотрудником).*

Связь типа один-ко-многим (1:N) означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Это наиболее часто используемый тип связи. Сущность со стороны "один" называется родительской, сущность со стороны "много" - дочерней. *(В этой связи, которая называется ОБЩЕЖИТИЕ—ЖИЛЕЦ, единичный экземпляр сущности класса ОБЩЕЖИТИЕ связан со множеством экземпляров сущности класса СТУДЕНТ. В соответствии с этим рисунком, в общежитии проживает много студентов, но каждый студент живет только в одном общежитии.*

Важна позиция, в которой стоят 1 и N. Единица стоит на той стороне связи, где располагается ОБЩЕЖИТИЕ, а N стоит на той стороне связи, где располагается СТУДЕНТ. Если бы мы поменяли местами 1 и N и записали бы эту связь как N:l, получилось бы, что в общежитии проживает всего один студент, причем студент может жить в нескольких общежитиях. Это, разумеется, не так).

Связь типа много-ко-многим (N:M) означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи *много*-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности. *(Связь СТУДЕНТ—КЛУБ является примером N:M. Один студент может быть членом нескольких клубов, а в одном клубе может состоять много студентов*).

Числа внутри ромба, символизирующего связь, обозначают максимальное количество сущностей на каждой стороне связи. Эти ограничения называются *максимальными кардинальными числами*. Например, о связи, изображенной на рис. 2.7, б, говорят, что она обладает максимальной кардинальностью 1:N. Кардинальные числа могут иметь и другие значения, а не только 1 и N. Например, связь между сущностями БАСКЕТБОЛЫНАЯ_КОМАНДА и ИГРОК может иметь кардинальность 1:5, что говорит нам о том, что в баскетбольной команде может быть не более пяти игроков.

Связи трех типов, представленных на рис. 2.7, называются *связями* принадлежности (HAS-A relationships).

1.4.3. Диаграммы расширенной модели «сущность-связь»

Схемы, изображенные на рис. 2.7, называются *диаграммами «сущность—связь»* (entity-relationship diagrams, ER-diagrams). Как в оригинальной, так и в расширенной модели «сущность—связь» классы сущностей обозначаются прямоугольниками, связи обозначаются ромбами, а максимальное кардинальное число каждой связи указывается внутри ромба. Имя сущности указывается внутри

прямоугольника, а имя связи указывается рядом с ромбом. В других версиях модели используются другие символы, как вы увидите позже.

Как мы уже говорили, максимальная кардинальность показывает максимальное число сущностей, которые могут участвовать в связи. Каково же минимальное число таких сущностей, приведенные диаграммы не сообщают. Например, рис. 2.7, б показывает, что студент может проживать максимум в одном общежитии, однако из него не ясно, обязан ли студент проживать в каком-либо общежитии.

Для указания минимальной кардинальности существует несколько способов. Один из них, продемонстрированный на рис. 2.8, заключается в следующем: чтобы показать, что сущность обязана участвовать в связи, на линию связи помещают перпендикулярную черту, а чтобы показать, что сущность может (но не обязана) участвовать в связи, на линию связи помещают овал. Соответственно, рис. 2.8 показывает, что сущность ОБЩЕЖИТИЕ должна быть связана как минимум с одной сущностью СТУДЕНТ, однако сущность СТУДЕНТ не обязана иметь связь с сущностью ОБЩЕЖИТИЕ. Полный набор накладываемых на связь ограничений состоит в том, что ОБЩЕЖИТИЕ имеет минимальное кардинальное число, равное единице, и максимальное кардинальное число, равное «многим» сущностям СТУДЕНТ. СТУДЕНТ имеет минимальное кардинальное число, равное нулю, и максимальное кардинальное число, равное одному экземпляру сущности ОБЩЕЖИТИЕ.

Рис. 2.8. Связь с указанной минимальной кардинальностью

Может существовать связь между сущностями одного и того же класса. Например, для сущностей класса СТУДЕНТ может быть определена связь СОСЕД_ПО_ КОМНАТЕ. Связи между сущностями одного и того же класса называются иногда рекурсивными связями.

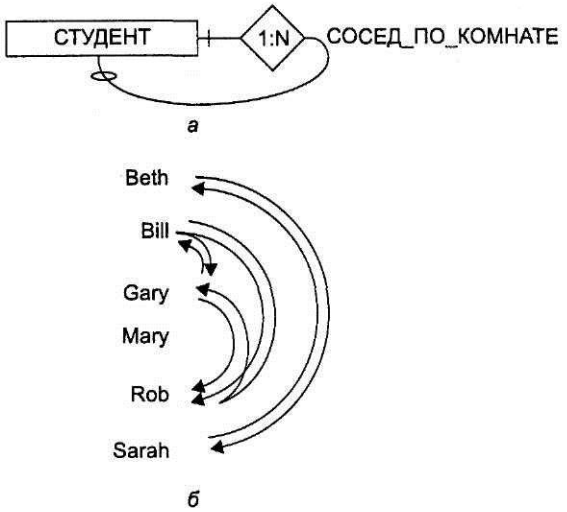


Рис. 2.9. Рекурсивная связь

Атрибуты в некоторых версиях диаграмм «сущность—связь» обозначаются эллипсами, соединенными с сущностью или связью, которой они принадлежат. На рис. 2.10 показаны сущности ОБЩЕЖИТИЕ и СТУДЕНТ и связь ОБЩЕЖИТИЕ—ЖИЛЕЦ с принадлежащими им атрибутами. Как видно из рисунка, сущность ОБЩЕЖИТИЕ имеет атрибуты НазваниеОбщежития, Местоположение и КоличествоКомнат, а сущность СТУДЕНТ имеет атрибуты НомерСтудента, ИмяСтудента и Курс. Связь ОБЩЕЖИТИЕ-ЖИЛЕЦ имеет атрибут Плата, который показывает внесенную студентом плату за проживание в конкретном общежитии.

Если сущность имеет много атрибутов, такое их перечисление на диаграмме может сделать ее чересчур громоздкой и трудной для восприятия.

В других версиях модели, как вы увидите далее, атрибуты отображаются по-другому.

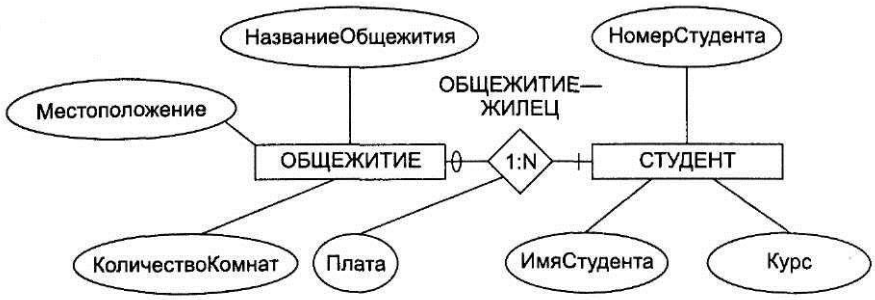


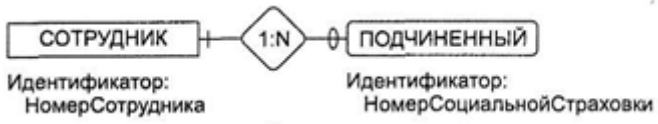
Рис. 2.10. Изображение свойств на диаграммах «сущность—связь»

Слабые сущности

Как уже было отмечено сущности в свою очередь делятся на два класса: сильные (правильные) сущности и слабые.

Пример: рассмотрим базу данных отдела кадров с классами сущностей СОТРУДНИК и ПОДЧИНЕННЫЙ. Предположим, существует правило, что экземпляр сущности СОТРУДНИК может существовать, не будучи связанным ни с одной сущностью класса ПОДЧИНЕННЫЙ, но сущность ПОДЧИНЕННЫЙ не может существовать вне связи с какой-либо сущностью класса СОТРУДНИК. Тогда сущность ПОДЧИНЕННЫЙ является слабой. Это означает, что запись о сущности ПОДЧИНЕННЫЙ может появиться в базе данных только в том случае, если эта сущность имеет связь с какой-либо сущностью класса СОТРУДНИК.

Как показано на рисунке ниже, слабые сущности обозначаются прямоугольниками со скругленными углами. Кроме того, связь, от которой зависит существование сущности, обозначается ромбом со скругленными углами. В некоторых диаграммах прямоугольники для слабых сущностей рисуются двойной линией, а связи, от которых зависит существование этих сущностей, изображаются двойными ромбами.



Подтипы сущностей

Некоторые сущности имеют необязательные наборы атрибутов; эти сущности часто представляются с помощью *подтипов*. Рассмотрим, например, сущность КЛИЕНТ (рис. 2.14) с атрибутами НомерКлиента, ИмяКлиента и СуммаКОплате. Предположим, что клиент может быть физическим лицом, товариществом или корпорацией и что необходимо указывать некоторую дополнительную информацию, зависящую от типа клиента. Пусть эта информация имеет следующее содержание:

ФИЗИЧЕСКОЕ_ЛИЦО:

- Адрес,
- НомерСоциальнойСтраховки

ТОВАРИЩЕСТВО:

- ИмяУправляющегоПартнера,
- Адрес,
- ИдентификационныйНалоговыйНомер

КОРПОРАЦИЯ:

- КонтактноеЛицо,
- Телефон,
- ИдентификационныйНалоговыйНомер

Одна из возможностей — отнести все эти атрибуты к сущности КЛИЕНТ, как показано на рис. 2.14, а. В этом случае, однако, некоторые атрибуты будут неприменимы. Например, такой атрибут, как имя управляющего партнера, не имеет смысла для индивидуального или корпоративного клиента, и, таким образом, он не может иметь какое-либо значение.

В модели, более близкой к реальной ситуации, вместо этого будут определены три сущности-подтипа, как показано на рис. 2.14, б. Здесь сущности ФИЗИЧЕСКОЕ_ЛИЦО, ТОВАРИЩЕСТВО и КОРПОРАЦИЯ изображены как *подтипы* сущности КЛИЕНТ. Последняя, в свою очередь, является *надтипом* для сущностей ФИЗИЧЕСКОЕ_ЛИЦО, ТОВАРИЩЕСТВО и КОРПОРАЦИЯ.

Символ “Ɛ” рядом с линиями связи указывает, что сущности ФИЗИЧЕСКОЕ_ЛИЦО, ТОВАРИЩЕСТВО и КОРПОРАЦИЯ являются подтипами сущности КЛИЕНТ. Каждый подтип должен принадлежать надтипу КЛИЕНТ. Кривая линия с цифрой 1 рядом показывает, что сущность КЛИЕНТ должна

принадлежать к одному и только одному подтипу. Это означает, что подтипы являются взаимоисключающими и что требуется только один из них.

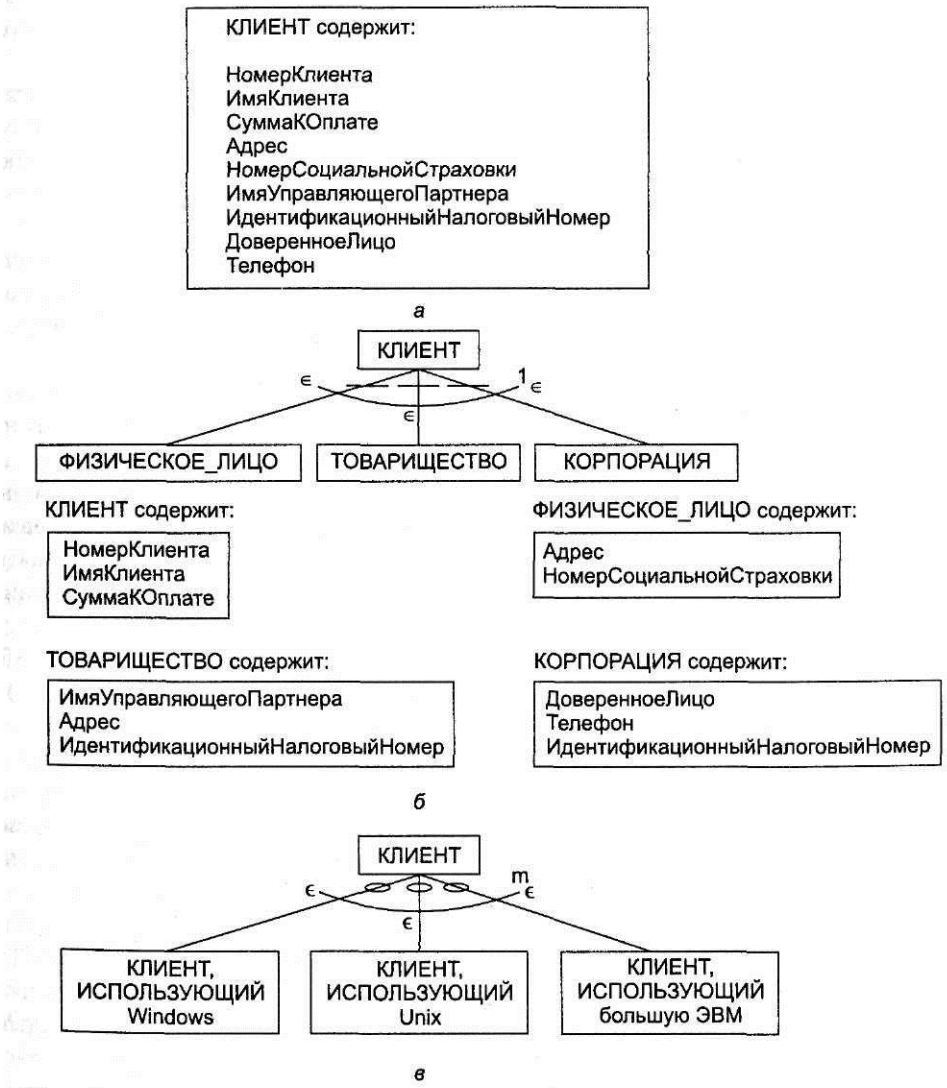


Рис. 2.14. Подтипы сущностей: *а* — КЛИЕНТ без подтипов; *б* — КЛИЕНТ с подтипами; *в* — невзаимоисключающие подтипы с необязательным надтипом

Сущности с такой связью должны иметь один и тот же идентификатор, поскольку они представляют различные аспекты одного и того же. В данном случае таким идентификатором является НомерКлиента.

Подтипы используются в моделировании данных с единственной целью: избежать ситуаций, при которых некоторые атрибуты должны иметь нулевые значения. Например, если атрибуту НомерСоциальнойСтраховки на рис. 2.14, а присвоено значение, то остальные четыре атрибута должны быть равны нулю.

Подтипы были добавлены в модель «сущность—связь» после публикации оригинальной статьи Чена, и они являются частью того, что называется расширенной моделью «сущность—связь».

Пример ER-диаграммы расширенной модели «сущность-связь»

На рис. 2.15 показан пример диаграммы, содержащей все элементы модели «сущность—связь». Она изображает сущности и связи для инженерной консалтинговой компании, которая занимается анализом строительства и состояния жилых домов, а также других зданий и сооружений.

На диаграмме есть класс сущностей, представляющий сотрудников компании. Поскольку некоторые сотрудники являются инженерами, сущность ИНЖЕНЕР связана с сущностью СОТРУДНИК как подтип. Каждый инженер должен быть сотрудником. Сущность ИНЖЕНЕР имеет связь 1:1 с сущностью ГРУЗОВИК: каждый грузовик должен быть закреплен за каким-то инженером, но не у всех инженеров есть грузовики.

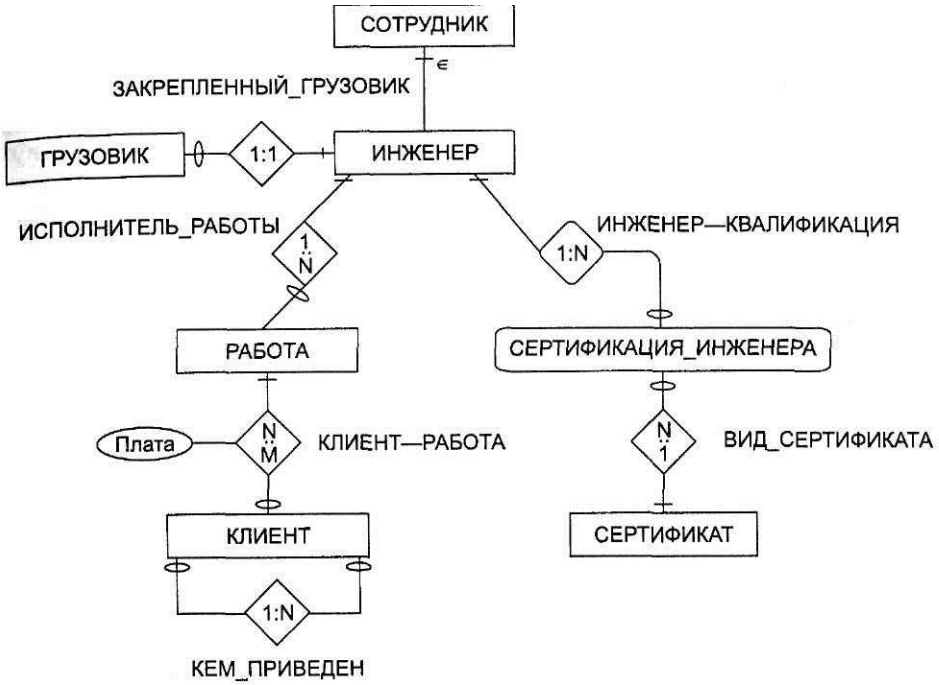


Рис. 2.15. Пример диаграммы «сущность—связь»

Инженеры выполняют работы (сущность РАБОТА) для клиентов (сущность КЛИЕНТ). Инженер может не выполнять никаких работ (иначе говоря, выполнять ноль работ) или выполнять много работ, но каждая отдельно взятая работа может выполняться только одним конкретным инженером. Для одного и того же клиента может выполняться много различных работ, и один и тот же вид работы может выполняться для множества клиентов. Связь КЛИЕНТ—РАБОТА имеет атрибут Плата, который показывает сумму, уплаченную конкретным клиентом за конкретную работу. (Прочие атрибуты сущностей и связей не показаны на этой диаграмме.)

Иногда одни клиенты приводят других, что показывается с помощью рекурсивной связи КЕМ_ПРИВЕДЕН. Клиент может привести одного или нескольких Других клиентов.

Сущность СЕРТИФИКАЦИЯ_ИНЖЕНЕРА показывает, что данный инженер получил образование и прошел тестирование, требуемое для получения конкретного

сертификата. Инженер может иметь сертификаты (сущность СЕРТИФИКАТ). Существование сущности СЕРТИФИКАЦИЯ_ИНЖЕНЕРА зависит от сущности ИНЖЕНЕР через связь ИНЖЕНЕР—КВАЛИФИКАЦИЯ.

1.4.4. ER-диаграммы в нотации Чена

В нотации Чена сущности изображаются прямоугольником, внутри которого помещается имя сущности. Прямоугольник, соответствующий слабой сущности, обводится двойной рамкой.

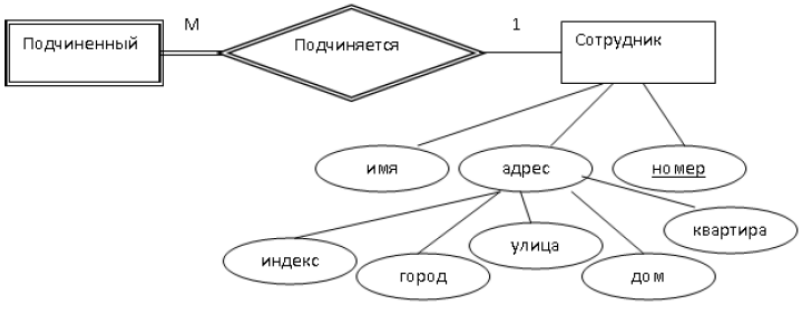
Атрибуты изображаются в виде овала, соединенного с соответствующим прямоугольником. Но обычно на диаграммах атрибуты вообще не отображают, в целях упрощения схемы. Ключевые атрибуты выделяются подчеркиванием или служебным символом в начале имени (например, «#»).

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.

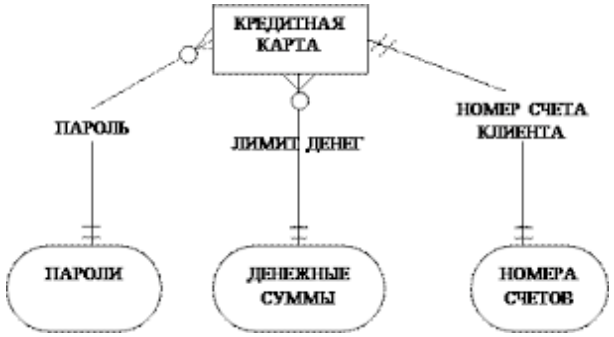
Для обозначения типа связи используются символы “1” и “M” (иногда вместо «M» используется символ бесконечности, или ставятся одинарные и двойные стрелки). Двойная линия обозначает полное участие сущности в связи.

Ассоциированные сущности изображаются ромбом, заключенным в прямоугольник. (*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 описывается набором собственных параметров: номер места, класс, цена, дата приобретения билета и т. д. Чтобы представить все эти данные, можно ввести составную сущность «Билет»*).

На рисунке ниже 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» – составной.



Детализация сущности осуществляется с использованием диаграмм атрибутов, которые раскрывают ассоциированные с сущностью атрибуты. Диаграмма атрибутов состоит из детализируемой сущности, соответствующих атрибутов и доменов, описывающих области значений атрибутов. На диаграмме каждый атрибут представляется в виде связи между сущностью и соответствующим доменом, являющимся графическим представлением множества возможных значений атрибута. Все атрибутные связи имеют значения на своем окончании. Для идентификации ключевого атрибута используется подчеркивание имени атрибута. Пример диаграммы атрибутов, детализирующей сущность *КРЕДИТНАЯ КАРТА* приведен ниже на рисунке.



Подтипы сущностей (сущности-категории) в нотации Чена

В данной нотации, как и в предыдущей версии модели «сущность-связь», рассматриваются подтипы, но нотация имеет свой собственный словарь терминов. Поэтому схожие понятия (в расширенной модели «сущность-связь» и нотация Чена) именуются иначе. К этому стоит привыкнуть.

Сущность может быть разделена и представлена в виде двух или более **сущностей-категорий**, каждая из которых имеет общие атрибуты и/или отношения, которые определяются однажды на верхнем уровне и наследуются на нижнем. Сущности-категории могут иметь и свои собственные атрибуты и/или отношения, а также, в свою очередь, могут быть декомпозированы своими сущностями-категориями на следующем уровне. Расщепляемая на категории сущность получила название **общей сущности.**

Для демонстрации декомпозиции сущности на категории используются **диаграммы категоризации**. Такая диаграмма содержит общую сущность, две и более сущности-категории и специальный **узел-дискриминатор,** который описывает способы декомпозиции сущностей.

Пример ER-диаграммы в нотации Чена

На рис.3.11 приведена диаграмма "сущность-связь", демонстрирующая отношения между объектами банковской системы. Согласно этой диаграмме каждый *БАНК ИМЕЕТ* один или более *БАНКОВСКИХ СЧЕТОВ*. Кроме того, каждый *КЛИЕНТ МОЖЕТ ВЛАДЕТЬ* (одновременно) одной или более *КРЕДИТНОЙ КАРТОЙ* и одним или более *БАНКОВСКИМ СЧЕТОМ*, каждый из которых *ОПРЕДЕЛЯЕТ* в точности одну *КРЕДИТНУЮ КАРТУ*(отметим, что у клиента может и не быть ни счета, ни кредитной карты). Каждая *КРЕДИТНАЯ КАРТА ИМЕЕТ* ровно один зависимый от нее *ПАРОЛЬ КАРТЫ*, а каждый *КЛИЕНТ ЗНАЕТ* (но может и забыть) *ПАРОЛЬ КАРТЫ*.

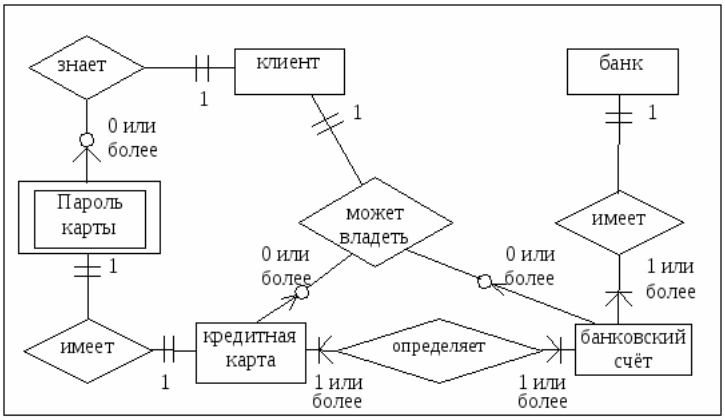


Рис. 3.11. ER-диаграмма в нотации Чена

1.4.5. ER-диаграммы в нотации Баркера и Мартина

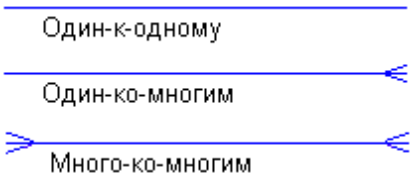
В начале 1980-х были предложены новые подходы к инфологическому проектированию баз данных, в большей степени ориентированных на БД реляционного типа. Среди работавших в этом направлении исследователей можно назвать Ричарда Баркера (Richard Barker) и авторов нотация Information Engineering (сокр. IE) Джеймса Мартина (James Martin) и Клива 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.

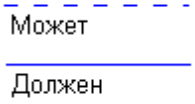
Все связи являются бинарными (т. е. только с двумя участниками) и изображаются линией, соединяющей сущности.

В нотации Баркера

связи представляются следующим образом:



Каждая связь может иметь одну из двух ***модальностей связи***:



Модальность "***может***" означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром.

Модальность "***должен***" означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности.

Другими словами, при обязательной связи рисуется непрерывная линия, при необязательной – пунктирная линия.

Описанный графический синтаксис позволяет *однозначно* читать диаграммы, пользуясь следующей схемой построения фраз:

<Каждый экземпляр СУЩНОСТИ_1> <МОДАЛЬНОСТЬ СВЯЗИ> <НАИМЕНОВАНИЕ СВЯЗИ> <ТИП СВЯЗИ> <экземпляр СУЩНОСТИ_2>.

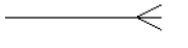
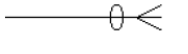
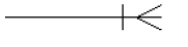
Каждая связь может быть прочитана как слева направо, так и справа налево.

Слева направо: "каждый сотрудник *может иметь* несколько детей".

Справа налево: "Каждый ребенок *обязан принадлежать* ровно одному сотруднику".

В нотации Мартина

связи представляются следующим образом:

Обозначение	Кардинальность
	нет
	1,1
	0,1
	M,N
	0,N
	1,N

Таким образом, в нотации Мартина происходит некое смешивание нотации Баркера и расширенной модели «сущность-связь».

Обе нотации предусматривают использование принципов подтипов сущности.

Нужно отметить, что из-за особенностей изображения связей нотации Баркера и Мартина в литературе иногда называют crow’s foot notation (дословно-нотация вороньей лапки).

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.

Рис. 6.3. ER-диаграмма в нотации Мартина

Переход от диаграммы к реляционной модели

Перед тем, как перейти к стандарту IDEFI1X, который получил широкое распространение в проектировании логической модели, обратимся к «алгоритму» перехода от диаграмм к реляционным отношениям.

Был выработан ряд правил, облегчающих «конвертацию» диаграммы в отношение:

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.

5) Атрибуты сущностей приводятся к атрибутам отношений.

Пример разработки простой ER-модели

На основе полученных знаний по ER-моделированию, попробуем разработать инфологическую модель предметной области. Для этого необходимо придерживаться следующего порядка получения информации:

1. Выявить список сущностей предметной области.
2. Выявить список атрибутов сущностей.
3. Описать взаимосвязи между сущностями.

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

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

Например, в ходе беседы с менеджером по продажам, выяснилось, что он (менеджер) считает, что проектируемая система должна выполнять следующие действия:

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

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

Покупатель - явный кандидат на сущность.

Накладная - явный кандидат на сущность.

Товар - явный кандидат на сущность

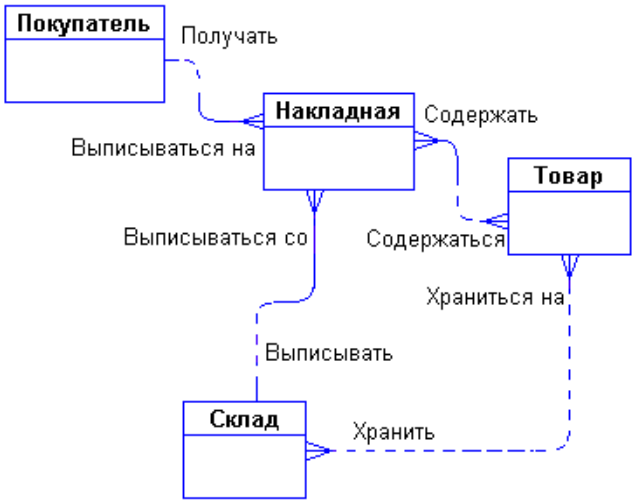
(?)Склад - а вообще, сколько складов имеет фирма? Если несколько, то это будет кандидатом на новую сущность.

(?)Наличие товара – это, скорее всего, атрибут, но атрибут какой сущности?

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

Задав дополнительные вопросы менеджеру, мы выяснили, что фирма имеет несколько складов. Причем, каждый товар может храниться на нескольких складах и быть проданным с любого склада.

Куда поместить сущности "Накладная" и "Склад" и с чем их связать? Спросим себя, как связаны эти сущности между собой и с сущностями "Покупатель" и "Товар"? Покупатели покупают товары, получая при этом накладные, в которые внесены данные о количестве и цене купленного товара. Каждый покупатель может получить несколько накладных. Каждая накладная обязана выписываться на одного покупателя. Каждая накладная обязана содержать несколько товаров (не бывает пустых накладных). Каждый товар, в свою очередь, может быть продан нескольким покупателям через несколько накладных. Кроме того, каждая накладная должна быть выписана с определенного склада, и с любого склада может быть выписано много накладных. Таким образом, после уточнения, диаграмма будет выглядеть следующим образом:



Пора подумать об атрибутах сущностей. Беседуя с сотрудниками фирмы, мы выяснили следующее:

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

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

Юридическое лицо - термин риторический, мы не работаем с физическими лицами. Не обращаем внимания.

Наименование покупателя - явная характеристика покупателя.

Адрес - явная характеристика покупателя.

Банковские реквизиты - явная характеристика покупателя.

Наименование товара - явная характеристика товара.

(?)Цена товара - похоже, что это характеристика товара. Отличается ли эта характеристика от цены в накладной?

Единица измерения - явная характеристика товара.

Номер накладной - явная уникальная характеристика накладной.

Дата накладной - явная характеристика накладной.

(?)Список товаров в накладной - список не может быть атрибутом. Вероятно, нужно выделить этот список в отдельную сущность.

(?)Количество товара в накладной - это явная характеристика, но характеристика чего? Это характеристика не просто "товара", а "товара в накладной".

(?)Цена товара в накладной - опять же это должна быть не просто характеристика товара, а характеристика товара в накладной. Но цена товара уже встречалась выше - это одно и то же?

Сумма накладной - явная характеристика накладной. Эта характеристика не является независимой. Сумма накладной равна сумме стоимостей всех товаров, входящих в накладную.

Наименование склада - явная характеристика склада.

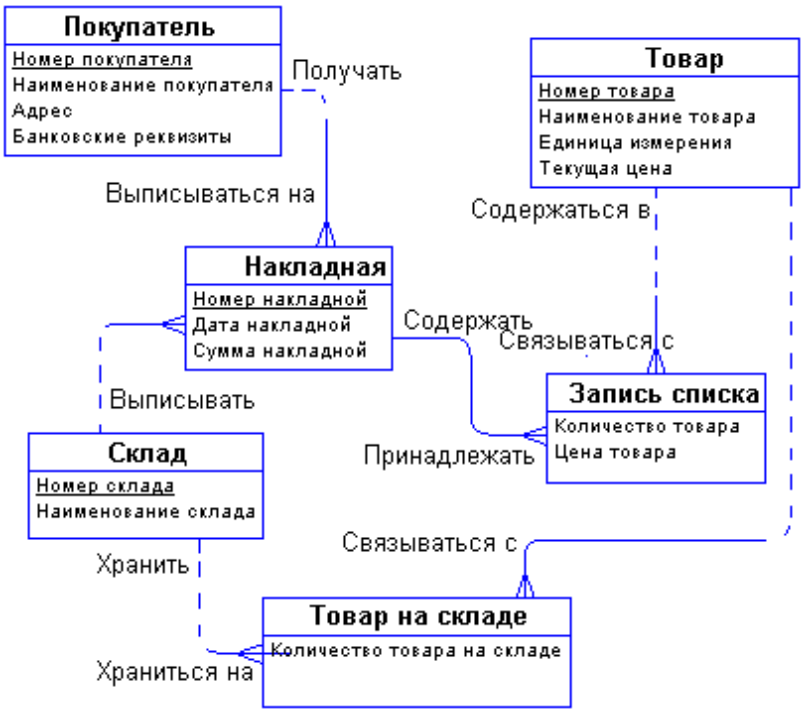
В ходе дополнительной беседы с менеджером удалось прояснить различные понятия цен. Оказалось, что каждый товар имеет некоторую текущую цену. Эта цена, по которой товар продается в данный момент. Естественно, что эта цена может меняться со временем. Цена одного и того же товара в разных накладных,

выписанных в разное время, может быть различной. Таким образом, имеется *две цены* - цена товара в накладной и текущая цена товара.

С возникающим понятием "Список товаров в накладной" все довольно ясно. Сущности "Накладная" и "Товар" связаны друг с другом отношением типа много-ко-многим. Такая связь, как мы отмечали ранее, должна быть расщеплена на две связи типа один-ко-многим. Для этого требуется дополнительная сущность. Этой сущностью и будет сущность "Список товаров в накладной". Связь ее с сущностями "Накладная" и "Товар" характеризуется следующими фразами - "каждая накладная обязана иметь несколько записей из списка товаров в накладной", "каждая запись из списка товаров в накладной обязана включаться ровно в одну накладную", "каждый товар может включаться в несколько записей из списка товаров в накладной", " каждая запись из списка товаров в накладной обязана быть связана ровно с одним товаром". Атрибуты "Количество товара в накладной" и "Цена товара в накладной" являются атрибутами сущности " Список товаров в накладной".

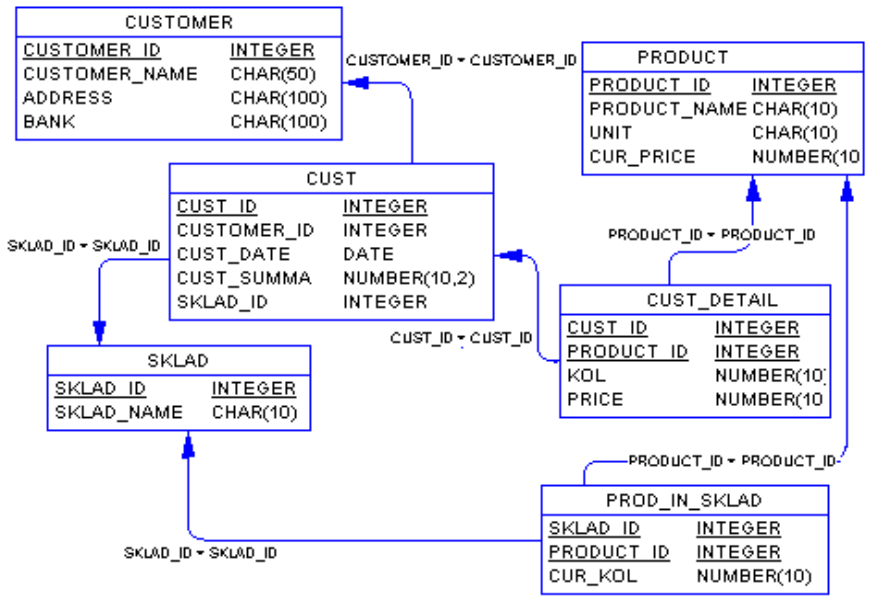
Точно также поступим со связью, соединяющей сущности "Склад" и "Товар". Введем дополнительную сущность "Товар на складе". Атрибутом этой сущности будет "Количество товара на складе". Таким образом, товар будет числиться на любом складе и количество его на каждом складе будет свое.

Теперь можно внести все это в диаграмму:



Разработанный выше пример ER-диаграммы является примером ***концептуальной диаграммы***. Это означает, что диаграмма *не учитывает* особенности конкретной СУБД. По данной концептуальной диаграмме можно построить ***физическую диаграмму*** (перевод диаграммы в физическую модель будет

рассмотрен на следующей неделе), которая уже будут учитываться такие особенности СУБД, как допустимые типы и наименования полей и таблиц, ограничения целостности и т.п. Физический вариант диаграммы, приведенной на может выглядеть, например, следующим образом:



Легко заметить, что полученные таблицы сразу находятся в 3НФ.

Case-средства

Задача проектирования базы данных для современной информационной системы корпоративного уровня может быть достаточно трудоёмкой и требовать совместной работы большой группы специалистов – аналитиков, разработчиков баз данных, разработчиков прикладного программного обеспечения (ПО), специалистов в предметной области, для которой разрабатывается БД. Для автоматизации этого процесса широко используются CASE-средства (от англ. Computer-Aided Software Engineering) – программные средства, поддерживающие одну или несколько технологий проектирования БД (также есть средства проектирования ПО и т. д.). В качестве примера можно назвать программные продукты ERwin Data Modeler (разработчик – компания CA Technologies), ER/Studio (разработчик – Embarcadero Technologies), PowerDesigner (разработчик – компания Sybase, в настоящее время приобретенная SAP). Отчасти, подобная функциональность реализована и в популярном офисном программном продукте Microsoft Visio.

ERwin и подобные ему CASE-средства позволяют решать как задачи прямого проектирования (англ. forward-engineering), т. е. получения структуры БД на основе построенной ER-диаграммы, так и обратного проектирования (англ. reverse-engineering), когда ER-диаграмма создается на основе анализа структуры существующей БД.

Выводы

Реальным средством моделирования данных является не формальный метод нормализации отношений, а так называемое ***семантическое моделирование***.

В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность-связь*** (***ER - Entity-Relationship***).

Диаграммы сущность-связь позволяют использовать наглядные графические обозначения для моделирования сущностей и их взаимосвязей.

Различают ***концептуальные*** и ***физические*** ER-диаграммы. Концептуальные диаграммы не учитывают особенностей конкретных СУБД. Физические диаграммы строятся по концептуальным и представляют собой прообраз конкретной базы данных. Сущности, определенные в концептуальной диаграмме становятся таблицами, атрибуты становятся колонками таблиц (при этом учитываются допустимые для данной СУБД типы данных и наименования столбцов).

При правильном определении сущностей, полученные таблицы будут сразу находиться в 3НФ. Основное достоинство метода состоит в том, модель строится методом последовательных уточнений первоначальных диаграмм.