План

1. Этапы проектирования
2. Описание процесса проектирования на упрощенном примере предметной области «Фильмотека»
3. Модель данных. Нотация Чена.
    a. Тип и экземпляр атрибута и сущности
    b. Степень и мощность связи
    c. Типы бинарных связей (1:1, 1:N, M:N)
    d. Рекурсивные связи
    e. Составные и многозначные атрибуты
    f. Слабые сущности. Зависимость существования и идентификационная зависимость.
    g. Преобразование множественных атрибутов при помощи слабых сущностей
    h. Подтипы сущностей

Часть 1. Этапы проектирования

Проектирование БД

Принципы проектирования БД. Анализ предметной области. Бизнес-логика. Отношения "сущность-связь". Нормализация и денормализация. Естественные и суррогатные ключи. Модель БД.

Предметная область - это часть реального мира, данные о которой мы хотим отразить в базе данных. Например, в качестве предметной области можно выбрать бухгалтерию какого-либо предприятия, отдел кадров, банк, магазин и т.д. Предметная область бесконечна и содержит как существенно важные понятия и данные, так и малозначащие или вообще не значащие данные. Так, если в качестве предметной области выбрать учет товаров на складе, то понятия "накладная" и "счет-фактура" являются существенно важными понятиями, а то, что сотрудница, принимающая накладные, имеет двоих детей - это для учета товаров неважно. Однако с точки зрения отдела кадров данные о наличии детей являются существенно важными. Таким образом, важность данных зависит от выбора предметной области и требований пользователей.

Модель предметной области. Модель предметной области - это наши знания о предметной области. Знания могут быть как в виде неформальных знаний в мозгу эксперта, так и выражены формально при помощи каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т.п. Опыт показывает, что текстовый способ представления модели предметной области крайне неэффективен. Гораздо более информативными и

полезными при разработке баз данных являются описания предметной области, выполненные при помощи специализированных графических нотаций. Имеется большое количество методик описания предметной области. Из наиболее известных можно назвать методику структурного анализа SADT и основанную на нем IDEF0, диаграммы потоков данных Гейна-Сарсона, методику объектно-ориентированного анализа UML, и др. Модель предметной области описывает скорее ***процессы***, происходящие в предметной области и данные, используемые этими процессами. От того, насколько правильно смоделирована предметная область, зависит успех дальнейшей разработки приложений.

Логическая модель данных. На следующем, более низком уровне находится логическая модель данных предметной области. Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Примеры понятий - "сотрудник", "отдел", "проект", "зарплата". Примеры взаимосвязей между понятиями - "сотрудник числится ровно в одном отделе", "сотрудник может выполнять несколько проектов", "над одним проектом может работать несколько сотрудников". Примеры ограничений - "возраст сотрудника не менее 16 и не более 60 лет".

Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных. ***Основным средством разработки*** логической модели данных в настоящий момент являются различные ***варианты ER-диаграмм*** (*Entity-Relationship*, *диаграммы сущность-связь*). Одну и ту же ER-модель можно преобразовать как в реляционную модель данных, так и в модель данных для иерархических и сетевых СУБД, или в постреляционную модель данных. Однако, т.к. мы рассматриваем именно реляционные СУБД, то можно считать, что логическая модель данных для нас формулируется в терминах реляционной модели данных.

Решения, принятые на предыдущем уровне, при разработке модели предметной области, определяют некоторые границы, в пределах которых можно развивать логическую модель данных, в пределах же этих границ можно принимать различные решения. Например, модель предметной области складского учета содержит понятия "склад", "накладная", "товар". При разработке соответствующей реляционной модели эти термины обязательно должны быть использованы, но различных способов реализации тут много - можно создать одно отношение, в котором будут присутствовать в качестве атрибутов "склад", "накладная", "товар", а можно создать три отдельных отношения, по одному на каждое понятие.

При разработке логической модели данных возникают вопросы: хорошо ли спроектированы отношения? Правильно ли они отражают модель предметной области, а следовательно и саму предметную область?

Физическая модель данных. На еще более низком уровне находится физическая модель данных. Физическая модель данных описывает данные средствами конкретной СУБД. Будем считать, что физическая модель данных реализована средствами именно реляционной СУБД, хотя, как уже сказано выше, это необязательно. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом, как уже говорилось ранее, решения, принятые на уровне логического моделирования определяют некоторые границы, в пределах которых можно развивать физическую модель данных. Точно также, в пределах этих границ можно принимать различные решения - например, о том, что отношения, содержащиеся в логической модели данных, должны быть преобразованы в таблицы, но для каждой таблицы можно дополнительно объявить различные индексы, повышающие скорость обращения к данным. Многое тут зависит от конкретной СУБД.

При разработке физической модели данных возникают вопросы: хорошо ли спроектированы таблицы? Правильно ли выбраны индексы? Насколько много программного кода в виде триггеров и хранимых процедур необходимо разработать для поддержания целостности данных?

Собственно база данных и приложения. И, наконец, как результат предыдущих этапов появляется собственно сама база данных. База данных реализована на конкретной программно-аппаратной основе, и выбор этой основы позволяет существенно повысить скорость работы с базой данных. Например, можно выбирать различные типы компьютеров, менять количество процессоров, объем оперативной памяти, дисковые подсистемы и т.п. Очень большое значение имеет также настройка СУБД в пределах выбранной программно-аппаратной платформы.

Но опять решения, принятые на предыдущем уровне – уровне физического проектирования, определяют границы, в пределах которых можно принимать решения по выбору программно-аппаратной платформы и настройки СУБД.

Таким образом, ясно, что решения, принятые на каждом этапе моделирования и разработки базы данных, будут сказываться на дальнейших этапах. Поэтому особую роль играет принятие правильных решений на ранних этапах моделирования.

Часть 2. Описание процесса проектирования на упрощенном примере предметной области «Фильмотека»

Рассмотрим процесс проектирования ИС на примере пункта проката фильмов.

Описание ПО.

Пусть имеется прокатный пункт, в котором имеются копии фильмов. Каждая копия фильма содержит один фильм, хранится на определенном носителе и имеет свой каталожный номер. Копии фильмов даются в прокат зарегистрированным пользователям. Оформление выдачи и возврата копии производится сотрудником пункта. При этом хранятся данные о клиенте, копии, дате выдачи и возврата копии. Ситуации несвоевременного возврата, утраты копии, увольнения сотрудника и пр. рассматривать не будем (с целью упрощения).

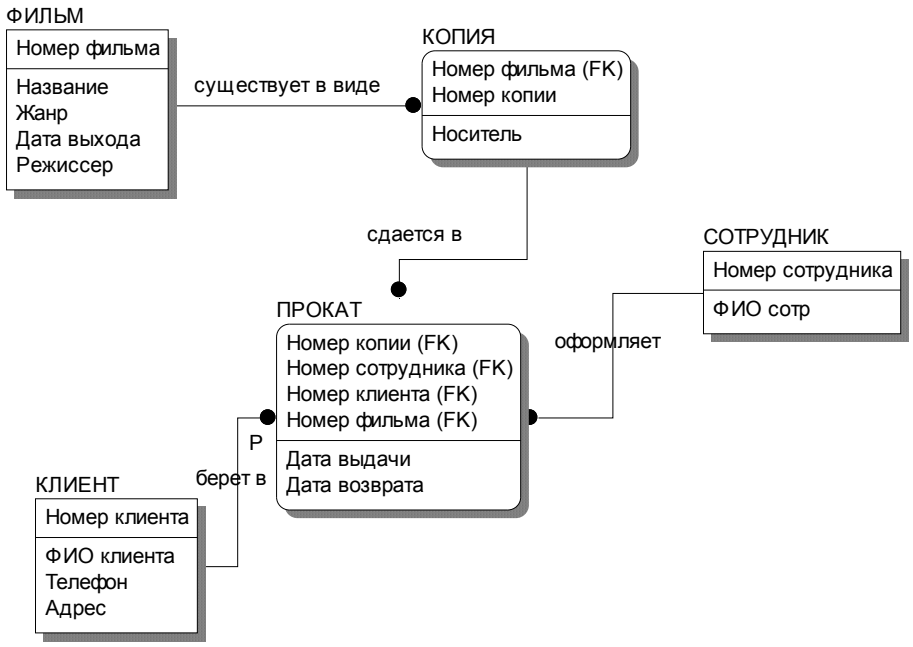
Ограничения ПО

Примечание [M1]: Сформулировать ограничения

1. Копия может и не побывать в прокате некоторое время.
2. Учитываются данные о фильмах, вышедших не ранее 01 января 1970 года.

Логическая модель ПО.

Используя методологию IDEF1x, построим ER-диаграмму. Она отображает взаимосвязи объектов ПО.



На основе ER-модели строится реляционная модель, в которой сущности обычно соответствует отношение, или таблица. Каждому атрибуту

соответствует столбец. Имена таблиц и полей обычно соответствуют именам сущностей и атрибутов, при этом нет привязки к определенной СУБД.

На основе реляционной модели строится физическая модель, которая имеет привязку к конкретной СУБД.

Пояснения :
Прямоугольники – это сущности (зависимые и независимые). Линии – соединения, или связи между сущностями.

Физическая модель ПО.
На основе ER модели строиться физическая модель. Она привязана к конкретной СУБД и содержит, в частности, описание таблиц, полей, индексов.

Часть 3. Модели данных. ER модель и Нотация Чена.

СУЩНОСТИ

Сущность (entity) — это некоторый объект, идентифицируемый в рабочей среде пользователя, нечто такое, за чем пользователь хотел бы наблюдать. Примерами сущностей могут служить СОТРУДНИК Мэри Доу, КЛИЕНТ 12345, ЗАКАЗ 1000, ПРОДАВЕЦ Джон Смит или ПРОДУКТ А4200. Сущности одного и того же типа группируются в *классы сущностей* (entity classes).

Важно уяснить разницу между классом сущностей и экземпляром сущности. *Класс сущностей* — это совокупность сущностей, и описывается он структурой или форматом сущностей, составляющих этот класс. *Экземпляр сущности* (entity instance) представляет конкретную сущность, такую как КЛИЕНТ 12345; он описывается значениями атрибутов данной сущности. Обычно класс сущностей содержит множество экземпляров сущности. Например, класс КЛИЕНТ содержит множество экземпляров — по одному на каждого клиента, для которого имеется запись в базе данных. Пример класса сущностей и двух экземпляров сущности показан на рисунке. Далее в тексте понятия «класс» и «тип» будем рассматривать как равнозначные. Также под понятиями «атрибут» и «сущность» будут подразумеваться тип атрибута и тип сущности.

На ER-диаграммах сущности изображаются в виде прямоугольников с острыми или скругленными углами.

СТУДЕНТ	СТУДЕНТ	СТУДЕНТ
Фамилия	Маслов	Кондратьев
Имя	Александр	Михаил
№Группы	5832	5832
ДР	ДР	ДР
№ телефона	№ телефона	№ телефона

Атрибуты.

У сущностей есть *атрибуты* (attributes), или, как их иногда называют, *свойства* (properties), которые описывают характеристики сущности. Примеры атрибутов – Фамилия, Имя, №Группы, ДР,№ телефона.

В модели «сущность—связь» предполагается, что все экземпляры одного и того же класса сущностей имеют одинаковые атрибуты.

Также различают класс и экземпляр атрибута. Класс – это наименование характеристики, экземпляр – конкретное значение. Например, «Цвет» является типом атрибута, который отражает определенное свойство сущности, а «зеленый», «синий», «красный» - это экземпляры данного атрибута.

Исходное определение модели «сущность—связь» включает в себя композитные атрибуты (composite attributes) и многозначные атрибуты (multi-valued attributes). Композитный, или составной атрибут – это атрибут, состоящий из фиксированного набора других атрибутов. В качестве примера композитного атрибута можно привести атрибут Адрес, состоящий из группы атрибутов {Улица, Город, Штат, Индекс}.

.

Множественный атрибут – это такой атрибут, который для одного и того же экземпляра сущности может принимать множество значений.

Примером многозначного атрибута может служить атрибут Номер телефона для сущности СТУДЕНТ, если студент может иметь несколько телефонов.

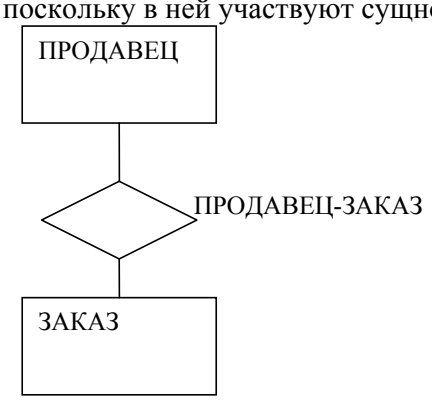
Ключи сущностей.

СВЯЗИ.

Взаимоотношения сущностей выражаются *связями* (relationships). Модель «сущность—связь» включает в себя классы связей и экземпляры связей. *Классы связей* (relationship classes) — это взаимоотношения между классами сущностей, а *экземпляры связи* (relationship instances) — взаимоотношения между экземплярами сущностей. У связей могут быть атрибуты.

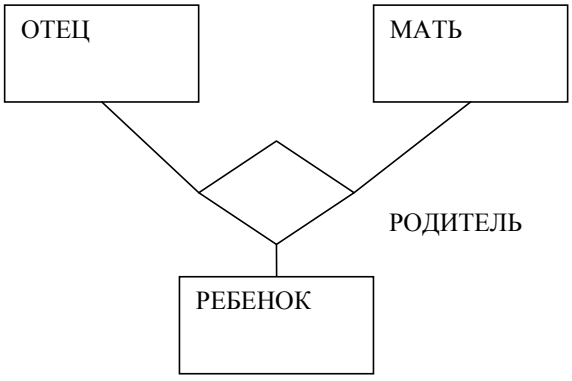
Класс связей может затрагивать несколько классов сущностей. Число классов сущностей, участвующих в связи, называется *степенью связи* (relationship

degree). Изображенная на рисунке связь ПРОДАВЕЦ-ЗАКАЗ имеет степень 2, поскольку в ней участвуют сущности двух классов – ПРОДАВЕЦ и ЗАКАЗ.



Такая связь рассматривается как отображение того факта, что продавец обслужил заказ. На диаграммах связи отображаются ромбами, в которых указывается кардинальность связи. Имя класса связи указывается ниже .

Связь РОДИТЕЛЬ имеется степень 3, т.к. в ней участвуют сущности трех классов – МАТЬ, ОТЕЦ и РЕБЕНОК.
Связи более высоких степеней возможны, но их достаточно трудно интерпретировать.
Связи степени 2 называются бинарными, степени 3 – тренарными.



На рис. 3.3 показаны три типа бинарных связей. В связи 1:1 («один к одному»)

одиночный экземпляр сущности одного типа связан с одиночным экземпляром сущности другого типа. На рис. 3.3, а связь СЛУЖЕБНЫЙ_АВТОМОБИЛЬ связывает одиночную сущность класса СОТРУДНИК с одиночной сущностью класса АВТОМОБИЛЬ. В соответствии с этой диаграммой, ни за одним сотрудником не закреплено более одного автомобиля, и ни один автомобиль не закреплен более чем за одним сотрудником.

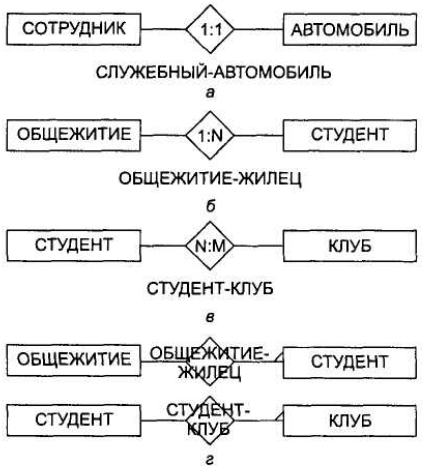


Рис. 3.3. Три типа бинарных связей: *а* — бинарная связь 1:1; *б* — бинарная связь 1:N; *в* — бинарная связь N:M; *г* — представление связи с помощью разветвлений

На рис. 3.3, б изображен второй тип связи, 1:N («один к N» или «один ко многим»). В этой связи, которая называется ОБЩЕЖИТИЕ-ЖИЛЕЦ, единичный экземпляр сущности класса ОБЩЕЖИТИЕ связан со многими экземплярами сущности класса СТУДЕНТ. В соответствии с этим рисунком, в общежитии проживает много студентов, но каждый студент живет только в одном общежитии.

Позиция, в которой стоят 1 и N, имеет значение. Единица стоит на той стороне связи, где располагается ОБЩЕЖИТИЕ, а N стоит на той стороне связи, где располагается СТУДЕНТ. Если бы 1 и N располагались наоборот, и связь записывалась бы как N:l, получилось бы, что в общежитии живет один студент, причем каждый студент живет в нескольких общежитиях. Это, разумеется, не так.

На рис. 3.3, в показан третий тип бинарной связи, N:M (читается «N к M» или «многие ко многим»). Эта связь называется СТУДЕНТ-КЛУБ, и она связывает экземпляры сущностей класса СТУДЕНТ с экземплярами сущностей класса КЛУБ. 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, а совокупность из двух таких ограничений для обеих сторон связи называется максимальной кардинальностью (maximum cardinality) связи. Например, о связи, изображенной на рис. 3.3, 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 БАСКЕТБОЛЬНАЯ_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. 3.3, называются иногда связями типа «ИМЕЕТ», или связями обладания (HAS-A relationships). Такой термин используется потому, что одна сущность имеет (has) связь с другой сущностью. Например: сотрудник имеет автомобиль, студент имеет общежитие, клуб имеет студентов.

Для указания минимальной кардинальности (minimum cardinality) существует несколько способов. Один из них, продемонстрированный на рис. 3.4, 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. 3.4 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.

Для указания минимальной кардинальности (minimum cardinality) существует несколько способов. Один из них, продемонстрированный на рис. 3.4, 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. 3.4 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.

Может существовать связь между сущностями одного и того же класса. Например, для сущностей класса СТУДЕНТ может быть определена связь СОСЕД_ ПО_КОМНАТЕ. Такая связь показана на рис. 3.5, а, а на рис. 3.5, б изображены экземпляры сущностей, охваченных этой связью. Связи между сущностями одного и того же класса называются иногда рекурсивными связями (recursive relationships).

Изображение атрибутов в диаграммах «сущность—связь»

В некоторых версиях ER-диаграмм атрибуты обозначаются эллипсами, соединенными с сущностью или связью, которой они принадлежат. На рис. 3.6, а 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 в ER-диаграмме может сделать ее чересчур громоздкой и трудной для восприятия. В подобных случаях

список атрибутов сущностей дается отдельно, как показано на рис. 3.6, б. Многие CASE-средства показывают такие атрибуты в раскрывающихся окнах.

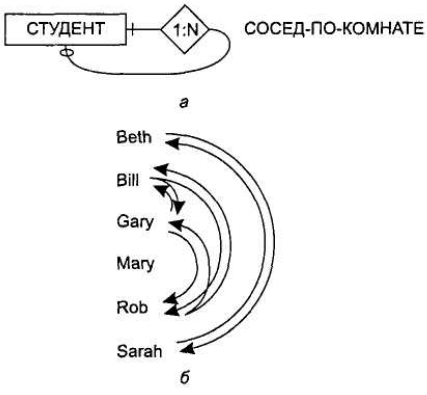


Рис. 3.5. Рекурсивная связь

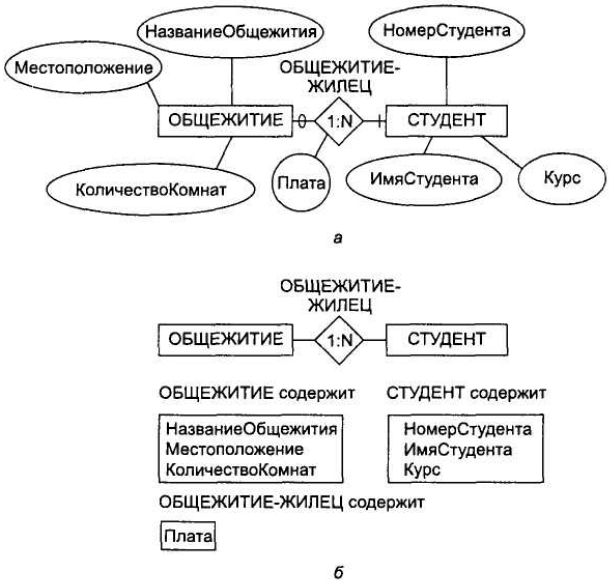


Рис. 3.6. Изображение свойств на диаграммах «сущность—связь»:
а — указание на диаграмме; *б* — отдельное перечисление

Слабые сущности

В модели «сущность—связь» определен особый тип сущности, называемый слабой сущностью (weak entity). К слабым сущностям относятся такие сущности, которые могут существовать в базе данных только в том случае, если в ней присутствует сущность некоторого другого типа. Сущность, не являющаяся слабой, называется сильной сущностью (strong entity).

Чтобы разобраться в том, что такое слабые сущности, рассмотрим базу данных отдела кадров с классами сущностей СОТРУДНИК и ПОДЧИНЕННЫЙ. Предположим, что, в соответствии с деловым регламентом, 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 данные о сущности ПОДЧИНЕННЫЙ могут появиться в базе данных только в том случае, если эта сущность имеет связь с какой-либо сущностью класса СОТРУДНИК.

Рис. 3.7. Слабые сущности: *а* — пример слабой сущности; *б* — пример идентификационно-зависимой сущности

Как показано на рис. 3.7, б, слабые сущности обозначаются прямоугольниками со скругленными углами. Кроме того, связь, от которой зависит существование сущности, обозначается ромбом со скругленными углами. В качестве альтернативы, в некоторых ER-диаграммах (не показано здесь) прямоугольники для слабых сущностей рисуются двойной линией, а связи, от которых зависит существование этих сущностей, изображаются двойными ромбами.

Идентификационно-зависимые сущности

В модели «сущность—связь» имеется особый тип слабых сущностей, называемый идентификационно-зависимыми сущностями (ID-dependent entities). Это такие сущности, идентификаторы которых содержат идентификатор другой сущности. Рассмотрим сущности ДОМ и КВАРТИРА. Пусть идентификатором сущности ДОМ является атрибут НазваниеДома, а идентификатором сущности КВАРТИРА является композитный идентификатор {НазваниеДома, НомерКвартиры}. Поскольку идентификатор сущности КВАРТИРА содержит в себе идентификатор сущности ДОМ (НазваниеДома), то сущность КВАРТИРА является идентификационно-зависимой от сущности ДОМ. Сравните рис. 3.7, б с рис. 3.7, а. По-другому можно представить это так, что, как логически, так и физически, квартира не может существовать, если не существует соответствующего здания.

Идентификационно-зависимые сущности встречаются часто. Еще одним примером может служить сущность ВЕРСИЯ в связи с сущностью ПРОДУКТ, где ПРОДУКТ — это некоторый программный продукт, а ВЕРСИЯ — номер его версии. Идентификатором продукта является атрибут НазваниеПродукта, а идентификатором версии является совокупность {НазваниеПродукта, НомерВерсии}. Третий пример — это сущность ИЗДАНИЕ в связи с сущностью УЧЕБНИК. Идентификатором сущности УЧЕБНИК является атрибут Заглавие, а идентификатором издания является совокупность {Заглавие, ПорядковыйНомерИздания}.