Учебное пособие

ОСНОВЫ ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

ОГЛАВЛЕНИЕ

Глава 1. Что такое базы данных и СУБД

1.1. Данные и ЭВМ

Восприятие реального мира можно соотнести с последовательностью разных, хотя иногда и взаимосвязанных, явлений. С давних времен люди пытались описать эти явления (даже тогда, когда не могли их понять). Такое описание называют *данными*.

Традиционно фиксация данных осуществляется с помощью конкретного средства общения (например, с помощью естественного языка или изображений) на конкретном носителе (например, камне или бумаге). Обычно данные (факты, явления, события, идеи или предметы) и их интерпретация (семантика) фиксируются совместно, так как естественный язык достаточно гибок для представления того и другого. Примером может служить утверждение "Стоимость авиабилета 128". Здесь "128" - данное, а "Стоимость авиабилета" - его семантика.

Нередко данные и интерпретация разделены. Например, "Расписание движения самолетов" может быть представлено в виде таблицы (рис. 1.1), в верхней части которой (отдельно от данных) приводится их интерпретация. Такое разделение затрудняет работу с данными (попробуйте быстро получить сведения из нижней части таблицы).

Интерпретация							
Номер рейса	**Дни недели**	**Пункт отправления**	**Время вылета**	**Пункт назначения**	**Время прибытия**	**Тип самолета**	**Стоимость билета**
Данные							
138	2_4_7	Баку	21.12	Москва	0.52	ИЛ-86	115.00
57	3_6	Ереван	7.20	Киев	9.25	ТУ-154	92.00
1234	2_6	Казань	22.40	Баку	23.50	ТУ-134	73.50
242	1 по 7	Киев	14.10	Москва	16.15	ТУ-154	57.00
86	2_3_5	Минск	10.50	Сочи	13.06	ИЛ-86	78.50
137	1_3_6	Москва	15.17	Баку	18.44	ИЛ-86	115.00
241	1 по 7	Москва	9.05	Киев	11.05	ТУ-154	57.00
577	1_3_5	Рига	21.53	Таллин	22.57	АН-24	21.50
78	3_6	Сочи	18.25	Баку	20.12	ТУ-134	44.00
578	2_4_6	Таллин	6.30	Рига	7.37	АН-24	21.50

Рис. 1.1. К разделению данных и их интерпретации

Применение ЭВМ для ведения[1] и обработки данных обычно приводит к еще большему разделению данных и интерпретации. ЭВМ имеет дело главным образом с данными как таковыми. Большая часть интерпретирующей информации вообще не фиксируется в явной форме (ЭВМ не "знает", является ли "21.50" стоимостью авиабилета или временем вылета). Почему же это произошло?

Существует по крайней мере две исторические причины, по которым применение ЭВМ привело к отделению данных от интерпретации. Во-первых, ЭВМ не обладали достаточными возможностями для обработки текстов на естественном языке - основном языке интерпретации данных. Во-вторых, стоимость памяти ЭВМ была первоначально весьма велика. Память использовалась для хранения самих данных, а интерпретация традиционно возлагалась на пользователя. Пользователь закладывал интерпретацию данных в свою программу, которая "знала", например, что шестое вводимое значение связано с временем прибытия самолета, а четвертое - с временем его вылета. Это существенно повышало роль программы, так как вне интерпретации данные представляют собой не более чем совокупность битов на запоминающем устройстве.

Жесткая зависимость между данными и использующими их программами создает серьезные проблемы в ведении данных и делает использования их менее гибкими.

Нередки случаи, когда пользователи одной и той же ЭВМ создают и используют в своих программах разные наборы данных, содержащие сходную информацию. Иногда это связано с тем, что пользователь не знает (либо не захотел узнать), что в соседней комнате или за соседним столом сидит сотрудник, который уже давно ввел в ЭВМ нужные данные. Чаще потому, что при совместном использовании одних и тех же данных возникает масса проблем.

Разработчики прикладных программ (написанных, например, на Бейсике, Паскале или Си) размещают нужные им данные в файлах, организуя их наиболее удобным для себя образом. При этом одни и те же данные могут иметь в разных приложениях совершенно разную организацию (разную последовательность размещения в записи, разные форматы одних и тех же полей и т.п.). Обобществить такие данные чрезвычайно трудно: например, любое изменение структуры записи файла, производимое одним из разработчиков, приводит к необходимости изменения другими разработчиками тех программ, которые используют записи этого файла.

Для иллюстрации обратимся к примеру, приведенному в книге: У.Девис, Операционные системы, М., Мир, 1980:

"Несколько лет назад почтовое ведомство (из лучших побуждений) пришло к решению, что все адреса должны обязательно включать почтовый индекс. Во многих вычислительных центрах это, казалось бы, незначительное изменение привело к ужасным последствиям. Добавление к адресу нового поля, содержащего шесть символов, означало необходимость внесения изменений в каждую программу, использующую данные этой задачи в соответствии с изменившейся суммарной длиной полей. Тот факт, что какой-то программе для выполнения ее функций не требуется знания почтового индекса, во внимание не принимался: если в некоторой программе содержалось обращение к новой, более длинной записи, то в такую программу вносились изменения, обеспечивающие дополнительное место в памяти.

[1] Ведение (сопровождение, поддержка) данных - термин объединяющий действия по добавлению, удалению или изменению хранимых данных.

В условиях автоматизированного управления централизованной базой данных все такие изменения связаны с функциями управляющей программы базы данных. Программы, не использующие значения почтового индекса, не нуждаются в модификации - в них, как и прежде, в соответствии с запросами посылаются те же элементы данных. В таких случаях внесенное изменение неощутимо. Модифицировать необходимо только те программы, которые пользуются новым элементом данных.".

1.2. Концепция баз данных

Активная деятельность по отысканию приемлемых способов обобществления непрерывно растущего объема информации привела к созданию в начале 60-х годов специальных программных комплексов, называемых "*Системы управления базами данных*" (СУБД).

Основная особенность СУБД - это наличие процедур для ввода и хранения не только самих данных, но и описаний их структуры. Файлы, снабженные описанием хранимых в них данных и находящиеся под управлением СУБД, стали называть банки данных, а затем "*Базы данных*" (БД).

Пусть, например, требуется хранить расписание движения самолетов (рис. 1.1) и ряд других данных, связанных с организацией работы аэропорта (БД "Аэропорт"). Используя для этого одну из современных "русифицированных" СУБД, можно подготовить следующее описание расписания:

```
СОЗДАТЬ ТАБЛИЦУ Расписание
  (Номер_рейса        Целое
   Дни_недели         Текст (8)
   Пункт_отправления  Текст (24)
   Время_вылета       Время
   Пункт_назначения   Текст (24)
   Время_прибытия     Время
   Тип_самолета       Текст (8)
   Стоимость_билета   Валюта);
```

и ввести его вместе с данными в БД "Аэропорт".

Язык запросов СУБД позволяет обращаться за данными как из программ, так и с терминалов (рис. 1.2). Сформировав запрос

```
ВЫБРАТЬ  Номер_рейса, Дни_недели, Время_вылета
ИЗ ТАБЛИЦЫ  Расписание
ГДЕ  Пункт_отправления = 'Москва'
  И  Пункт_назначения  = 'Киев'
  И  Время_вылета > 17;
```

получим расписание "Москва-Киев" на вечернее время, а по запросу

```
ВЫБРАТЬ  КОЛИЧЕСТВО(Номер_рейса)
ИЗ ТАБЛИЦЫ  Расписание
ГДЕ  Пункт_отправления = 'Москва'
  И  Пункт_назначения  = 'Минск';
```

получим количество рейсов "Москва-Минск".

Рис. 1.2. Связь программ и данных при использовании СУБД

Эти запросы не потеряют актуальности и при расширении таблицы:

```
ДОБАВИТЬ В ТАБЛИЦУ Расписание
   Длительность_полета  Целое;
```

как это было с программами обработки почтовых адресов при введении почтового индекса (см. п. 1.1).

Однако, за все надо расплачиваться: на обмен данными через СУБД требуется большее время, чем на обмен аналогичными данными прямо из файлов, специально созданных для того или иного приложения.

1.3. Архитектура СУБД

СУБД должна предоставлять доступ к данным любым пользователям, включая и тех, которые практически не имеют и (или) не хотят иметь представления о:

- физическом размещении в памяти данных и их описаний;
- механизмах поиска запрашиваемых данных;
- проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);
- способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;
- поддержании баз данных в актуальном состоянии

и множестве других функций СУБД.

При выполнении основных из этих функций СУБД должна использовать различные описания данных. А как создавать эти описания?

Естественно, что проект базы данных надо начинать с анализа предметной области и выявления требований к ней отдельных пользователей (сотрудников организации, для кото-

рых создается база данных). Подробнее этот процесс будет рассмотрен ниже, а здесь отметим, что проектирование обычно поручается человеку (группе лиц) - *администратору базы данных* (АБД). Им может быть как специально выделенный сотрудник организации, так и будущий пользователь базы данных, достаточно хорошо знакомый с машинной обработкой данных.

Объединяя частные представления о содержимом базы данных, полученные в результате опроса пользователей, и свои представления о данных, которые могут потребоваться в будущих приложениях, 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 данных* (рис. 1.3).

Рис. 1.3. Уровни моделей данных

Такая человеко-ориентированная модель полностью независима от физических параметров среды хранения данных. В конце концов этой средой может быть память человека, а не ЭВМ. Поэтому инфологическая модель не должна изменяться до тех пор, пока какие-то изменения в реальном мире не потребуют изменения в ней некоторого определения, чтобы эта модель продолжала отражать предметную область.

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 данных*.

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 данных*.

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.

1.4. Модели данных

Как отмечалось в п. 1.3, инфологическая модель отображает реальный мир в некоторые понятные человеку концепции, полностью независимые от параметров среды хранения данных. Существует множество подходов к построению таких моделей: графовые модели, семантические сети, модель "сущность-связь" и т.д. [11]. Наиболее популярной из них оказалась модель "сущность-связь", которая будет рассмотрена в главе 2.

Инфологическая модель должна быть отображена в компьютеро-ориентированную даталогическую модель, "понятную" СУБД. В процессе развития теории и практического использования баз данных, а также средств вычислительной техники создавались СУБД, поддерживающие различные даталогические модели [1, 2, 8, 11].

Сначала стали использовать иерархические даталогические модели. Простота организации, наличие заранее заданных связей между сущностями, сходство с физическими моделями данных позволяли добиваться приемлемой производительности иерархических СУБД на медленных ЭВМ с весьма ограниченными объемами памяти. Но, если данные не имели древовидной структуры, то возникала масса сложностей при построении иерархической модели и желании добиться нужной производительности.

Сетевые модели также создавались для мало ресурсных ЭВМ. Это достаточно сложные структуры, состоящие из "наборов" - поименованных двухуровневых деревьев. "Наборы" соединяются с помощью "записей-связок", образуя цепочки и т.д. При разработке сетевых моделей было выдумано множество "маленьких хитростей", позволяющих увеличить производительность СУБД, но существенно усложнивших последние. Прикладной программист должен знать массу терминов, изучить несколько внутренних языков СУБД, детально представлять логическую структуру базы данных для осуществления навигации среди различных экземпляров, наборов, записей и т.п. Один из разработчиков операционной системы UNIX сказал "Сетевая база - это самый верный способ потерять данные".

Сложность практического использования иерархических и и сетевых СУБД заставляла искать иные способы представления данных. В конце 60-х годов появились СУБД на основе инвертированных файлов, отличающиеся простотой организации и наличием весьма удобных языков манипулирования данными. Однако такие СУБД обладают рядом ограничений на количество файлов для хранения данных, количество связей между ними, длину записи и количество ее полей.

Сегодня наиболее распространены реляционные модели, которые будут подробно рассмотрены в главе 3.

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 подстройки модели под конкретную БД. Разнообразие способов корректировки физических моделей современных промышленных СУБД не позволяет рассмотреть их в этом разделе.

Глава 2. Инфологическая модель данных "Сущность-связь"

2.1. Основные понятия

Цель инфологического моделирования - 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). Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Сущность - любой различимый объект (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как *тип сущности* и *экземпляр сущности*. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к конкретной вещи в наборе. Например, типом сущности может быть ГОРОД, а экземпляром - Москва, Киев и т.д.

Атрибут - поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей (например, ЦВЕТ может быть определен для многих сущностей: СОБАКА, АВТОМОБИЛЬ, ДЫМ и т.д.). Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Примерами атрибутов для сущности АВТОМОБИЛЬ являются ТИП, МАРКА, НОМЕРНОЙ ЗНАК, ЦВЕТ и т.д. Здесь также существует различие между типом и экземпляром. Тип атрибута ЦВЕТ имеет много экземпляров или значений:

Красный, Синий, Банановый, Белая ночь и т.д.,

однако каждому экземпляру сущности присваивается только одно значение атрибута.

Абсолютное различие между типами сущностей и атрибутами отсутствует. Атрибут является таковым только в связи с типом сущности. В другом контексте атрибут может выступать как самостоятельная сущность. Например, для автомобильного завода цвет - это только атрибут продукта производства, а для лакокрасочной фабрики цвет - тип сущности.

Ключ - минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Для сущности Расписание (п. 1.2) ключом является атрибут Номер_рейса или набор: Пункт_отправления, Время_вылета и Пункт_назначения (при условии, что из пункта в пункт вылетает в каждый момент времени один самолет).

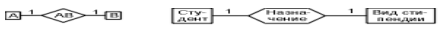
Связь - ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных - это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

2.2. Характеристика связей и язык моделирования

При построении инфологических моделей можно использовать язык *ER-диаграмм* (от англ. Entity-Relationship, т.е. сущность-связь). В них сущности изображаются помеченными прямоугольниками, ассоциации - помеченными ромбами или шестиугольниками, атрибуты - помеченными овалами, а связи между ними - ненаправленными ребрами, над которыми может проставляться степень связи (1 или буква, заменяющая слово "много") и необходимое пояснение.

Между двумя сущностям, например, A и B возможны четыре вида связей.

Первый тип - связь ОДИН-К-ОДНОМУ (1:1): в каждый момент времени каждому представителю (экземпляру) сущности A соответствует 1 или 0 представителей сущности B:



Студент может не "заработать" стипендию, получить обычную или одну из повышенных стипендий.

Второй тип - связь ОДИН-КО-МНОГИМ (1:M): одному представителю сущности A соответствуют 0, 1 или несколько представителей сущности B.

Квартира может пустовать, в ней может жить один или несколько жильцов.

Так как между двумя сущностями возможны связи в обоих направлениях, то существует еще два типа связи МНОГИЕ-К-ОДНОМУ (M:1) и МНОГИЕ-КО-МНОГИМ (M:N).

Пример 2.1. Если связь между сущностями МУЖЧИНЫ и ЖЕНЩИНЫ называется БРАК, то существует четыре возможныхпредставления такой связи:

Характер связей между сущностями не ограничивается перечисленными. Существуют и более сложные связи:

- множество связей между одними и теми же сущностями

(пациент, имея одного лечащего врача, может иметь также несколько врачей-консультантов; врач может быть лечащим врачом нескольких пациентов и может одновременно консультировать несколько других пациентов);

- тернарные связи

(врач может назначить несколько пациентов на несколько анализов, анализ может быть назначен несколькими врачами нескольким пациентам и пациент может быть назначен на несколько анализов несколькими врачами);

- связи более высоких порядков, семантика (смысл) которых иногда очень сложна.

В приведенных примерах для повышения иллюстративности рассматриваемых связей не показаны атрибуты сущностей и ассоциаций во всех ER-диаграммах. Так, ввод лишь нескольких основных атрибутов в описание брачных связей значительно усложнит ER-диаграмму (рис. 2.1,а). В связи с этим язык ER-диаграмм используется для построении небольших моделей и иллюстрации отдельных фрагментов больших. Чаще же применяется менее наглядный, но более содержательный *язык инфологического моделирования* (ЯИМ), в котором сущности и ассоциации представляются предложениями вида:

```
СУЩНОСТЬ (атрибут 1, атрибут 2 , ..., атрибут n)
АССОЦИАЦИЯ [СУЩНОСТЬ S1, СУЩНОСТЬ S2, ...]
           (атрибут 1, атрибут 2, ..., атрибут n)
```

где S - степень связи, а атрибуты, входящие в ключ, должны быть отмечены с помощью подчеркивания.

Так, рассмотренный выше пример множества связей между сущностями, может быть описан на ЯИМ следующим образом:

```
Врач (Номер_врача, Фамилия, Имя, Отчество, Специальность)
Пациент (Регистрационный_номер, Номер койки, Фамилия,
         Имя, Отчество, Адрес, Дата рождения, Пол)
Лечащий_врач [Врач 1, Пациент M]
              (Номер_врача, Регистрационный_номер)
Консультант [Врач M,Пациент N]
             (Номер_врача, Регистрационный_номер).
```

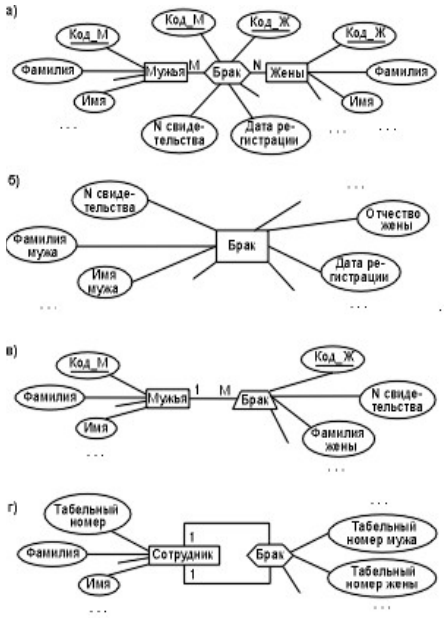


Рис. 2.1. Примеры ER-диаграмм

Для выявления связей между сущностями необходимо, как минимум, определить сами сущности. Но это не простая задача, так как в разных предметных областях один и тот же объект может быть сущностью, атрибутом или ассоциацией. Проиллюстрируем такое утверждение на примерах, связанных с описанием брачных связей (см. пример 2.1).

Пример 2.2. Отдел записей актов гражданского состояния (ЗАГС) имеет дело не со всеми людьми, а только с теми, кто обратился с просьбой о регистрации брака, рождения или смерти. Поэтому в странах, где допускаются лишь традиционные браки, отделы ЗАГС могут размещать сведения о регистрируемых браках в единственной сущности:

```
Брак (Номер_свидетельства, Фамилия_мужа, Имя_мужа,
      Отчество_мужа, Дата_рождения_мужа, Фамилия_жены,
      ... , Дата_регистрации, Место_регистрации, ...),
```

ER-диаграмма которой приведена на рис. 2.1,б.

Пример 2.3. Теперь рассмотрим ситуацию, когда отдел ЗАГС расположен в стране, допускающей многоженство. Если для регистрации браков использовать сущность "Брак" примера 2.2, то будут дублироваться сведения о мужьях, имеющих несколько жен (см. табл. 2.1).

Таблица 2.1

Номер свидетельства	**Фамилия мужа**	**...**	**Фамилия жены**	**...**	**Дата регистрации**
1-ЮБ 154745	Петухов	...	Курочкина	...	06/03/1991
1-ЮБ 163489	Петухов	...	Пеструшкина	...	11/08/1991
1-ЮБ 169887	Петухов	...	Рябова	...	12/12/1992
1-ЮБ 169878	Селезнев	...	Уточкина	...	12/12/1992
1-ЮБ 154746	Парасюк	...	Свинюшкина	...	06/03/1991

1-ЮБ 169879	Парасюк	...	Хаврония	...	12/12/1992
...	...	...	...	...	...

Дублирование можно исключить созданием дополнительной сущности "Мужья"

```
Мужья (Код_М, Фамилия, Имя, Отчество, Дата рождения, Место рождения)
```

и заменой сущности "Брак" характеристикой (см. п. 2.3) со ссылкой на соответствующее описание в сущности "Мужья".

```
Брак (Номер свидетельства, Код_М, Фамилия жены, ...,
      Дата регистрации, ...){Мужья}.
```

ER-диаграмма связи этих сущностей показана на рис. 2.1,в, а пример их экземпляров в табл. 2.2 и 2.3.

Таблица 2.2

Код_М	**Фамилия**	**Имя**	**Отчество**	**Год/р.**	**Место рожд.**
111	Петухов	Альфред	Остапович	1971	г. Цапелька
112	Селезнев	Вавила	Абрамович	1973	г. Гусев
113	Парасюк	Гораций	Федулович	1972	г. Свиньин
...	...	...	...	...	...

Таблица 2.3

Номер свидетельства	**Код_М**	**Фамилия жены**	**Имя жены**	**Дата регистрации**	**...**
1-ЮБ 154745	111	Курочкина	Августина	06/03/1991	...
1-ЮБ 163489	111	Пеструшкина	Мариана	11/08/1991	...
1-ЮБ 169877	111	Рябова	Милана	12/12/1992	...
1-ЮБ 169878	112	Уточкина	Вероника	12/12/1992	...
1-ЮБ 154746	113	Свинюшкина	Эльвира	06/03/1991	...
1_ЮБ 169879	113	Хаврония	Руфина	12/12/1992	...
...	...	...	...	...	...

Пример 2.4. Наконец, рассмотрим случай, когда какой-либо организации потребовались данные о наличии в ней семейных пар, а для хранения сведений о сотрудниках уже имеется сущность

```
Сотрудники (Табельный_номер, Фамилия, Имя, ...).
```

Использование, рассмотренной в примере 2.2, сущности "Брак" нецелесообразно: в "Сотрудники" уже есть фамилии, имена, отчества супругов. Поэтому создадим ассоциацию

```
Брак [Сотрудник 1, Сотрудник 1]
     (Табельный_номер_мужа, Табельный_номер_жены, ...),
```

связывающую между собой определенные экземпляры сущности "Сотрудники" (рис. 2.1,г).

В заключение отметим, что ER-диаграмма рис. 2.1,а описывает структуру размещения данных о браках в отделах ЗАГС стран, допускающих групповые браки, а ER-диаграммы примера 2.1, описания любых видов браков в организациях, где есть сущности "мужчины" и "женщины", включающие холостых и незамужних.

Что же такое "связь"? В ER-диаграммах это линия, соединяющая геометрические фигуры, изображающие сущности, атрибуты, ассоциации и другие информационные объекты. В тексте же этот термин используется для указания на взаимозависимость сущностей. Если эта взаимозависимость имеет атрибуты, то она называется ассоциацией.

2.3. Классификация сущностей

Настал момент разобраться в терминологии. К.Дейт [3] определяет три основные класса сущностей: *стержневые*, *ассоциативные* и *характеристические*, а также подкласс ассоциативных сущностей - *обозначения*.

Стержневая сущность (*стержень*) - это независимая сущность (несколько подробнее она будет определена ниже).

В рассмотренных ранее примерах стержни - это "Студент", "Квартира", "Мужчины", "Врач", "Брак" (из примера 2.2) и другие, названия которых помещены в прямоугольники.

Ассоциативная сущность (*ассоциация*) - это связь вида "многие-ко-многим" ("-ко-многим" и т.д.) между двумя или более сущностями или экземплярами сущности (как в примере 2.4). Ассоциации рассматриваются как полноправные сущности:

они могут участвовать в других ассоциациях и обозначениях точно так же, как стержневые сущности;

могут обладать свойствами, т.е. иметь не только набор ключевых атрибутов, необходимых для указания связей, но и любое число других атрибутов, характеризующих связь. Например, ассоциации "Брак" из примеров 2.1 и 2.4 содержат ключевые атрибуты "Код_М", "Код_Ж" и "Табельный номер мужа", "Табельный номер жены", а также уточняющие атрибуты "Номер свидетельства", "Дата регистрации", "Место_регистрации", "Номер записи в книгу ЗАГС" и т.д.

Характеристическая сущность (*характеристика*) - это связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Единственная цель характеристики в рамках рассматриваемой предметной области состоит в описании или уточнении некоторой другой сущности. Необходимость в них возникает в связи с тем, что сущности реального мира имеют иногда многозначные свойства. Муж может иметь несколько жен (пример 2.3), книга - несколько характеристик переиздания (исправленное, дополненное, переработанное, ...) и т.д.

Существование характеристики полностью зависит от характеризуемой сущности: женщины лишаются статуса жен, если умирает их муж.

Для описания характеристики используется новое предложение ЯИМ, имеющее в общем случае вид:

```
ХАРАКТЕРИСТИКА (атрибут 1, атрибут 2, ...)
               {СПИСОК ХАРАКТЕРИЗУЕМЫХ СУЩНОСТЕЙ}.
```

Расширим также язык ER-диаграмм, введя для изображения характеристики трапецию (рис. 2.2).

Рис. 2.2. Элементы расширенного языка ER-диаграмм

Обозначающая сущность или *обозначение* - это связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями и отличается от характеристики тем, что не зависит от обозначаемой сущности.

Рассмотрим пример, связанный с зачислением сотрудников в различные отделы организации.

При отсутствии жестких правил (сотрудник может одновременно зачисляться в несколько отделов или не зачисляться ни в один отдел) необходимо создать описание с ассоциацией Зачисление:

```
Отделы (Номер отдела, Название отдела, ...)
Служащие (Табельный номер, Фамилия, ...)
Зачисление [Отделы M, Служащие N]
           (Номер отдела, Табельный номер, Дата зачисления).
```

Однако, при условии, что каждый из сотрудников должен быть обязательно зачислен в один из отделов, можно создать описание с обозначением Служащие:

```
Отделы (Номер отдела, Название отдела, ...)
Служащие (Табельный номер, Фамилия, ... , Номер отдела,
          Дата зачисления)[Отделы]
```

В данном примере служащие имеют независимое существование (если удаляется отдел, то из этого не следует, что также должны быть удалены служащие такого отдела). Поэтому они не могут быть характеристиками отделов и названы обозначениями.

Обозначения используют для хранения повторяющихся значений больших текстовых атрибутов: "кодификаторы" изучаемых студентами дисциплин, наименований организаций и их отделов, перечней товаров и т.п.

Описание обозначения внешне отличается от описания характеристики только тем, что обозначаемые сущности заключается не в фигурные скобки, а в квадратные:

```
ОБОЗНАЧЕНИЕ (атрибут 1, атрибут 2, ...)[СПИСОК
             ОБОЗНАЧАЕМЫХ СУЩНОСТЕЙ].
```

Как правило, обозначения не рассматриваются как полноправные сущности, хотя это не привело бы к какой-либо ошибке.

Обозначения и характеристики не являются полностью независимыми сущностями, поскольку они предполагают наличие некоторой другой сущности, которая будет "обозначаться" или "характеризоваться". Однако они все же представляют собой частные случаи сущности и могут, конечно, иметь свойства, могут участвовать в ассоциациях, обозначениях и иметь свои собственные (более низкого уровня) характеристики. Подчеркнем также, что все экземпляры характеристики должны быть обязательно связаны с каким-либо экземпляром характеризуемой сущности. Однако допускается, чтобы некоторые экземпляры характеризуемой сущности не имели связей. Правда, если это касается браков, то сущность "Мужья" должна быть заменена на сущность "Мужчины" (нет мужа без жены).

Переопределим теперь стержневую сущность как сущность, которая не является ни ассоциацией, ни обозначением, ни характеристикой. Такие сущности имеют независимое существование, хотя они и могут обозначать другие сущности, как, например, сотрудники обозначают отделы.

В заключение рассмотрим пример построения инфологической модели базы данных "Питание", где должна храниться информация о блюдах (рис. 2.3), их ежедневном потреблении, продуктах, из которых приготавливаются эти блюда, и поставщиках этих продуктов. Информация будет использоваться поваром и руководителем небольшого предприятия общественного питания, а также его посетителями.

1. Лобио по грузински
Ломаную очищенную фасоль, нашинкованный лук посолить, посыпать перцем и припустить в масле с небольшим количеством бульона; добавить кинзу, зелень петрушки, рейган (базилик) и довести до готовности. Затем запечь в духовке.
Фасоль стручковая (свежая или консервированная) 200,
Лук зеленый 40, Масло сливочное 30, Зелень 10.
Выход 210. Калорий 725.

Рис. 2.3. Пример кулинарного рецепта

С помощью указанных пользователей выделены следующие объекты и характеристики проектируемой базы:

1. Блюда, для описания которых нужны данные, входящие в их кулинарные рецепты: номер блюда (например, из книги кулинарных рецептов), название блюда, вид блюда (закуска, суп, горячее и т.п.), рецепт (технология приготовления блюда), выход (вес порции), название, калорийность и вес каждого продукта, входящего в блюдо.
2. Для каждого поставщика продуктов: наименование, адрес, название поставляемого продукта, дата поставки и цена на момент поставки.
3. Ежедневное потребление блюд (расход): блюдо, количество порций, дата.

Анализ объектов позволяет выделить:

- стержни Блюда, Продукты и Города;
- ассоциации Состав (связывает Блюда с Продуктами) и

Поставки (связывает Поставщиков с Продуктами);
- обозначение Поставщики;
- характеристики Рецепты и Расход.

ER-диаграмма модели показана на рис. 2.4. а модель на языке ЯИМ имеет следующий вид:

```
Блюда (БЛ, Блюдо, Вид)
Продукты (ПР, Продукт, Калорийность)
Поставщики (ПОС, Город, Поставщик) [Город]
Состав [Блюда M, Продукты N] (БЛ, ПР, Вес (г))
Поставки [Поставщики M, Продукты N] (ПОС, ПР, Дата_П, Цена, Вес (кг))
Города (Город, Страна)
Рецепты (БЛ, Рецепт) {Блюда}
Расход (БЛ, Дата_Р, Порций) {Блюда}
```

В этих моделях Блюдо, Продукт и Поставщик - наименования, а БЛ, ПР и ПОС - цифровые коды блюд, продуктов и организаций, поставляющих эти продукты.

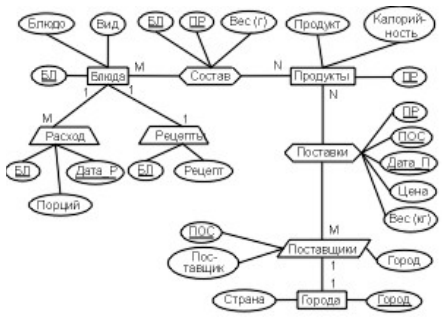


Рис. 2.4. Инфологическая модель базы данных "Питание"

2.4. О первичных и внешних ключах

Напомним, что *ключ* или *возможный ключ* - это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность обладает хотя бы одним возможным ключом. Один из них принимается за *первичный ключ*. При выборе первичного ключа следует отдавать предпочтение несоставным ключам или ключам, составленным из минимального числа атрибутов. Нецелесообразно также использовать ключи с длинными текстовыми значениями (предпочтительнее использовать целочисленные атрибуты). Так, для идентификации студента можно использовать либо уникальный номер зачетной книжки, либо набор из фамилии, имени, отчества, номера группы и может быть дополнительных атрибутов, так как не исключено появление в группе двух студентов (а чаще студенток) с одинаковыми фамилиями, именами и отчествами. Плохо также использовать в качестве ключа не номер блюда, а его название, например, " Закуска из плавленых сырков "Дружба" с ветчиной и соленым огурцом" или "Заяц в сметане с картофельными крокетами и салатом из красной капусты".

Не допускается, чтобы первичный ключ стержневой сущности (любой атрибут, участвующий в первичном ключе) принимал неопределенное значение. Иначе возникнет противоречивая ситуация: появится не обладающий индивидуальностью, и, следовательно не существующий экземпляр стержневой сущности. По тем же причинам необходимо обеспечить уникальность *первичного ключа*.

Теперь о внешних ключах:

- Если сущность C связывает сущности A и B, то она должна включать внешние ключи, соответствующие первичным ключам сущностей A и B.
- Если сущность B обозначает сущность A, то она должна включать внешний ключ, соответствующий первичному ключу сущности A.

В п. 2.3 рассматривался пример, где "Служащие" обозначали "Отделы" и включали внешний ключ "Номер отдела", соответствующий первичному ключу сущности "Отделы".

Связь между первичными и внешними ключами сущностей иллюстрируется рис. 2.5.

Рис. 2.5. Структуры: а - ассоциации; б - обозначения (характеристики)

Здесь для обозначения любой из ассоциируемых сущностей (стержней, характеристик, обозначений или даже ассоциаций) используется новый обобщающий термин "Цель" или "Целевая сущность".

Таким образом, при рассмотрении проблемы выбора способа представления ассоциаций и обозначений в базе данных основной вопрос, на который следует получить ответ: "Каковы внешние ключи?". И далее, для каждого внешнего ключа необходимо решить три вопроса:

1. Может ли данный внешний ключ принимать неопределенные значения (NULL-значения)? Иначе говоря, может ли существовать некоторый экземпляр сущности данного типа, для которого неизвестна целевая сущность, указываемая внешним ключом? В случае поставок это, вероятно, невозможно - поставка, осуществляемая неизвестным поставщиком, или поставка неизвестного продукта не имеют смысла. Но в случае с сотрудниками такая ситуация однако могла бы иметь смысл - вполне возможно, что какой-либо сотрудник в данный момент не зачислен вообще ни в какой отдел. Заметим, что ответ на данный вопрос не зависит от прихоти проектировщика базы данных, а определяется фактическим образом действий, принятым в той части реального мира, которая должна быть представлена в рассматриваемой базе данных. Подобные замечания имеют отношение и к вопросам, обсуждаемым ниже.

2. Что должно случиться при попытке УДАЛЕНИЯ целевой сущности, на которую ссылается внешний ключ? Например, при удалении поставщика, который осуществил по крайней мере одну поставку. Существует три возможности:

КАСКАДИРУЕТСЯ	Операция удаления "каскадируется" с тем, чтобы удалить также поставки этого поставщика.
ОГРАНИЧИВАЕТСЯ	Удаляются лишь те поставщики, которые еще не осуществляли поставок. Иначе операция удаления отвергается.

УСТАНАВЛИВАЕТСЯ Для всех поставок удаляемого поставщика NULL-значение внешний ключ устанавливается в неопределенное значение, а затем этот поставщик удаляется. Такая возможность, конечно, неприменима, если данный внешний ключ не должен содержать NULL-значений.

3. Что должно происходить при попытке ОБНОВЛЕНИЯ первичного ключа целевой сущности, на которую ссылается некоторый внешний ключ? Например, может быть предпринята попытка обновить номер такого поставщика, для которого имеется по крайней мере одна соответствующая поставка. Этот случай для определенности снова рассмотрим подробнее. Имеются те же три возможности, как и при удалении:

КАСКАДИРУЕТСЯ Операция обновления "каскадируется" с тем, чтобы обновить также и внешний ключ впоставках этого поставщика.

ОГРАНИЧИВАЕТСЯ Обновляются первичные ключи лишь тех поставщиков, которые еще не осуществляли поставок. Иначе операция обновления отвергается.

УСТАНАВЛИВАЕТСЯ Для всех поставок такого поставщика NULL-значение внешний ключ устанавливается в неопределенное значение, а затем обновляется первичный ключ поставщика. Такая возможность, конечно, неприменима, если данный внешний ключ не должен содержать NULL-значений.

Таким образом, для каждого внешнего ключа в проекте проектировщик базы данных должен специфицировать не только поле или комбинацию полей, составляющих этот внешний ключ, и целевую таблицу, которая идентифицируется этим ключом, но также и ответы на указанные выше вопроса (три ограничения, которые относятся к этому внешнему ключу).

Наконец, о характеристиках - обозначающих сущностях, существование которых зависит от типа обозначаемых сущностей. Обозначение представляется внешним ключом в таблице, соответствующей этой характеристике. Но три рассмотренные выше ограничения на внешний ключ для данного случая должны специфицироваться следующим образом:

```
NULL-значения не допустимы
УДАЛЕНИЕ ИЗ (цель) КАСКАДИРУЕТСЯ
ОБНОВЛЕНИЕ (первичный ключ цели) КАСКАДИРУЕТСЯ
```

Указанные спецификации представляют зависимость по существованию характеристических сущностей.

2.5. Ограничения целостности

Целостность (от англ. integrity - нетронутость, неприкосновенность, сохранность, целостность) - понимается как правильность данных в любой момент времени. Но эта цель может быть достигнута лишь в определенных пределах: СУБД не может контролировать правильность каждого отдельного значения, вводимого в базу данных (хотя каждое значение можно проверить на правдоподобность). Например, нельзя обнаружить, что вводимое значение 5 (представляющее номер дня недели) в действительности должно быть равно 3. С другой стороны, значение 9 явно будет ошибочным и СУБД должна его отвергнуть. Однако для этого ей следует сообщить, что номера дней недели должны принадлежать набору (1,2,3,4,5,6,7).

Поддержание целостности базы данных может рассматриваться как защита данных от неверных изменений или разрушений (не путать с незаконными изменениями и разрушениями, являющимися проблемой безопасности). Современные СУБД имеют ряд средств для обеспечения поддержания целостности (так же, как и средств обеспечения поддержания безопасности).

Выделяют три группы правил целостности:

1. Целостность по сущностям.
2. Целостность по ссылкам.
3. Целостность, определяемая пользователем.

В п. 2.4 была рассмотрена мотивировка двух правил целостности, общих для любых реляционных баз данных.

1. Не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение.
2. Значение внешнего ключа должно либо:
    1. быть равным значению первичного ключа цели;
    2. быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.
3. Для любой конкретной базы данных существует ряд дополнительных специфических правил, которые относятся к ней одной и определяются разработчиком. Чаще всего контролируется:

уникальность тех или иных атрибутов,
диапазон значений (экзаменационная оценка от 2 до 5),
принадлежность набору значений (пол "М" или "Ж").

2.6. О построении инфологической модели

Читатель, познакомившийся лишь с материалом данной и предшествующей глав, не сможет правильно воспринять и оценить тех советов и рекомендаций по построению хорошей инфологической модели, которые десятилетиями формировались крупнейшими специалистами в области обработки данных. Для этого надо, по крайней мере, изучить последующие материалы. В идеале же необходимо, чтобы читатель предварительно реализовал хотя бы один проект информационной системы, предложил его реальным пользователям и побыл администратором базы данных и приложений столь долго, чтобы осознать хотя бы небольшую толику проблем, возникающих из-за недостаточно продуманного проекта. Опыт автора и всех знакомых ему специалистов по информационным системам показывает, что любые теоретические рекомендации воспринимаются всерьез лишь после нескольких безрезультатных попыток оживления неудачно спроектированных систем. (Хотя есть и такие проектировщики, которые продолжают верить, что смогут реанимировать умирающий проект с помощью изменения программ, а не инфологической модели базы данных.)

Основная сложность восприятия рекомендаций, приведенных в четвертой главе и приложении Б, чисто психологического плана.

Действительно, для определения перечня и структуры хранимых данных надо собрать информацию о реальных и потенциальных приложениях, а также о пользователях базы данных, а при построении инфологической модели следует заботиться лишь о надежности

хранения этих данных, напрочь забывая о приложениях и пользователях, для которых создается база данных.

Это связано с абсолютно различающимися требованиями к базе данных прикладных программистов и администратора базы данных. Первые хотели бы иметь в одном месте (например, в одной таблице) все данные, необходимые им для реализации запроса из прикладной программы или с терминала. Вторые же заботятся о исключении возможных искажений хранимых данных при вводе в базу данных новой информации и обновлении или удалении существующей. Для этого они удаляют из базы данных дубликаты и нежелательные функциональные связи между атрибутами, разбивая базу данных на множество маленьких таблиц (см. п. 4.6). Так как многолетний мировой опыт использования информационных систем, построенных на основе баз данных, показывает, что недостатки проекта невозможно устранить любыми ухищрениями в программах приложений, то опытные проектировщики не позволяют себе идти навстречу прикладным программистам (даже тогда, когда они сами являются таковыми).

И хотя автор осознает, что большинство людей предпочитает учиться на собственных ошибках, он все же еще раз советует неопытным проектировщикам баз данных:

- четко разграничивать такие понятия как запрос на данные и ведение данных (ввод, изменение и удаление);
- помнить, что, как правило, база данных является информационной основой не одного, а нескольких приложений, часть их которых появится в будущем;
- плохой проект базы данных не может быть исправлен с помощью любых (даже самых изощренных) приложений.

Глава 3. Реляционный подход

3.1. Реляционная структура данных

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных, т.е. возможности использования привычных и естественных способов представления данных. Наиболее значительной из них была статья сотрудника фирмы IBM д-ра Э.Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), где, вероятно, впервые был применен термин "реляционная модель данных".

Будучи математиком по образованию Э.Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как *отношение* - relation (англ.) [3, 7, 9].

Наименьшая единица данных реляционной модели - это отдельное *атомарное* (неразложимое) для данной модели значение данных. Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой - как три различных значения.

Доменом называется множество атомарных значений одного и того же типа. Так, на рис. 1.1 домен пунктов отправления (назначения) - множество названий населенных пунктов, а домен номеров рейса - множество целых положительных чисел.

Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос "Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву"). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета?

Отношение на доменах D1, D2, ..., Dn (не обязательно, чтобы все они были различны) состоит из заголовка и тела. На рис. 3.1 приведен пример отношения для расписания движения самолетов (рис. 1.1).

Заголовок (на рис. 1.1 он назывался интерпретацией) состоит из такого фиксированного множества атрибутов A1, A2, ..., An, что существует взаимно однозначное соответствие между этими атрибутами Ai и определяющими их доменами Di (i=1,2,...,n).

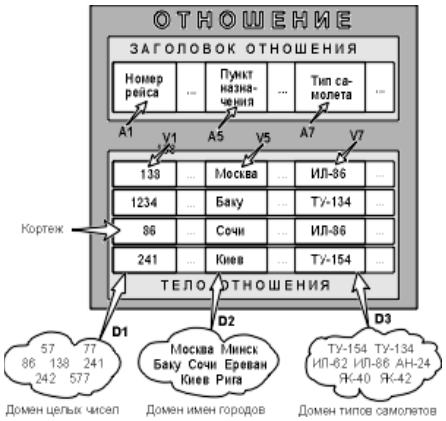


Рис. 3.1. Отношение с математической точки зрения (Ai - атрибуты, Vi - значения атрибутов)

Тело состоит из меняющегося во времени множества *кортежей*, где каждый кортеж состоит в свою очередь из множества пар атрибут-значение (Ai:Vi), (i=1,2,...,n), по одной такой паре для каждого атрибута Ai в заголовке. Для любой заданной пары атрибут-значение (Ai:Vi) Vi является значением из единственного домена Di, который связан с атрибутом Ai.

Степень отношения - это число его атрибутов. Отношение степени один называют унарным, степени два - бинарным, степени три - тернарным, ..., а степени n - n-арным. Степень отношения "Рейс" (рис. 1.1) - 8.

Кардинальное число или *мощность отношения* - это число его кортежей. Мощность отношения "Рейс" равна 10. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение - это множество, а множества по определению не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно-заданный момент времени. Пусть R - отношение с атрибутами A1, A2, ..., An. Говорят, что множество атрибутов K=(Ai, Aj, ..., Ak) отношения R является возможным ключом R тогда и только тогда, когда удовлетворяются два независимых от времени условия:

1. Уникальность: в произвольный заданный момент времени никакие два различных кортежа R не имеют одного и того же значения для Ai, Aj, ..., Ak.
2. Минимальность: ни один из атрибутов Ai, Aj, ..., Ak не может быть исключен из K без нарушения уникальности.

Каждое отношение обладает хотя бы одним возможным ключом, поскольку по меньшей мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Остальные возможные ключи, если они есть, называются альтернативными ключами.

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных. Однако для массового пользователя реляционных СУБД можно с успехом использовать неформальные эквиваленты этих понятий:

Отношение - Таблица (иногда Файл),
Кортеж - Строка (иногда Запись),
Атрибут - Столбец, Поле.

При этом принимается, что "запись" означает "экземпляр записи", а "поле" означает "имя и тип поля".

3.2. Реляционная база данных

Реляционная база данных - это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Однако пользователи могут воспринимать такую базу данных как совокупность таблиц. Так на рис. 3.2 показаны таблицы базы данных, построенные по инфологической модели базы данных "Питание" рис. 2.4.

Блюда

БЛ	Блюдо	Вид
1	Лобио	Закуска
2	Харчо	Суп
3	Шашлык	Горячее
4	Кофе	Десерт

Расход

БЛ	Порций	Дата_Р
1	158	1/9/94
2	144	1/9/94
3	207	1/9/94
4	235	1/9/94
...	...	...

Продукты

ПР	Продукт	Калор.
1	Фасоль	3070
2	Лук	450
3	Масло	7420
4	Зелень	180
5	Мясо	1660
6	Томаты	240
7	Рис	3340
8	Кофе	2750

Рецепты

БЛ	Рецепт
1	Ломаную очищ

Состав

БЛ	ПР	Вес (г)
1	1	200
1	2	40
1	3	30
1	4	10
2	5	80
2	2	30
2	6	40
2	7	50
2	3	15
2	4	15
3	5	180
3	6	100

...	...

3	2	40
3	4	20
4	8	8

Поставщики

ПОС	Поставщик	Город
1	"Полесье"	Киев
2	"Наталка"	Киев
3	"Хуанхэ"	Пекин
4	"Лайма"	Рига
5	"Юрмала"	Рига
6	"Даугава"	Рига

Города

Город	Страна
Киев	Украина
Пекин	Китай
Рига	Латвия

Поставки

ПОС	ПР	Вес (кг)	Цена	Дата_П
1	6	120	0.45	27/8/94
1	3	50	1.82	27/8/94
1	2	50	0.61	27/8/94
2	2	100	0.52	27/8/94
2	5	100	2.18	27/8/94
2	4	10	0.88	27/8/94
3	1	250	0.37	24/8/94
3	7	75	0.44	24/8/94
3	8	40	2.87	24/8/94
4	3	70	1.56	30/8/94
5	5	200	2.05	30/8/94
6	6	15	0.99	30/8/94

Рис. 3.2. База данных "Питание" (см. п. 2.3)

1. Каждая таблица состоит из однотипных строк и имеет уникальное имя.

2. Строки имеют фиксированное число полей (столбцов) и значений (множественные поля и повторяющиеся группы недопустимы). Иначе говоря, в каждой позиции таблицы на пересечении строки и столбца всегда имеется в точности одно значение или ничего.

3. Строки таблицы обязательно отличаются друг от друга хотя бы единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы.

4. Столбцам таблицы однозначно присваиваются имена, и в каждом из них размещаются однородные значения данных (даты, фамилии, целые числа или денежные суммы).

5. Полное информационное содержание базы данных представляется в виде явных значений данных и такой метод представления является единственным. В частности, не существует каких-либо специальных "связей" или указателей, соединяющих одну таблицу с другой. Так, связи между строкой с БЛ = 2 таблицы "Блюда" на рис. 3.2 и строкой с ПР = 7 таблицы продукты (для приготовления Харчо нужен Рис), представляется не с помощью указателей, а благодаря существованию в таблице "Состав" строки, в которой номер блюда равен 2, а номер продукта - 7.

6. При выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию. Этому способствует наличие имен таблиц и их столбцов, а также возможность выделения любой их строки или любого набора строк с указанными признаками (например, рейсов с пунктом назначения "Париж" и временем прибытия до 12 часов).

3.3. Манипулирование реляционными данными

В главе 4 будет показано, что стремление к минимизации числа таблиц для хранения данных может привести к возникновению различных проблем при их обновлении и будут даны рекомендации по разбиению некоторых больших таблиц на несколько маленьких. Но как сформировать требуемый ответ, если нужные для него данные хранятся в разных таблицах?

Предложив реляционную модель данных, Э.Ф.Кодд создал и инструмент для удобной работы с отношениями - реляционную алгебру. Каждая операция этой алгебры использует одну или несколько таблиц (отношений) в качестве ее операндов и продуцирует в результате новую таблицу, т.е. позволяет "разрезать" или "склеивать" таблицы (рис. 3.3).

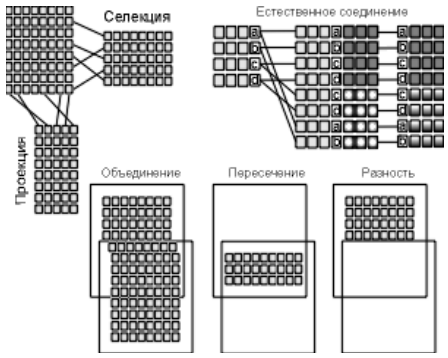


Рис. 3.3. Некоторые операции реляционной алгебры

Созданы языки манипулирования данными, позволяющие реализовать все операции реляционной алгебры и практически любые их сочетания. Среди них наиболее распространены SQL (Structured Query Language - *структуризованный язык запросов*) и QBE (Quere-By-Example - *запросы по образцу*) [3, 5]. Оба относятся к языкам очень высокого уровня, с помощью которых пользователь указывает, какие данные необходимо получить, не уточняя процедуру их получения.

С помощью единственного запроса на любом из этих языков можно соединить несколько таблиц во временную таблицу и вырезать из нее требуемые строки и столбцы (селекция и проекция).

Глава 4. Введение в проектирование реляционных баз данных

4.1. Цели проектирования

Только небольшие организации могут обобществить данные в одной полностью интегрированной базе данных. Чаще всего администратор баз данных (даже если это группа лиц) практически не в состоянии охватить и осмыслить все информационные требования сотрудников организации (т.е. будущих пользователей системы). Поэтому информационные системы больших организаций содержат несколько десятков БД, нередко распределенных

между несколькими взаимосвязанными ЭВМ различных подразделений. (Так в больших городах создается не одна, а несколько овощных баз, расположенных в разных районах.)

Отдельные БД могут объединять все данные, необходимые для решения одной или нескольких прикладных задач, или данные, относящиеся к какой-либо предметной области (например, финансам, студентам, преподавателям, кулинарии и т.п.). Первые обычно называют *прикладными БД*, а вторые - *предметными БД* (соотносящимся с предметами организации, а не с ее информационными приложениями). (Первые можно сравнить с базами материально-технического снабжения или отдыха, а вторые - с овощными и обувными базами.)

Предметные БД позволяют обеспечить поддержку любых текущих и будущих приложений, поскольку набор их элементов данных включает в себя наборы элементов данных прикладных БД. Вследствие этого предметные БД создают основу для обработки неформализованных, изменяющихся и неизвестных запросов и приложений (приложений, для которых невозможно заранее определить требования к данным). Такая гибкость и приспосабливаемость позволяет создавать на основе предметных БД достаточно стабильные информационные системы, т.е. системы, в которых большинство изменений можно осуществить без вынужденного переписывания старых приложений.

Основывая же проектирование БД на текущих и предвидимых приложениях, можно существенно ускорить создание высокоэффективной информационной системы, т.е. системы, структура которой учитывает наиболее часто встречающиеся пути доступа к данным. Поэтому прикладное проектирование до сих пор привлекает некоторых разработчиков. Однако по мере роста числа приложений таких информационных систем быстро увеличивается число прикладных БД, резко возрастает уровень дублирования данных и повышается стоимость их ведения.

Таким образом, каждый из рассмотренных подходов к проектированию воздействует на результаты проектирования в разных направлениях. Желание достичь и гибкости, и эффективности привело к формированию методологии проектирования, использующей как предметный, так и прикладной подходы. В общем случае предметный подход используется для построения первоначальной информационной структуры, а прикладной - для ее совершенствования с целью повышения эффективности обработки данных.

При проектировании информационной системы необходимо провести анализ целей этой системы и выявить требования к ней отдельных пользователей (сотрудников организации) [2, 3, 4, 6, 8, 9, 10]. Сбор данных начинается с изучения сущностей организации и процессов, использующих эти сущности (подробнее в приложении Б). Сущности группируются по "сходству" (частоте их использования для выполнения тех или иных действий) и по количеству ассоциативных связей между ними (самолет - пассажир, преподаватель - дисциплина, студент - сессия и т.д.). Сущности или группы сущностей, обладающие наибольшим сходством и (или) с наибольшей частотой ассоциативных связей объединяются в предметные БД. (Нередко сущности объединяются в предметные БД без использования формальных методик - по "здравому смыслу".) Для проектирования и ведения каждой предметной БД (нескольких БД) назначается АБД, который далее занимается детальным проектированием базы.

Далее будут рассматриваться вопросы, связанные с проектированием отдельных реляционных предметных БД.

Основная цель проектирования БД - это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте. Так называемый, "чистый" проект БД ("Каждый факт в одном месте") можно создать, используя методологию нормализации отношений. И хотя нормализация должна использоваться на завершающей проверочной стадии проектирования БД, мы начнем обсуждение вопросов проектирования с рассмотрения причин, которые заставили Кодда создать основы теории нормализации.

4.2. Универсальное отношение

Предположим, что проектирование базы данных "Питание" (рис. 3.2) начинается с выявления атрибутов и подбора данных, образец которых (часть блюд изготовленных и реализованных 1/9/94 г.) показан на рис. 4.1.

Этот вариант таблицы "Питание" не является отношением, так как большинство ее строк не атомарны. Атомарными являются лишь значения полей Блюдо, Вид, Рецепт (хотя он и большой), Порций и Дата_Р остальные же поля таблицы рис. 4.1 - множественные. Для придания таким данным формы отношения необходимо реконструировать таблицу. Наиболее просто это сделать с помощью простого процесса вставки, результат которой показан на рис. 4.2. Однако такое преобразование приводит к возникновению большого объема избыточных данных.

Блюдо	Вид	Рецепт	Порций	Дата Р	Продукт	Калорийность	Вес (г)	Поставщик	Город	Страна	Вес (кг)	Цена ($)	Дата П
Лобио	Закуска	Лом.	158	1/9/94	Фасоль	3070	200	"Хуанхэ"	Пекин	Китай	250	0.37	24/8/94
					Лук	450	40	"Наталка"	Киев	Украина	100	0.52	27/8/94
					Масло	7420	30	"Лайма"	Рига	Латвия	70	1.55	30/8/94
					Зелень	180	10	"Даугава"	Рига	Латвия	15	0.99	30/8/94
Харчо	Суп	...	144	1/9/94	Мясо	1660	80	"Наталка"	Киев	Украина	100	2.18	27/8/94
					Лук	450	30	"Наталка"	Киев	Украина	100	0.52	27/8/94
					Томаты	240	40	"Полесье"	Киев	Украина	120	0.45	27/8/94
					Рис	3340	50	"Хуанхэ"	Пекин	Китай	75	0.44	24/8/94
					Масло	7420	15	"Полесье"	Киев	Украина	50	1.62	27/8/94
					Зелень	180	15	"Наталка"	Киев	Украина	10	0.88	27/8/94
Шашлык	Горячее	...	207	1/9/94	Мясо	1660	180	"Юрмала"	Рига	Латвия	200	2.05	30/8/94
					Лук	450	40	"Полесье"	Киев	Украина	50	0.61	27/8/94
					Томаты	240	100	"Полесье"	Киев	Украина	120	0.45	27/8/94
					Зелень	180	20	"Даугава"	Рига	Латвия	15	0.99	30/8/94
Кофе	Десерт	...	235	1/9/94	Кофе	2750	8	"Хуанхэ"	Пекин	Китай	40	2.87	24/8/94

Рис. 4.1. Данные, необходимые для создания базы данных "Питание"

Таблица на рис. 4.2 представляет собой экземпляр корректного отношения. Его называют универсальным отношением проектируемой БД. В одно универсальное отношение включаются все представляющие интерес атрибуты, и оно может содержать все данные, которые предполагается размещать в БД в будущем. Для малых БД (включающих не более 15 атрибутов) универсальное отношение может использоваться в качестве отправной точки при проектировании БД.

Блюдо	Вид	Рецепт	Порций	Дата Р	Продукт	Калорийность	Вес (г)	Поставщик	Город	Страна	Вес (кг)	Цена ($)	Дата П
Лобио	Закуска	Лом.	158	1/9/94	Фасоль	3070	200	"Хуанхэ"	Пекин	Китай	250	0.37	24/8/94
Лобио	Закуска	Лом	108	1/9/94	Лук	450	40	"Наталка"	Киев	Украина	100	0.52	27/8/94
Лобио	Закуска	Лом	108	1/9/94	Масло	7420	30	"Лайма"	Рига	Латвия	70	1.55	30/8/94
Лобио	Закуска	Лом	108	1/9/94	Зелень	180	10	"Даугава"	Рига	Латвия	15	0.99	30/8/94
Харчо	Суп	...	144	1/9/94	Мясо	1660	80	"Наталка"	Киев	Украина	100	2.18	27/8/94
Харчо	Суп	...	144	1/9/94	Лук	450	30	"Наталка"	Киев	Украина	100	0.52	27/8/94
Харчо	Суп	...	144	1/9/94	Томаты	240	40	"Полесье"	Киев	Украина	120	0.45	27/8/94
Харчо	Суп	...	144	1/9/94	Рис	3340	50	"Хуанхэ"	Пекин	Китай	75	0.44	24/8/94
Харчо	Суп	...	144	1/9/94	Масло	7420	15	"Полесье"	Киев	Украина	50	1.62	27/8/94
Харчо	Суп	...	144	1/9/94	Зелень	180	15	"Наталка"	Киев	Украина	10	0.88	27/8/94
Шашлык	Горячее	...	207	1/9/94	Мясо	1660	180	"Юрмала"	Рига	Латвия	200	2.05	30/8/94
Шашлык	Горячее	...	207	1/9/94	Лук	450	40	"Полесье"	Киев	Украина	50	0.61	27/8/94
Шашлык	Горячее	...	207	1/9/94	Томаты	240	100	"Полесье"	Киев	Украина	120	0.45	27/8/94
Шашлык	Горячее	...	207	1/9/94	Зелень	180	20	"Даугава"	Рига	Латвия	15	0.99	30/8/94
Кофе	Десерт	...	235	1/9/94	Кофе	2750	8	"Хуанхэ"	Пекин	Китай	40	2.87	24/8/94

Рис. 4.2. Универсальное отношение "Питание"

4.3. Почему проект БД может быть плохим?

Начинающий проектировщик будет использовать отношение "Питание" (рис. 4.2) в качестве завершенной БД. Действительно, зачем разбивать отношение "Питание" на несколько более мелких отношений (см. например, рис. 3.2), если оно заключает в себе все данные? А разбивать надо потому, что при использовании универсального отношения возникает несколько проблем:

1. *Избыточность*. Данные практически всех столбцов многократно повторяются. Повторяются и некоторые наборы данных (Блюдо-Вид-Рецепт, Продукт-Калорийность, Поставщик-Город-Страна). Нежелательно повторение рецептов, некоторые из которых намного больше рецепта "Лобио" (см. рис. 2.3). И уж совсем плохо, что все данные о блюде (включая рецепт) повторяются каждый раз, когда это блюдо включается в меню.

2. *Потенциальная противоречивость (аномалии обновления)*. Вследствие избыточности можно обновить адрес поставщика в одной строке, оставляя его неизменным в других. Если поставщик кофе сообщил о своем переезде в Харбин и была обновлена строка с продуктом кофе, то у поставщика "Хуанхэ" появляется два адреса, один из которых не ак-

туален. Следовательно, при обновлениях необходимо просматривать всю таблицу для нахождения и изменения всех подходящих строк.

3. *Аномалии включения*. В БД не может быть записан новый поставщик ("Няринга", Вильнюс, Литва), если поставляемый им продукт (Огурцы) не используется ни в одном блюде. Можно, конечно, поместить неопределенные значения в столбцы Блюдо, Вид, Порций и Вес (г) для этого поставщика. Но если появится блюдо, в котором используется этот продукт, не забудем ли мы удалить строку с неопределенными значениями?

По аналогичным причинам нельзя ввести и новый продукт (например, Баклажаны), который предлагает существующий поставщик (например, "Полесье"). А как ввести новое блюдо, если в нем используется новый продукт (Крабы)?

4. *Аномалии удаления*. Обратная проблема возникает при необходимости удаления всех продуктов, поставляемых данным поставщиком или всех блюд, использующих эти продукты. При таких удалениях будут утрачены сведения о таком поставщике.

Многие проблемы этого примера исчезнут, если выделить в отдельные таблицы сведения о блюдах, рецептах, расходе блюд, продуктах и их поставщиках, а также создать связующие таблицы "Состав" и "Поставки" (рис. 4.3).

Блюда

Блюдо	**Вид**
Лобио	Закуска
Харчо	Суп
Шашлык	Горячее
Кофе	Десерт
...	...

Рецепты

Блюдо	**Рецепт**
Лобио	Ломаную очищ
...	...

Расход

Блюдо	**Порций**	**Дата_Р**
Лобио	158	1/9/94
Харчо	144	1/9/94
Шашлык	207	1/9/94
Кофе	235	1/9/94
...	...	...

Продукты

Продукт	**Калор.**
Фасоль	3070
Лук	450
Масло	7420
Зелень	180
Мясо	1660
...	...

Состав

Блюдо	**Продукт**	**Вес (г)**
Лобио	Фасоль	200
Лобио	Лук	40
Лобио	Масло	30
Лобио	Зелень	10
Харчо	Мясо	80
...	...	...

Поставщики

Поставщик	**Город**	**Страна**
"Полесье"	Киев	Украина
"Наталка"	Киев	Украина
"Хуанхэ"	Пекин	Китай
"Лайма"	Рига	Латвия
"Юрмала"	Рига	Латвия
...	...	...

Поставки

Поставщик	**Город**	**Продукт**	**Вес (кг)**	**Цена ($)**	**Дата_П**
"Полесье"	Киев	Томаты	120	0.45	27/8/94
"Полесье"	Киев	Масло	50	1.62	27/8/94

"Полесье"	Киев	Лук	50	0.61	27/8/94
"Наталка"	Киев	Лук	100	0.52	27/8/94
...	...	...	...	...	...

Рис. 4.3. Преобразование универсального отношения "Питание" (первый вариант)

Включение. Простым добавлением строк (Поставщики; "Няринга", Вильнюс, Литва) и (Поставки; "Няринга", Вильнюс, Огурцы, 40) можно ввести информацию о новом поставщике. Аналогично можно ввести данные о новом продукте (Продукты; Баклажаны, 240) и (Поставки; "Полесье", Киев, Баклажаны, 50).

Удаление. Удаление сведений о некоторых поставках или блюдах не приводит к потере сведений о поставщиках.

Обновление. В таблицах рис. 4.3 все еще много повторяющихся данных, находящихся в связующих таблицах (Состав и Поставки). Следовательно, в данном варианте БД сохранилась потенциальная противоречивость: для изменения названия поставщика с "Полесье" на "Днепро" придется изменять не только строку таблицы Поставщики, но и множество строк таблицы Поставки. При этом не исключено, что в БД будут одновременно храниться: "Полесье", "Палесье", "Днепро", "Днипро" и другие варианты названий.

Кроме того, повторяющиеся текстовые данные (такие как название блюда "Рулет из телячей грудинки с сосисками и гарниром из разноцветного пюре" или продукта "Колбаса московская сырокопченая") существенно увеличивают объем хранимых данных.

Для исключения ссылок на длинные текстовые значения последние обычно нумеруют: нумеруют блюда в больших кулинарных книгах, товары (продукты) в каталогах и т.д. Воспользуемся этим приемом для исключения избыточного дублирования данных и появления ошибок при копировании длинных текстовых значений (рис. 4.4). Теперь при изменении названия поставщика "Полесье" на "Днепро" исправляется единственное значение в таблице Поставщики. И даже если оно вводится с ошибкой ("Днипро"), то это не может повлиять на связь между поставщиками и продуктами (в связующей таблице Поставки используются номера поставщиков и продуктов, а не их названия).

Блюда

БЛ	**Блюдо**	**Вид**
1	Лобио	Закуска
2	Харчо	Суп
3	Шашлык	Горячее
4	Кофе	Десерт
...	...	...

Рецепты

Блюдо	**Рецепт**
Лобио	Ломаную очищ
...	...

Расход

Блюдо	**Порций**	**Дата_Р**
Лобио	158	1/9/94
Харчо	144	1/9/94
Шашлык	207	1/9/94
Кофе	235	1/9/94
...	...	...

Продукты

ПР	**Продукт**	**Калор.**

Состав

БЛ	**ПР**	**Вес (г)**

Поставщики

ПОС	**Поставщик**	**Город**	**Страна**

1	Фасоль	3070
2	Лук	450
3	Масло	7420
4	Зелень	180
5	Мясо	1660
...	...	...

1	1	200
1	2	40
1	3	30
1	4	10
2	5	80
...	...	...

1	"Полесье"	Киев	Украина
2	"Наталка"	Киев	Украина
3	"Хуанхэ"	Пекин	Китай
4	"Лайма"	Рига	Латвия
5	"Юрмала"	Рига	Латвия
...	...	...	...

Поставки

ПОС	ПР	Вес (кг)	Цена ($)	Дата_П
1	6	120	0.45	27/8/94
1	3	50	1.62	27/8/94
1	2	50	0.61	27/8/94
2	2	100	0.52	27/8/94
...	...	...	...	...

Рис. 4.4. Преобразование универсального отношения "Питание" (второй вариант)

4.4. О нормализации, функциональных и многозначных зависимостях

Нормализация - это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором *каждый факт появляется лишь в одном месте*, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Как указывалось в п. 3.1, каждая таблица в реляционной БД удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное атомарное значение, и никогда не может быть множества таких значений. Любая таблица, удовлетворяющая этому условию, называется *нормализованной* (см. таблицы рис. 4.2 - 4.4). Фактически, ненормализованные таблицы, т.е. таблицы, содержащие повторяющиеся группы (см. рис. 4.1), даже не допускаются в реляционной БД.

Всякая нормализованная таблица автоматически считается таблицей в *первой нормальной форме*, сокращенно *1НФ*. Таким образом, строго говоря, "нормализованная" и "находящаяся в 1НФ" означают одно и то же. Однако на практике термин "нормализованная" часто используется в более узком смысле - "полностью нормализованная", который означает, что в проекте не нарушаются никакие принципы нормализации.

Теперь в дополнение к 1НФ можно определить дальнейшие уровни нормализации - *вторую нормальную форму* (*2НФ*), *третью нормальную форму* (*3НФ*) и т.д. По существу, таблица находится в 2НФ, если она находится в 1НФ и удовлетворяет, кроме того, некоторому дополнительному условию, суть которого будет рассмотрена ниже. Таблица находится

в 3НФ, если она находится в 2НФ и, помимо этого, удовлетворяет еще другому дополнительному условию и т.д.

Таким образом, каждая нормальная форма является в некотором смысле более ограниченной, но и более *желательной*, чем предшествующая. Это связано с тем, что "(N+1)-я нормальная форма" не обладает некоторыми непривлекательными особенностями, свойственным "N-й нормальной форме". Общий смысл дополнительного условия, налагаемого на (N+1)-ю нормальную форму по отношению к N-й нормальной форме, состоит в исключении этих непривлекательных особенностей. В п. 4.3 мы выявляли непривлекательные особенности таблицы рис. 4.2 и для их исключения выполняли "интуитивную нормализацию".

Теория нормализации основывается на наличии той или иной зависимости между полями таблицы. Определены два вида таких зависимостей: функциональные и многозначные.

Функциональная зависимость. Поле B таблицы функционально зависит от поля A той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля A обязательно существует только одно из различных значений поля B. Отметим, что здесь допускается, что поля A и B могут быть составными.

Например, в таблице Блюда (рис. 4.4) поля Блюдо и Вид функционально зависят от ключа БЛ, а в таблице Поставщики рис. 4.3 поле Страна функционально зависит от составного ключа (Поставщик, Город). Однако последняя зависимость не является функционально полной, так как Страна функционально зависит и от части ключа - поля Город.

Полная функциональная зависимость. Поле B находится в полной функциональной зависимости от составного поля A, если оно функционально зависит от A и не зависит функционально от любого подмножества поля A.

Многозначная зависимость. Поле A многозначно определяет поле B той же таблицы, если для каждого значения поля A существует хорошо определенное множество соответствующих значений B.

Обучение

Дисциплина	**Преподаватель**	**Учебник**
Информатика	Шипилов П.А.	Форсайт Р. Паскаль для всех
Информатика	Шипилов П.А.	Уэйт М. и др. Язык Си
Информатика	Голованевский Г.Л.	Форсайт Р. Паскаль для всех
Информатика	Голованевский Г.Л.	Уэйт М. и др. Язык Си
...	...	...

Рис. 4.5. К иллюстрации многозначных зависимостей

Для примера рассмотрим таблицу "Обучение" (рис. 4.5). В ней есть многозначная зависимость "Дисциплина-Преподаватель": дисциплина (в примере Информатика) может может читаться несколькими преподавателями (в примере Шипиловым и Голованевским). Есть и другая многозначная зависимость "Дисциплина-Учебник": при изучении Информатики используются учебники "Паскаль для всех" и "Язык Си". При этом Преподаватель и Учебник не связныфункциональной зависимостью, что приводит к появлению избыточности

(для добавление еще одного учебника придется ввести в таблицу две новых строки). Дело улучшается при замене этой таблицы на две: (Дисциплина-Преподаватель и Дисциплина-Учебник).

4.5. Нормальные формы

В п. 4.4 было дано определение первой нормальной формы (1НФ). Приведем здесь более строгое ее определение, а также определения других нормальных форм.

Таблица находится в *первой нормальной форме* (*1НФ*) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто.

Из таблиц, рассмотренных в п. 4, не удовлетворяет этим требованиям (т.е. не находится в 1НФ) только таблица рис. 4.1.

Таблица находится во *второй нормальной форме* (*2НФ*), если она удовлетворяет определению 1НФ и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Кроме таблицы рис. 4.1 не удовлетворяет этим требованиям только таблица 4.2.

Как обосновано ниже (пример 4.2) она имеет составной первичный ключ

```
Блюдо, Дата_Р, Продукт, Поставщик, Город, Дата_П
```

и содержит множество неключевых полей (Вид, Рецепт, Порций, Калорийность и т.д.), зависящих лишь от той или иной части первичного ключа. Так поля Вид и Рецепт зависят только от поля Блюдо, Калорийность - от поля Продукт и т.п. Следовательно, эти поля не связаны с первичным ключом полной функциональной зависимостью.

Ко второй нормальной форме приведены почти все таблицы рис. 4.3 кроме таблицы Поставщики, в которой Страна зависит только от поля Город, который является частью первичного ключа (Поставщик, Город). Последнее обстоятельство приводит к проблемам при:

- включении данных (пока не появится поставщик из Вильнюса, нельзя зафиксировать, что этот город Литвы),
- удалении данных (исключение поставщика может привести к потере информации о местонахождении города),
- обновлении данных (при изменении названия страны приходится просматривать множество строк, чтобы исключить получение противоречивого результата).

Разбивая эту таблицу на две таблицы Поставщики и Города (рис. 3.2), можно исключить указанные аномалии.

Что же касается таблиц рис. 4.4, то ввод в них отсутствующих в предметной области цифровых первичных и внешних ключей формально затрудняет процедуру выявления функциональных связей между этими ключами и остальными полями. Действительно, легко

установить связь между атрибутом Блюдо и Вид (блюда): Харчо - Суп, Лобио - Закуска и т.п., но нет прямой зависимости между полями БЛ и Вид (блюда), если не помнить, что значение БЛ соответствует номеру блюда.

Для упрощения нормализации подобных таблиц целесообразно использовать следующую рекомендацию.

Рекомендация. При проведении нормализации таблиц, в которые введены цифровые (или другие) заменители составных и (или) текстовых первичных и внешних ключей, следует хотя бы мысленно подменять их на исходные ключи, а после окончания нормализации снова восстанавливать.

При использовании этой рекомендации таблицы рис. 4.4 временно превращаются в таблицы рис. 4.3, а после выполнения нормализации и восстановления полей БЛ, ПР и ПОС - в нормализованные таблицы рис. 3.2.

Таблица находится в *третьей нормальной форме* (*3НФ*), если она удовлетворяет определению 2НФ и не одно из ее неключевых полей не зависит функционально от любого другого неключевого поля.

После разделения таблицы Поставщики рис. 4.3 на две части все таблицы этого проекта удовлетворяют определению 2НФ, а так как в них нет неключевых полей, функционально зависящих друг от друга, то все они находятся в 3НФ.

Как ни странно, этого нельзя сказать об аналогичных таблицах рис 4.4. Если забыть рекомендацию о подмене на время нормализации ключей БЛ, ПР и ПОС на Блюдо, Продукт и (Поставщик, Город), то среди этих таблиц появятся две, не удовлетворяющие определению 3НФ. Действительно, так как после ввода первичных ключей БЛ и ПР поля Блюдо и Продукт стали неключевыми - появились несуществовавшие ранее функциональные зависимости между неключевыми полями:

```
Блюдо->Вид и Продукт->Калорийность.
```

Следовательно, для приведения таблиц Блюда и Продукты рис. 4.4 к 3НФ их надо разбить на

```
Блюда(БЛ, Блюдо),
Вид_блюда(БЛ, Вид);
Продукты(ПР, Продукт);
Калор_прод(ПР,Калорийносить),
```

хотя интуиция подсказывает, что это лишнее разбиение, совсем не улучшающее проекта базы данных.

Столкнувшись с подобными несуразностями, которые могут возникать не только из-за введения кодированных первичных ключей, теоретики реляционных систем Кодд и Бойс обосновали и предложили более строгое определение для 3НФ, которое учитывает, что в таблице может быть несколько *возможных* ключей.

Таблица находится в *нормальной форме Бойса-Кодда* (*НФБК*), если и только если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от *возможного* ключа.

В соответствие с этой формулировкой таблицы Блюда и Продукты рис. 4.4, имеющие по паре возможных ключей (БЛ и Блюдо) и (ПР и Продукт) находятся в НФБК или в 3НФ.

В следующих нормальных формах (4НФ и 5НФ) учитываются не только функциональные, но и многозначные зависимости между полями таблицы. Для их описания познакомимся с понятием полной декомпозиции таблицы.

Полной декомпозицией таблицы называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым таблицы.

Например, естественным соединением (см. п. 3.3) таблиц рис. 4.3 можно образовать исходную таблицу, приведенную на рис. 4.2. Ту же таблицу можно получить композицией таблиц рис. 3.2. Следовательно, таблицы рис. 4.3, 4.4 и 3.2 являются полными декомпозициями таблицы Питание рис. 4.2.

Теперь можно дать определения высших нормальных форм. И сначала будет дано определение для последней из предложенных - 5НФ.

Таблица находится в *пятой нормальной форме* (*5НФ*) тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный ключ. Таблица, не имеющая ни одной полной декомпозиции, также находится в 5НФ.

Четвертая нормальная форма (*4НФ*) является частным случаем 5НФ, когда полная декомпозиция должна быть соединением ровно двух проекций. Весьма не просто подобрать реальную таблицу, которая находилась бы в 4НФ, но не была бы в 5НФ.

4.6. Процедура нормализации

Как уже говорилось, нормализация - это разбиение таблицы на несколько, обладающих лучшими свойствами при обновлении, включении и удалении данных. Теперь можно дать и другое определение: нормализация - это процесс последовательной замены таблицы ее полными декомпозициями до тех пор, пока все они не будут находиться в 5НФ. На практике же достаточно привести таблицы к НФБК и с большой гарантией считать, что они находятся в 5НФ. Разумеется, этот факт нуждается в проверке, однако пока не существует эффективного алгоритма такой проверки. Поэтому остановимся лишь на процедуре приведения таблиц к НФБК.

Эта процедура основывается на том, что единственными функциональными зависимостями в любой таблице должны быть зависимости вида K->F, где K - первичный ключ, а F - некоторое другое поле. Заметим, что это следует из определения первичного ключа таблицы, в соответствии с которым K->F всегда имеет место для всех полей данной таблицы. "Один факт в одном месте" говорит о том, что не имеют силы никакие другие функциональные зависимости. Цель нормализации состоит именно в том, чтобы избавиться от

всех этих "других" функциональных зависимостей, т.е. таких, которые имеют иной вид, чем K->F.

Если воспользоваться рекомендацией п. 4.5 и подменить на время нормализации коды первичных (внешних) ключей на исходные ключи, то, по существу, следует рассмотреть лишь два случая:

1. Таблица имеет составной первичный ключ вида, скажем, (K1,K2), и включает также поле F, которое функционально зависит от части этого ключа, например, от K2, но не от полного ключа. В этом случае рекомендуется сформировать другую таблицу, содержащую K2 и F (первичный ключ - K2), и удалить F из первоначальной таблицы:

```
Заменить T(K1,K2,F),  первичный ключ (K1,K2), ФЗ K2->F
 на       T1(K1,K2),   первичный ключ (K1,K2),
 и        T2(K2,F),    первичный ключ K2.
```

2. Таблица имеет первичный (возможный) ключ K, не являющееся возможным ключом поле F1, которое, конечно, функционально зависит от K, и другое неключевое поле F2, которое функционально зависит от F1. Решение здесь, по существу, то же самое, что и прежде - формируется другая таблица, содержащая F1 и F2, с первичным ключом F1, и F2 удаляется из первоначальной таблицы:

```
Заменить T(K,F1,F2),  первичный ключ K, ФЗ F1->F2
 на       T1(K,F1),    первичный ключ K,
 и        T2(F1,F2),   первичный ключ F1.
```

Для любой заданной таблицы, повторяя применение двух рассмотренных правил, почти во всех практических ситуациях можно получить в конечном счете множество таблиц, которые находятся в "окончательной" нормальной форме и, таким образом, не содержат каких-либо функциональных зависимостей вида, отличного от K->F.

Для выполнения этих операций необходимо первоначально иметь в качестве входных данных какие-либо "большие" таблицы (например, универсальные отношения). Но нормализация ничего не говорит о том, как получить эти большие таблицы. В следующей главе будет рассмотрена процедура получения таких исходных таблиц, а здесь приведем примеры нормализации.

Пример 4.1. Применим рассмотренные правила для полной нормализации универсального отношения "Питание" (рис. 4.2).

Шаг 1. Определение первичного ключа таблицы.

Предположим, что каждое блюдо имеет уникальное название, относится к единственному виду и приготавливается по единственному рецепту, т.е. название блюда однозначно определяет его вид и рецепт. Предположим также, что название организации поставщика уникально для того города, в котором он расположен, и названия городов уникальны для каждой из стран, т.е. название поставщика и город однозначно определяют этого поставщика, а город - страну его нахождения. Наконец, предположим, что поставщик может осуществлять в один и тот же день только одну поставку каждого продукта, т.е. название продукта, название организации поставщика, город и дата поставки однозначно определяют вес и цену поставленного продукта. Тогда в качестве первичного ключа отношения "Питание" можно использовать следующий набор атрибутов:

```
Блюдо, Дата_Р, Продукт, Поставщик, Город, Дата_П.
```

Шаг 2. Выявление полей, функционально зависящих от части составного ключа.

Поле Вид функционально зависит только от поля Блюдо, т.е.

```
Блюдо->Вид.
```

Аналогичным образом можно получить зависимости:

```
Блюдо->Рецепт
(Блюдо, Дата_Р)->Порций
Продукт->Калорийность
(Блюдо, Продукт)->Вес
Город->Страна
(Поставщик, Город, Дата_П)->Цена
```

Шаг 3. Формирование новых таблиц.

Полученные функциональные зависимости определяют состав таблиц, которые можно сформировать из данных универсального отношения:

```
Блюда (Блюдо, Вид)
Рецепты (Блюдо, Рецепт)
Расход (Блюдо, Дата_Р, Порций)
Продукты (Продукт, Калорийность)
Состав (Блюдо, Продукт, Вес (г))
Города (Город, Страна)
Поставки (Поставщик, Город, Дата_П, Вес (кг), Цена).
```

Шаг 4. Корректировка исходной таблицы.

После выделения из состава универсального отношения указанных выше таблиц, там остались лишь сведения о поставщиках, для хранения которых целесообразно создать таблицу

```
Поставщики (Поставщик, Город),
```

т.е. использовать часть исходного первичного ключа, так как остальные его части уже ничего не определяют.

Таким образом, процедура последовательной нормализации позволила получить проект, лучший, чем приведен на рис. 4.3.

Пример 4.2. Для улучшения проекта, приведенного на рис. 4.4, нужно определить первичные ключи таблиц и выявить, нет ли в таблицах полей, зависящих лишь от части этих ключей. Такое поле есть только в одной таблице. Это поле Страна в таблице Поставщики. Выделяя его вместе с ключем Город в таблицу Страны, получим проект, приведенный на рис. 3.2.

4.7. Процедура проектирования

Процесс проектирования информационных систем является достаточно сложной задачей. Он начинается с построения инфологической модели данных (п. 2), т.е. идентификации

сущностей. Затем необходимо выполнить следующие шаги процедуры проектирования даталогической модели.

1. Представить каждый стержень (независимую сущность) таблицей базы данных (базовой таблицей) и специфицировать первичный ключ этой базовой таблицы.

2. Представить каждую ассоциацию (связь вида "многие-ко-многим" или "многие-ко-многим-ко-многим" и т.д. между сущностями) как базовую таблицу. Использовать в этой таблице внешние ключи для идентификации участников ассоциации и специфицировать ограничения, связанные с каждым из этих внешних ключей.

3. Представить каждую характеристику как базовую таблицу с внешним ключом, идентифицирующим сущность, описываемую этой характеристикой. Специфицировать ограничения на внешний ключ этой таблицы и ее первичный ключ - по всей вероятности, комбинации этого внешнего ключа и свойства, которое гарантирует "уникальность в рамках описываемой сущности".

4. Представить каждое обозначение, которое не рассматривалось в предыдущем пункте, как базовую таблицу с внешним ключом, идентифицирующим обозначаемую сущность. Специфицировать связанные с каждым таким внешним ключом ограничения.

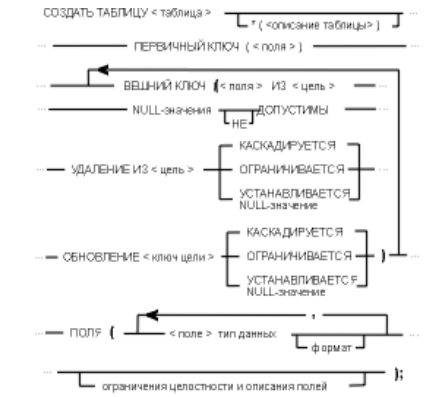
5. Представить каждое свойство как поле в базовой таблице, представляющей сущность, которая непосредственно описывается этим свойством.

6. Для того чтобы исключить в проекте непреднамеренные нарушения каких-либо принципов нормализации, выполнить описанную в п. 4.6 процедуру нормализации.

7. Если в процессе нормализации было произведено разделение каких-либо таблиц, то следует модифицировать инфологическую модель базы данных и повторить перечисленные шаги.

8. Указать ограничения целостности проектируемой базы данных и дать (если это необходимо) краткое описание полученных таблиц и их полей.

На рис. 4.6 показан синтаксис предложения, предлагаемого для регистрации принимаемых проектных решений.



В этом описании:

"таблица"	– имя проектируемой базовой таблицы;
"описание таблицы"	– любой текст, характеризующий назначение и (или) содержание таблицы;
"поля"	– список имен полей, разделенных запятыми;
"цель"	– имя таблицы, первичный ключ которой используется как внешний ключ проектируемой;
"ключ цели"	– специфицирует "первичный ключ" цели;
"поле"	– имя поля проектируемой таблицы (включая и имена внешних ключей).

Рис. 4.6. Синтаксис описания проектных решений

Для примера приведем описания таблиц "Блюда" и "Состав":

```
СОЗДАТЬ ТАБЛИЦУ  Блюда *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ  ( БЛ )
           ПОЛЯ  ( БЛ Целое, Блюдо Текст 60, Вид Текст 7 )
    ОГРАНИЧЕНИЯ  ( 1. Значения поля Блюдо должны быть
                   уникальными; при нарушении вывод
                   сообщения "Такое блюдо уже есть".
                   2. Значения поля Вид должны принадлежать
                   набору: Закуска, Суп, Горячее, Десерт,
                   Напиток; при нарушении вывод сообщения
                   "Можно лишь Закуска, Суп, Горячее,
                   Десерт, Напиток");
СОЗДАТЬ ТАБЛИЦУ  Состав *( Связывает Блюда и Продукты )
 ПЕРВИЧНЫЙ КЛЮЧ  ( БЛ, ПР )
   ВНЕШНИЙ КЛЮЧ  ( БЛ ИЗ Блюда
                   NULL-значения НЕ ДОПУСТИМЫ
                   УДАЛЕНИЕ ИЗ Блюда КАСКАДИРУЕТСЯ
                   ОБНОВЛЕНИЕ Блюда.БЛ КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ  ( ПР ИЗ Продукты
                   NULL-значения НЕ ДОПУСТИМЫ
                   УДАЛЕНИЕ ИЗ Продукты ОГРАНИЧИВАЕТСЯ
                   ОБНОВЛЕНИЕ Продукты.ПР КАСКАДИРУЕТСЯ)
           ПОЛЯ  ( БЛ Целое, ПР Целое, Вес Целое )
    ОГРАНИЧЕНИЯ  ( 1. Значения полей БЛ и ПР должны принадлежать
                   набору значений из соответствующих полей таблиц
                   Блюда и Продукты; при нарушении вывод сообщения
                   "Такого блюда нет" или "Такого продукта нет".
                   2. Значение поля Вес должно лежать в пределах
                   от 0.1 до 500 г. );
```

Рассмотренный язык описания данных, основанный на языке SQL [5], позволяет дать удобное и полное описание любой сущности и, следовательно, всей базы данных. Однако такое описание, как и любое подробное описание, не отличается наглядностью. Для достижения большей иллюстративности целесообразно дополнять проект инфологической моделью, но менее громоздкой, чем рассмотренная в главе 2.

Для наиболее распространенных реляционных баз данных можно предложить язык инфологического моделирования "Таблица-связь", пример использования которого приведен на рис. 4.7. В нем все сущности изображаются одностолбцовыми таблицами с заголовками, состоящими из имени и типа сущности. Строки таблицы - это перечень атрибутов сущности, а те из них, которые составляют первичный ключ, располагаются рядом и обводятся рамкой. Связи между сущностями указываются стрелками, направленными от первичных ключей или их составляющих.

Рис. 4.7. Инфологическая модель базы данных "Питание", построенная с помощью языка "Таблицы-связи"

4.8. Различные советы и рекомендации

Векторы. Представляйте векторы по столбцам, а не по строкам. Например, диаграмму продаж товаров x, y, ... за последние годы лучше представить в виде:

```
ТОВАР    МЕСЯЦ   КОЛ-ВО
-----   -------  ------
  x     ЯНВАРЬ    100
  x     ФЕВРАЛЬ    50
 ...      ...     ...
  x     ДЕКАБРЬ   360
  y     ЯНВАРЬ     75
  y     ФЕВРАЛЬ   144
 ...      ...     ...
  y     ДЕКАБРЬ    35
 ...      ...     ...
```

а не так, как показано ниже:

```
ТОВАР   КОЛ-ВО   КОЛ-ВО          КОЛ-ВО
        ЯНВАРЬ   ФЕВРАЛЬ  ...   ДЕКАБРЬ
-----   -------  -------        -------
  x       100       50    ...     360
  y        75      144    ...      35
 ...      ...       ...    ...     ...
```

Одна из причин такой рекомендации заключается в том, что при этом значительно проще записываются обобщенные (параметризованные) запросы. Рассмотрите, например, как выглядит сравнение сведений из диаграммы продаж товара i в месяце с номером m со сведениями для товара j в месяце с номером n, где i, j, m и n - параметры.

Неопределенные значения. Будьте очень внимательны с неопределенными (NULL) значениями. В поведении неопределенных значений проявляется много произвола и противоречивости. В разных СУБД при выполнении различных операций (сравнение, объединение, сортировка, группирование и другие) два неопределенных значения могут быть или не быть равными друг другу. Они могут по разному влиять на результат выполнения опе-

раций по определению средних значений и нахождения количества значений. Для исключения ошибок в ряде СУБД существует возможность замены NULL-значения нулем при выполнении расчетов, объявление всех NULL-значений равными друг другу и т.п.

Глава 5. Пример проектирования базы данных "Библиотека"

5.1. Назначение и предметная область

База данных предназначена для хранения данных о приобретенных библиотекой изданиях (монографиях, справочниках, сборниках статей и т.п.), информации о местонахождении отдельных экземпляров (переплетов) каждого издания и сведений о читателях.

Д27 **Дейт К.** Руководство по реляционной СУБД DB2 / Пер. с англ. и предисл. М.Р.Когаловского. - М.: Финансы и статистика, 1988. - 320 с.: ил.

ISBN 5-279-00063-9

Книга американского специалиста в области реляционных баз данных К.Дейта, автора популярной в СССР монографии "Введение в системы баз данных" (М.: Наука, 1981), представляет собой руководство по перспективной СУБД фирмы ИБМ DB2, сочетающей возможности широко известной системы IMS/VS и реляционной СУБД.
Для специалистов по программному обеспечению информационных систем и студентов вузов.

ББК 32.973

Рис. 5.1. Макет аннотированной каталожной карточки

Для ведения библиотечных каталогов, организации поиска требуемых изданий и библиотечной статистики в базе должны храниться сведения, большая часть которых размещаются в аннотированных каталожных карточках (рис. 5.1). Анализ запросов на литературу (как читателями, так и сотрудниками библиотек) показывает, что для поиска подходящих изданий (по тематике, автору, художнику, издательству и т.п.) и отбора нужного (например, по аннотации) следует выделить следующие атрибуты каталожной карточки:

1. Автор (фамилия и имена (инициалы) или псевдоним каждого автора издания).

2. Название (заглавие) издания.

3. Номер тома (части, книги, выпуска).

4. Вид издания (сборник, справочник, монография, ...).

5. Составитель (фамилия и имена (инициалы) каждого из составителе издания).

6. Язык, с которого выполнен перевод издания.

7. Переводчик (фамилия и инициалы каждого переводчика).

8. Под чей редакцией (фамилия и имена (инициалы) каждого из титульных редакторов).

9. Художник (фамилия и имена (инициалы) каждого художника-иллюстратора) - для художественных изданий, иллюстрируемых оригинальными рисунками.

10. Повторность издания (второе, одиннадцатое и т.п.).

11. Характер переиздания (исправленное, дополненное, переработанное, стереотипное и т.п.).

12. Место издания (город).

13. Издательство (название издательства).

14. Год выпуска издания.

15. Издательская аннотация или реферат.

16. Библиотечный шифр (например, ББК 32.973).

17. Авторский знак (например, Д27).

Библиотечный шифр и авторский знак используются при составлении каталогов и организации расстановки изданий на полках: по содержанию (в соответствии с библиотечным шифром) и алфавиту (в соответствии с авторским знаком).

Библиотечно-библиографическая классификация (ББК) распределяет издания по отраслям знания в соответствии с их содержанием. В ней используется цифро-буквенные индексы ступенчатой структуры.

Каждый из девяти классов (1. Марксизм-ленинизм; 2. Естественные науки; 3. Техника. Технические науки; 4. Сельское и лесное хозяйство; 5. Здравоохранение; 6/8. Общественные и гуманитарные науки; 9. Библиографические пособия. Справочные издания. Журналы.) делится на подклассы и следующие ступени деления:

3. Техника. Технические науки.
32 Радиоэлектроника.
32.97 Вычислительная техника.
32.973 Электронные вычислительные машины и устройства.
32.973.2 Электронно вычислительные машины и устройства дискретного действия.

Шифр ББК используется при выделении хранимым изданиям определенных комнат, стеллажей и полок, а также для составления каталогов и статистических отчетов.

Авторский знак, состоящий из первой буквы фамилии (псевдонима) автора или названия издания (для изданий без автора) и числа, соответствующего слогу, наиболее приближающегося по написанию к первым буквам фамилии (названия), упрощает расстановку книг на полках в алфавитном порядке.

К объектам и атрибутам, позволяющим охарактеризовать отдельные экземпляры изданий (переплеты), места их хранения и читателей, можно отнести:

18. Номер комнаты (помещения для хранения переплетов).

19. Номер стеллажа в комнате.

20. Номер полки на стеллаже.

21. Номер (инвентарный номер) переплета.

22. Дата приобретения конкретного переплета.

23. Цена конкретного переплета.

24. Дата размещения конкретного переплета на конкретном месте.

25. Дата изъятия переплета с установленного места.

26. Номер читательского билета (формуляра).

27. Фамилия читателя.

28. Имя читателя.

29. Отчество читателя.

30. Адрес читателя.

31. Телефон читателя.

32. Дата выдачи читателю конкретного переплета.

33. Срок, на который конкретный переплет выдан читателю.

34. Дата возврата переплета.

5.2. Построение инфологической модели

Анализ определенных выше объектов и атрибутов позволяет выделить сущности проектируемой базы данных и, приняв решение о создании реляционной базы данных, построить ее инфологическую модель на языке "Таблицы-связи" (рис. 5.2).

К стержневым сущностям можно отнести:

1. Создатели (Код создателя, Создатель).

   Эта сущность отводится для хранения сведений об основных людях, принимавших участие в подготовке рукописи издания (авторах, составителях, титульных редакторах, переводчиках и художниках). Такое объединение допустимо, так как данные о разных создателях выбираются из одного домена (фамилия и имена) и исключает дублирование данных (один и тот же человек может играть разные роли в подготовке разных изданий). Например, С.Я.Маршак писал стихи (Сказка о глупом мышонке) и пьесы (Двенадцать месяцев), переводил Дж.Байрона, Р.Бернса, Г.Гейне и составлял сборники стихов.

   Так как фамилия и имена (инициалы) создателя могут быть достаточно громоздки-

ми (М.Е. Салтыков-Щедрин, Франсуа Рене де Шатобриан, Остен Жюль Жан-Батист Ипполит и т.п.) и будут многократно встречаться в разных изданиях, то их целесообразно нумеровать и ссылаться на эти номера. Для этого вводится целочисленный атрибут "Код_создателя", который будет автоматически наращиваться на единицу при вводе в базу данных нового автора, переводчика или другого создателя.

Аналогично создаются: Код_издательства, Код_заглавия, Вид_ издания, Код_характера, Код_языка, Номер_билета, Номер_пере- плета, Код_места и Код_издания, замещающие от одного до девяти атрибутов.

2. Издательства (Код_издательства, Название, Город).
3. Заглавия (Код_заглавия, Заглавие).

   Выделение этой сущности позволит сократить объем данных и снизить вероятность возникновения противоречивости (исключается необходимость ввода длинных текстовых названий для различных томов собраний сочинений, повторных изданий, учебников и т.п.).
4. Вид_издания (Вид_издания, Название_вида).
5. Характеры (Код_характера, Характер_переиздания).
6. Языки (Код_языка, Язык, Сокращение).

   Кроме названия языка хранится его общепринятое сокращение (англ., исп., нем., фр.), если оно существует.
7. Места (Код_места, Номер_комнаты, Номер_стеллажа, Номер_ полки).

   Один из кодов этой сущности (например, "-1") отведен для описания обобщенного места, находящегося за стенами хранилища книг (издание выдано читателю, временно передано другой библиотеке или организации).
8. Читатели (Номер_билета, Фамилия, Имя, Отчество, Адрес, Телефон).

Две ключевые сущности, описывающие издание и его конкретные экземпляры, оказываются зависимыми от других сущностей и попадают в класс обозначений:

1. Издание (Код_издания, Код_заглавия, Вид_издания, Номер_тома, Авторский_знак, Библиотечн_шифр, Повторность, Код_издательства, Год_издания, Аннотация) [Заглавия, Вид_издания, Издательства];
2. Переплеты (Номер_переплета, Код_издания, Цена, Дата_приобретения)[Издания];

Стержневые сущности и обозначения связаны между собой ассоциациями:

1. Авторы [Создатели M, Издание N] (Код_создателя, Код_издания).
2. Составители [Создатели M, Издания N] (Код_создателя, Код_издания).
3. Редакторы [Создатели M, Издания N] (Код_создателя, Код_издания).
4. Художники [Создатели M, Издания N] (Код_создателя, Код_издания).
5. Переводчики [Создатели M, Издания N] (Код_создателя, Код_издания, Язык).
6. Переиздания [Характеры M, Издания N] (Код_характера, Код_издания).
7. Размещение [Места M, Переплеты N] (Код_места, Номер_переплета, Дата_размещения, Дата_изъятия).
8. Выдача [Читатели M, Переплеты N] (Номер_билета, Номер_переплета, Дата_выдачи, Срок, Дата_возврата).

И, наконец, для уменьшения объема часто используемого обозначения "Издания" из него выделена характеристика:

1. Аннотации (Код_издания, Аннотация) {Издание}.

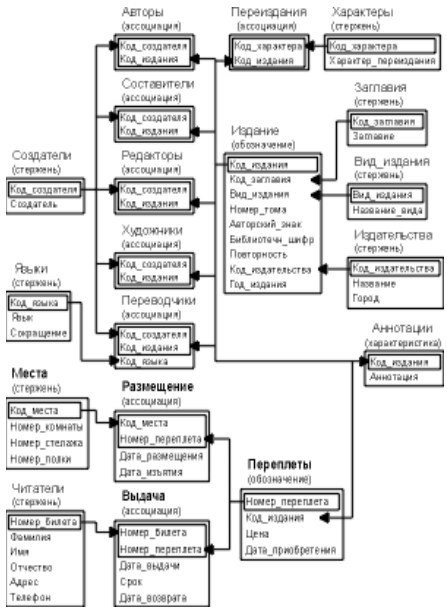


Рис. 5.2. Инфологическая модель базы данных "Библиотека", построенная с помощью языка "Таблицы-связи"

5.3. Проектирование базы данных

В соответствие с процедурой проектирования (п. 4.4) каждая из полученных сущностей должна быть представлена базовой таблицей. Первый вариант этих таблиц описывается так:

```
СОЗДАТЬ ТАБЛИЦУ Создатели *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_создат )
           ПОЛЯ ( Код_создат Целое, Фам_ИО Текст 30 );
СОЗДАТЬ ТАБЛИЦУ Издательства *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_издательства )
           ПОЛЯ ( Код_издательства Целое, Название
                  Текст 40, Город Текст 25 );
СОЗДАТЬ ТАБЛИЦУ Заглавия *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_заглавия )
           ПОЛЯ ( Код_заглавия Целое, Заглавие Запись );
СОЗДАТЬ ТАБЛИЦУ Вид_издания *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Вид_издания )
           ПОЛЯ ( Вид_издания Целое, Название_вида Текст 16);
СОЗДАТЬ ТАБЛИЦУ Характеры *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_характера )
           ПОЛЯ ( Код_характера Целое, Характер_переиздания Текст 16 );
СОЗДАТЬ ТАБЛИЦУ Языки *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_языка )
           ПОЛЯ ( Код_языка Целое, Язык Текст 16, Сокращение Текст 6 );
СОЗДАТЬ ТАБЛИЦУ Места *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_места )
           ПОЛЯ ( Код_места Целое, Номер_комнаты Целое,
                  Номер_стелажа Целое, Номер_полки Целое );
СОЗДАТЬ ТАБЛИЦУ Читатели *( Стержневая сущность )
 ПЕРВИЧНЫЙ КЛЮЧ ( Ном_билета )
```

```
            ПОЛЯ ( Ном_билета Целое, Фамилия Текст 20, Имя Текст 16,
                   Отчество Текст 20, Адрес Текст 60, Телефон Текст 9 );
СОЗДАТЬ ТАБЛИЦУ Издание *( Обозначение )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_издания )
   ВНЕШНИЙ КЛЮЧ ( Код_заглавия ИЗ Заглавия
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Заглавия ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Заглавия.Код_заглавия ОГРАНИЧИВАЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Вид_издания ИЗ Вид_издания
                  NULL-значения ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Вид_издания ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Вид_издания.Вид_издания КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Код_издательства ИЗ Издательства
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Издательства ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Издательства.Код_издательства КАСКАДИРУЕТСЯ)
            ПОЛЯ ( Код_издания Целое, Код_заглавия Целое,
                   Вид_издания Текст 16, Номер_тома Целое,
                   Авторский_знак Текст 3, Библиотечн_шифр Текст 12,
                   Повторность Целое, Код_издательст- ва Целое,
                   Год_издания Целое )
    ОГРАНИЧЕНИЯ ( 1. Значения полей Код_заглавия, Вид_издания
                  и Код_издательства должны принадлежать набору значений
                  соответствующих полей таблиц Заглавия, Вид_издания
                  и Издательства; при нарушении вывод сообщения "Такого
                  заглавия нет", "Такого вида издания нет" или "Такого
                  издательства нет". );
СОЗДАТЬ ТАБЛИЦУ Переплеты *( Обозначение )
 ПЕРВИЧНЫЙ КЛЮЧ ( Номер_переплета )
   ВНЕШНИЙ КЛЮЧ ( Код_издания ИЗ Издания
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Издания ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Издания.Код_издания КАСКАДИРУЕТСЯ)
            ПОЛЯ ( Номер_переплета Целое, Код_издания Целое, Цена Деньги,
                   Дата_приобретения Дата )
    ОГРАНИЧЕНИЯ ( Значения поля Код_издания должны принадлежать набору
                  значений соответствующего поля таблицы Издания;
                  при нарушении вывод сообщения "Такого издания нет" );
СОЗДАТЬ ТАБЛИЦУ Аннотации *( Характеризует Издания )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_издания )
   ВНЕШНИЙ КЛЮЧ ( Код_издания ИЗ Издания
                  NULL-значения ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Издания ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Издания.Код_издания КАСКАДИРУЕТСЯ)
            ПОЛЯ ( Код_издания Целое, Аннотация Запись )
    ОГРАНИЧЕНИЯ ( Значения поля Код_издания должны принадлежать набору
                  значений соответствующего поля таблицы Издания;
                  при нарушении вывод сообщения "Такого издания нет" );
СОЗДАТЬ ТАБЛИЦУ Авторы *( Связывает Создатели и Издания )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_создателя, Код_издания )
   ВНЕШНИЙ КЛЮЧ ( Код_создателя ИЗ Создатели
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Создатели ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Создатели.Код_создателя КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Код_издания ИЗ Издания
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Издания ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Издания.Код_издания КАСКАДИРУЕТСЯ)
            ПОЛЯ ( Код_создателя Целое, Код_издания Целое )
    ОГРАНИЧЕНИЯ ( Значения полей Код_создателя и Код_издания должны
                  принадлежать набору значений соответствующих полей
                  таблиц Создатели и Издание; при нарушении вывод
                  сообщения "Такого автора нет" или "Такого издания нет" );
```

Аналогичное содержание имеют описания таблиц Составители, Редакторы, Художники и Переиздания. Остальные же таблицы проектируемой базы данных описываются так:

```
СОЗДАТЬ ТАБЛИЦУ Переводчики *( Связывает Создатели, Издания и Языки)
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_создателя, Код_издания )
   ВНЕШНИЙ КЛЮЧ ( Код_создателя ИЗ Создатели
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Создатели ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Создатели.Код_создателя КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Код_издания ИЗ Издания
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Издания ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Издания.Код_издания КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Код_языка ИЗ Языки
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Языки ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Языки.Код_языка КАСКАДИРУЕТСЯ)
           ПОЛЯ ( Код_создателя Целое, Код_издания Целое )
    ОГРАНИЧЕНИЯ ( Значения полей Код_создателя, Код_издания и
                  Код_языка должны принадлежать набору значений
                  соответствующих полей таблиц Создатели, Издание
                  и Языки; при нарушении вывод сообщения "Такого
                  автора нет" или "Такого издания нет" или "Такого
                  языка нет");
СОЗДАТЬ ТАБЛИЦУ Размещение *( Связывает Места и Переплеты )
 ПЕРВИЧНЫЙ КЛЮЧ ( Код_места, Номер_переплета )
   ВНЕШНИЙ КЛЮЧ ( Код_места ИЗ Места
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Места ОГРАНИЧИВАЕТСЯ
                  ОБНОВЛЕНИЕ Места.Код_места КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Номер_переплета ИЗ Переплеты
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Переплеты КАСКАДИРУЕТСЯ
                  ОБНОВЛЕНИЕ Переплеты.Ном_переплета КАСКАДИРУЕТСЯ)
           ПОЛЯ ( Код_места Целое, Номер_переплета Целое,
                  Дата_размещения Дата, Дата_изъятия Дата )
    ОГРАНИЧЕНИЯ ( Значения полей Код_места и Номер_переплета
                  должны принадлежать набору значений соответствующих
                  полей таблиц Переплеты и Места; при нарушении вывод
                  сообщения "Такого переплета нет" или "Такого места нет" );
СОЗДАТЬ ТАБЛИЦУ Выдача *( Связывает Читатели и Переплеты )
 ПЕРВИЧНЫЙ КЛЮЧ ( Ном_билета, Ном_переплета )
   ВНЕШНИЙ КЛЮЧ ( Ном_билета ИЗ Читатели
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Читатели КАСКАДИРУЕТСЯ
                  ОБНОВЛЕНИЕ Читатели.Ном_билета КАСКАДИРУЕТСЯ)
   ВНЕШНИЙ КЛЮЧ ( Ном_переплета ИЗ Переплеты
                  NULL-значения НЕ ДОПУСТИМЫ
                  УДАЛЕНИЕ ИЗ Переплеты КАСКАДИРУЕТСЯ
                  ОБНОВЛЕНИЕ Переплеты.Ном_переплета КАСКАДИРУЕТСЯ)
           ПОЛЯ ( Ном_билета Целое, Ном_переплета Целое, Дата_выдачи Дата,
                  Срок Целое, Дата_возврата Дата )
    ОГРАНИЧЕНИЯ ( Значения полей Ном_билета и Ном_переплета должны
                  принадлежать набору значений соответствующих полей таблиц
                  Читатели и Переплеты; при нарушении вывод сообщения
                  "Такого читателя нет" или  "Такого переплета нет" );
```

Теперь следует проверить, не нарушены ли в данном прокете какие-либо принципы нормализации (п. 4.6), т.е. что любое неключевое поле каждой таблицы:

- функционально зависит от полного первичного ключа, а не от его части (если ключ составной);

- не имеет функциональной зависимости от другого неключевого поля.
- Сущности Авторы, Составители, Редакторы, Художники и Переиздания, не имеющие неключевых полей, безусловно нормализованы. Нормализованы и сущности Создатели, Характеры, Заглавия, Вид_издания и Аннотации, состоящие из несоставного ключа и единственного неключевого поля.

Анализ сущностей Переводчики, Размещение и Выдача, состоящих из составного ключа и неключевых полей, показал, что в них нет функциональных связей между неключевыми полями. Последние же не зависят функционально от какой-либо части составного ключа.

Наконец, анализ сущностей Издания, Переплеты, Места, Читатели и Языки, показал, что единственной "подозрительной" сущностью является стержень Языки, имеющий два функционально связанных неключевых поля: Язык и Сокращение.

Поле Язык стало неключевым из-за ввода цифрового первичного ключа Код_языка, заменяющего текстовый возможный ключ Язык. Это позволило уменьшить объем хранимых данных в таблице Переводчики, затраты труда на ввод множества текстовых значений и возможной противоречивости, которая часто возникает из-за ввода в разные поля ошибочных дубликатов (например, "Английский", "Англиский", "Анлийский", "Англйский" и т.п.). Если мы вспомним рекомендации п. 4.5 о замене на время нормализации цифровыз заменителей первичных ключей (Код_языка) на исходный ключ (Язык) или воспользуемся формулировкой НФБК, то окажется, что таблица Языки - нормализована.

Для завершения проекта необходимо было бы ввести в описания таблиц дополнительные сведения об ограничениях целостности (выше указан лишь минимальный их набор) и дать описание некоторых таблиц, но ограниченнный объем публикации не позволяет включать эти подробности, не являющиеся принципиальными для иллюстрации процедуры проектирования.

ЛИТЕРАТУРА

1. Атре Ш. Структурный подход к организации баз данных. - М.: Финансы и статистика, 1983. - 320 с.
2. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. - М.: Финансы и статистика, 1989. - 351 с.
3. Дейт К. Руководство по реляционной СУБД DB2. - М.: Финансы и статистика, 1988. - 320 с.
4. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ. -М.: Мир, 1991. - 252 с.
5. Кириллов В.В. Структуризованный язык запросов (SQL). - СПб.: ИТМО, 1994. - 80 с.
6. Мартин Дж. Планирование развития автоматизированных систем. - М.: Финансы и статистика, 1984. - 196 с.
7. Мейер М. Теория реляционных баз данных. - М.: Мир, 1987. - 608 с.
8. Тиори Т., Фрай Дж. Проектирование структур баз данных. В 2 кн., - М.: Мир, 1985. Кн. 1. - 287 с.: Кн. 2. - 320 с.
9. Ульман Дж. Базы данных на Паскале. - М.: Машиностроение, 1990. - 386 с.
10. Хаббард Дж. Автоматизированное проектирование баз данных. - М.: Мир, 1984. - 294 с.
11. Цикритизис Д., Лоховски Ф. Модели данных. - М.: Финансы и статистика, 1985. - 344 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- ассоциативная *
- обозначающая *
- стержневая *
- характеристическая *

Тело отношения *
Универсальное отношение *
Уникальность *
Функциональная зависимость *
Характеристика *
Целостность *
Цель *
Язык запросов по образцу (QBE) *
Язык инфологического моделирования (ЯИМ) *
Язык описания данных (ЯОД) *
Язык структурированных запросов (SQL) *
Язык "Таблица-связь" *